Numerical Approaches to Optimize Dispatch on Microgrids with Energy Storage

by

Lutao Xie

B.S. Mechanical Engineering
Union College, 2014

Submitted to the School of Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Signature redacted

Author_____
Center for Computational Engineering, MIT
May 6, 2016

Signature redacted

Certified by_____
W. Craig Carter
POSCO Professor of Materials Science and Engineering
Thesis Supervisor

Signature redacted

Accepted by_____
Nicolas Hadjiconstantinou
Professor of Mechanical Engineering Co-Director
Computation for Design and Optimization (CDO)

Numerical Approaches to Optimize Dispatch on Microgrids with Energy Storage

by

Lutao Xie

Submitted to the School of Engineering
on May 6, 2016 in partial fulfillment of the requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

Microgrids and distributed generation are predicted to become extremely dominant in developing nations, and will be largely beneficial to both electricity suppliers and consumers. With the penetration of renewable energy into the electricity supply, to maintain a balance between power supply and demand is becoming more difficult. Nevertheless, it is quite feasible that large electrical storage systems such as batteries can efficiently mitigate problems caused by the intermittency of renewables, and thus enable stable adoption of such power sources. In order to understand how the energy capacity and power characteristics of batteries should be specified to optimize economic or socio-economic benefits, an optimizing strategy for battery usage in microgrids energy scheduling was constructed. This strategy is based on the past power consumption, predictions of day-ahead power consumption, and historical trends of seasonal and daily trends, which gives a nonlinear, discontinuous and high dimensional objective function. Optimizing such an objective function is found to be very computational intensive and complex. In this paper, the nature of this large-scale optimization problem is studied. For real time dispatch, four optimization methods including active-set, interior-point method, sequential quadratic programming (SQP) and trust-region-reflective are discussed and compared to find the relatively fast and robust optimization algorithm. The computation was implemented by using the MATLAB nonlinear programming solver 'fmincon'. Three main objectives are carried out to improve the efficiency of solving this optimization problem: (1) determination of the mathematical& physical definitions of tolerances and discussion on convergence criteria with the corresponding tolerances; (2) Study and comparison on influences of the initial condition and the behavior of the objective function (highly related to peak demand charge); and (3) suggestions on modification of the model to achieve reduction of the computation time whilst maintain acceptable accuracy.

Thesis Supervisor: W. Craig Carter

Title: POSCO Professor of Materials Science and Engineering

Department: Materials Science and Engineering

Acknowledgements

I would like to express my sincere appreciation and gratitude to my thesis advisor, Dr. Craig W. Carter, for his full support and guidance during my graduate studies. I am truly thankful for the amount of help and time that professor Carter has contributed in assisting me to compete my graduate thesis. Without his great patience, understanding and encouragement through my study and research, this work would not have been possible. I am extremely grateful for the opportunity to collaborate with professor Carter over the past two years.

I would also like to express my special thanks to Dr. Marco Ferrara, who has also provided a large amount of support, assistance and dedication for this research project. I very much appreciate his insightful suggestions and guidance during the study. Thanks to his gracious sharing of thoughts and comments, I am able to encounter many of the difficulties and eventually make progresses.

Finally, I would like to thank the Center for Computational Engineering for offering me this precious opportunity to carry out my graduate study and further train myself through courses and research. I would also like to thank my friends for their encouragement and help, which made my stay and studies at MIT more enjoyable.

Contents

1 Introduction

1.1 Microgrids and Distributed Generation

The electric power system has been playing a significant role in delivering electric power from suppliers to consumers. It is no doubt that the grid has made essential contributions on almost every aspect of this world from the growth of worldwide industry to the maintenance of individuals' everyday life functions. In the United States, or in most countries in the world, the electricity power planning and operations are carried out through a centralized power grid system. There are many risks and issues with the current grid including the concerns of the reliability of the service, and environmental problems caused by traditional thermal energy source. For instance, local outrages can easily occur due to the damages of the distribution lines; blackouts in large regions can happen because of failures of transmission network; Starting up/shutting down thermal power plants is quite costly in terms of both time and money; and we cannot ignore the carbon pollution from power plants over time. In order to maintain the quality of service to the customers, instead of making a large amount of investment to improve the efficiency of the grid, implementing batteries may provide a more economic strategy for building grids in the developing worlds. In addition, with the large penetration of renewable energy and fast development of new technologies, electricity consumers are no longer satisfied with having the grid as the only source of the electricity power, but instead, are constructing more decentralized power generation stations nearby to add their own source of power supply, which are known as microgrids.

Microgrids are localized and small-scale grids that can be disconnected from the grid to provide electricity power on-site. Contrary to the traditional electricity power grid, microgrids allow autonomous operation with much more energy flexibility, cost-efficieny scheduling and largely reduce risks from the grid [1]. The power delivered by microgrids is noted as the distributed generation, which refers to small-scale power generation at or near the point of consumption. Renewable energy such as solar and wind power is the most popular distributed generation choice because of their environmental friendly characteristics and low or even zero cost of using such power source. Thermal sources

such as coal, gas and fossil fuel can also be used to support baseloads for distributed generation. Figure 1 shows the relationship of the grid, the microgrids and distributed generation in a brief drawing.

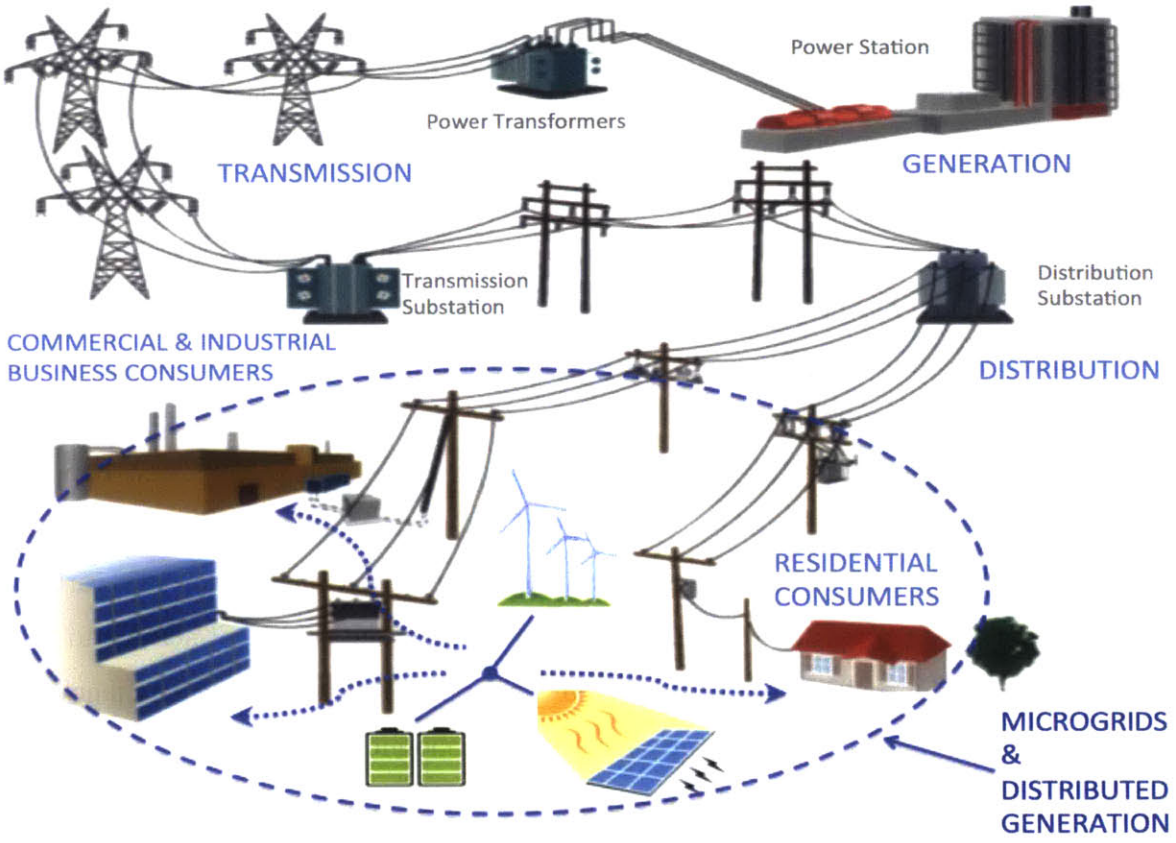


Figure 1. The relationship between centralized electric power grid and microgrids with distributed generation: after electricity is generated, the power is delivered through transmission and distribution network to local consumers through the grid; at the same time, the local consumers can also use the power provided by the microgrid with distributed generation and storage technique [2].

Due to the intermittency of renewable energy generation and thus the difficulty to make predictions on such power generation, energy storage techniques such as battery storage are often implemented with distributed generation. The microgrid, with a combination of renewable power generation and energy storage, has caught people's attention by its great advantages. For example, it is able to not only improve the efficiency but also enhance the reliability and boost the quality of the service from the power grid, especially with respect to congestion and power outages. In 2012 during the

Hurricane Sandy and its aftermath, a residential building in Greenwich Village was able to maintain its power, water and heat thanks to the microgrids [3]. Additionally, the microgrid is quite easy to build and can also reduce the carbon footprint. We certainly cannot ignore its large potential of promoting economic growth [4]. In deed, in the developing world, the microgrid will be extremely dominant in the near future.

1.2 Optimization of Dispatch of Microgrids Assets including Energy Storage

Renewable energy is considered as the source of the distributed generation for its low/zero variable cost and environmentally friendly characteristics, and here I choose to use solar power since it is very practical and easy to implement and has relatively the smallest environmental impacts of all renewables; unlike wind turbines, which require certain spaces and can cause noise pollution. Because of the intermittency of solar power generation, large electrical storage systems such as batteries are considered to mitigate such problem and enable stable adoption of renewables. Thus, with solar power generation, battery power storage, and the electric power supplied from the grid as the three main energy assets of the microgrid, we can construct an optimization problem by searching for the optimal dispatch of these energy assets described to minimize the cost of electric power usage. Solving this optimization problem can give us a better understanding of how the energy capacity and power characteristics of the batteries to optimize economic or socio-economic benefits. At the same time, it provides electricity consumers with a high degree of flexibility and efficiency in energy use as the demand side. Furthermore, successful utilization of such renewable energy source at microgrids will not only improve the quality of and reliability of the service at the local distribution network, but also reduce the carbon emission.

However, solving such an optimization problem is quite time-consuming and has certain computational challenges. It is observed that: (1) the objective function, which is also noted as the cost function that is to be minimized, is highly nonlinear, non-smooth and quite sensitive to the dispatch list; (2) it has very high dimensions and a large number of local minima; (3) to optimize such cost function is very computational intensive and complex. Thus the goal of this research is to improve the computation efficiency of solving this optimization problem by making suitable modifications of a model like this.

1.3 Specific Objectives of This Research

In this paper, to obtain a better understanding of the behavior of this optimization model, I first study and compare the applicability, cost and performance of the main four algorithms provided by MATLAB: active-set, interior-point method, sequential quadratic programming (SQP) and trust-region-reflective.

It is found that the convergence rate and accuracy of algorithms are highly related to the determination and uses of different tolerances, especially the convergence criteria on the change of step size and cost function values in iterations. The MATLAB mathematical formulations of calculating such tolerances for my specific case are determined by numerical experiments. The physical meaning of the tolerances and convergence criteria are studied and discussed.

In order to reduce the computation time without sacrificing much accuracy, impact of making changes on the objective function and initial conditions are analyzed and discussed. In particular, the solver is tested by using three types of initial condition: (1) perturbed flat initial condition; (2) guessed solution and (3) SQP approximated optimal solution. Furthermore, with flat initial condition, numerical experiments are performed with perturbed peak demand charge, which brings variations to the objective function. The tradeoffs between convergence accuracy and rate are discovered and suggestions on suitable modifications on such model are made to achieve the computation time.

2 The Microgrid Dispatch Optimization

2.1 Description of the Microgrid Dispatch Optimization Problem

In this paper, microgrids with solar power as the distributed generation and batteries as the storage technique installed are considered. I begin with learning the nature of this optimization problem. Knowing solar power penetration, the information about the grid and certain battery constraints, I aim to minimize the cost of consumers by solving the objective function for the optimal battery power dispatch. Equation (1) below gives a generation form of this cost function I am trying to optimize.

$$\text{Total Cost [\$]} = \text{Total Energy Cost [\$]} - \text{Energy Sales [\$]} \quad (1)$$

This objective function is constrained by batteries' state of charge (SOC), and is also bounded by the maximum and minimum rated power of the batteries. In addition, a list of some major inputs that may be used to construct the objective function for this optimization problem is shown in Table 1.

Table 1. A list of major inputs that can be considered to construct the objective function for the microgrid dispatch optimization. It includes the historical data from the grid, information about the renewables and the characteristics of batteries.

Type	**Unit**	**Definition**
Load	KW	Load data of the microgrid
Renewable	KW	Solar power generation for the microgrid
TOU Pricing	\$/KWh	Time-based pricing of the electric service, specifically Time-of-Use Pricing here
Peak Demand Charge	\$/KW-month	The rate for power usage during the peak hours
Renewable Incentives	\$/KWh	Compensation received for providing renewables generation back to the grids; dependent on the contract type: feed-in tariff (FIT) or net-metering
Time-Step Period	h	Time step for the data input is defined as 0.25h, equivalently 15 minutes
ACAC Efficiency	%	Discharge/Charge efficiency of batteries
SOC Constraints	KWh	The state of charge (SOC) constraints of batteries
Battery Rated-Power	KW	Rated power of batteries
Grid Availability	KW	The availability of the electricity power from the grid
...	...	...

Unlike the wind generation, the daily solar power generation may have some trends, but it can still be quite intermittent. An example of the hourly PV generation of ISO New England of each day for one week in July 2012 is shown in Figure 2. This can give us a general idea of how the solar generation would vary in general on daily bases.

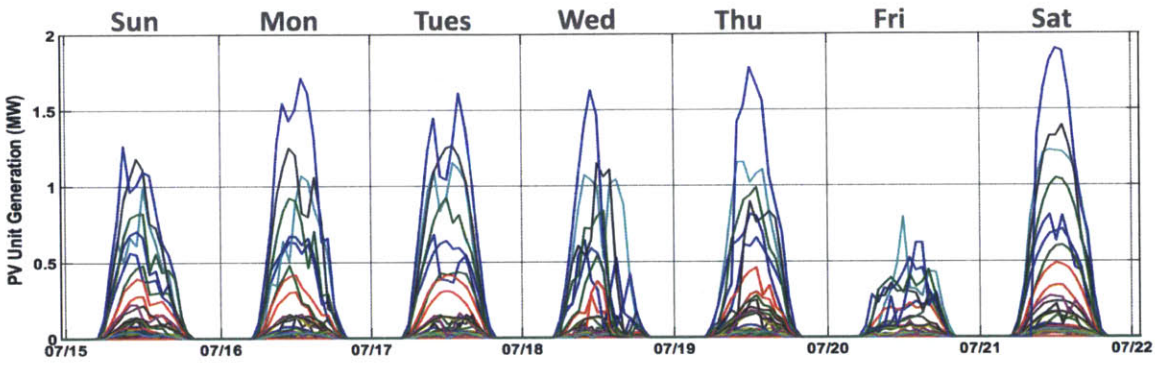
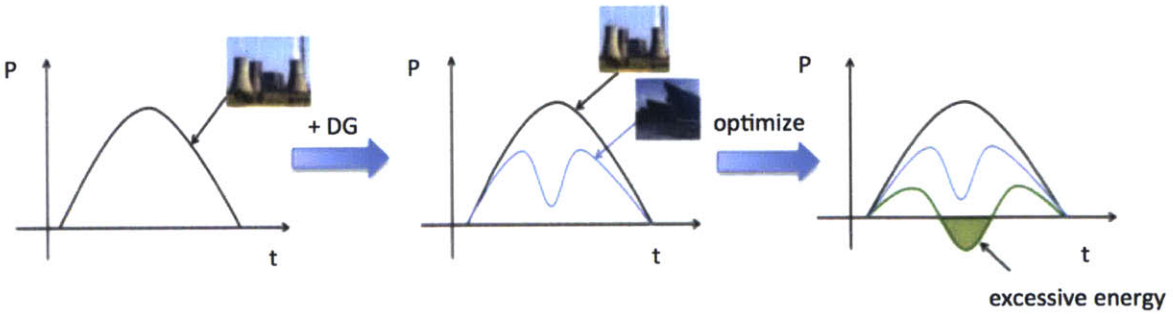


Figure 2. Hourly PV generation of ISO New England from July 15 to July 20 in 2012 [5].

In order to better understand this optimization dispatch problem, the expected load profile before and after being optimized is explained step by step as I add different energy assets to the microgrid. The first graph in Figure 3 shows a simplified load profile when only the electricity from the grid is supplying the power to the consumers. The area underneath the graph is the amount of energy that consumers are paying for. This is the case when there is no distributed generation supplied from the microgrid. Next, consider adding distributed generation (such as solar power) to the site, it is reasonable to expect the previous optimized load profile to act as the blue plot in the second graph in Figure 3, where a decent amount of energy bought from the grid is compensated by the renewable energy generated at the microgrid. If the load profile is fully optimized, I can even obtain excessive energy that may be sold back to the grid, shown as the green area below the x-axis in Figure 3 (3).



(1) (2) (3)

Figure 3. Simplified optimal load profile with different energy assets considered: (1) without any distributed generation, the load profile is shown as the black plot; (2) after the solar power being added, the load profile can be treated as the blue plot; (3) If the

load profile is fully optimized, the expected optimal load profile becomes the green plot, the green area underneath the x-axis stands for the excessive energy.

Figure 3 only shows the optimal load profile with distributed generation, and it is viable to achieve even better results with storage techniques added. Note that there are limitations and constraints that need to be considered when some storage technique (such as batteries) are applied to microgrids. Thus, an updated optimal load profile with battery constraints (such as SOC) included is shown in Figure 4. The resulted optimal load profile is represented as the red plot. With the batteries considered, a much flatter optimal load profile can be expected compared to the previous ones. Under this circumstance, adding batteries does not only enhance the flexibility of the energy assets dispatch, but also can largely save the cost of the energy and power usage. Note that for the purpose of better understanding this optimization problem, the graphs are highly simplified, whereas in the real situation, the load profile would be a lot more complicated.

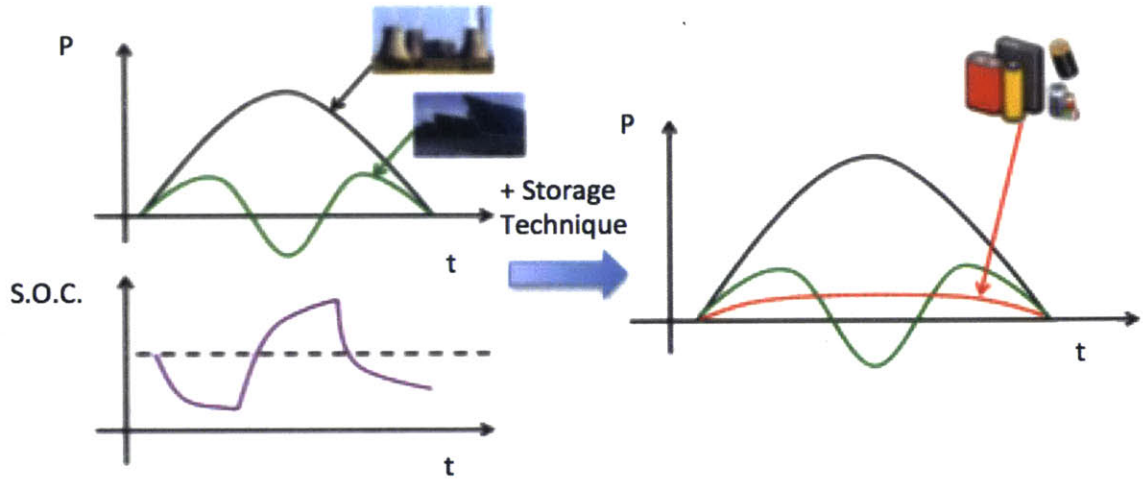


Figure 4. An updated simplified optimal load profile with the constraints of battery considered, shown as the red plot. With the SOC constraints considered, I will expect the optimal load profile to be a lot more flat compared to the previous ones.

2.2 Current Issues and Challenges

As shown in Figure 5, to solve this optimization problem, a guessed initial condition is always required to start the solving process (see section 3). In my case, the initial guess is actually the battery power dispatch, which can be represented as the SOC data. In most cases, this initial guess should be a feasible solution, which means that it

has to as least satisfy all the constraints of the problem. Then the initial condition will be put through some complex iterative algorithm, the goal of which is to minimize the cost. At each iteration, some feasible solution will be generated, and it will be sent back to the solver until the solution satisfy optimality conditions. Further details about the corresponding numerical algorithms are discussed in section 3.

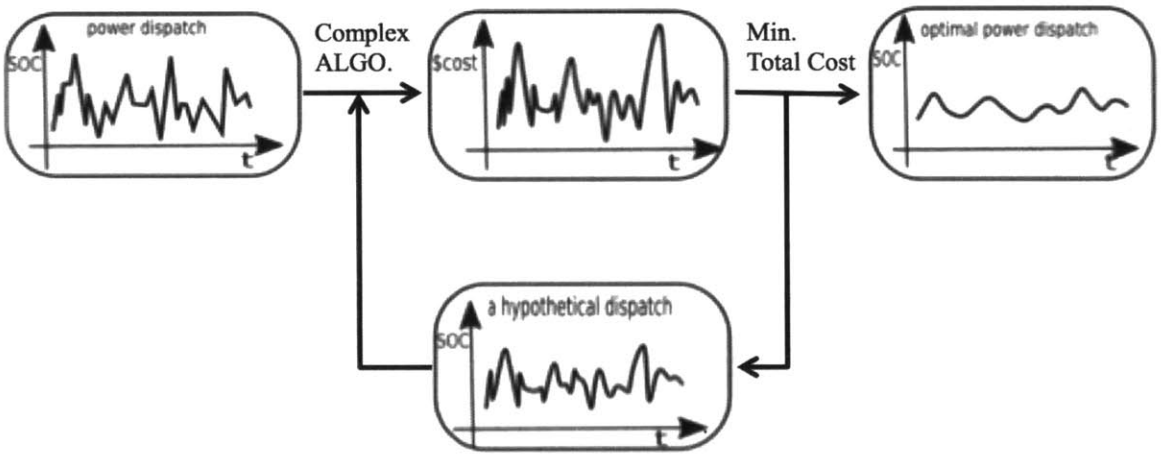


Figure 5. The major steps of solving the microgrid dispatch optimization problem.

The process shown in Figure 5 seems to be quite straightforward and ideal. However, to actually obtain the most optimal solution can be quite challenging and time-consuming. Figure 6 gives an example of the complication of the solving process in a simplified diagram. All the algorithms tend to break down the original problem into sub-problems and solve them through an iterative process (see section 3). Therefore, some feasible solution should always be obtained after each iteration, and this solution will be put back to the loop until the relatively most optimal solution is found. In general, a solution is considered to be optimal when it satisfies the optimality conditions by optimization theories (see section 3 and section 4.1). However, this does not simply mean that as the number of iterations gets bigger, the solutions will get better. Instead, the state of charge data obtained at each iteration can yield a completely different load profile and thus return very different cost results, which may not be a more minimized cost.

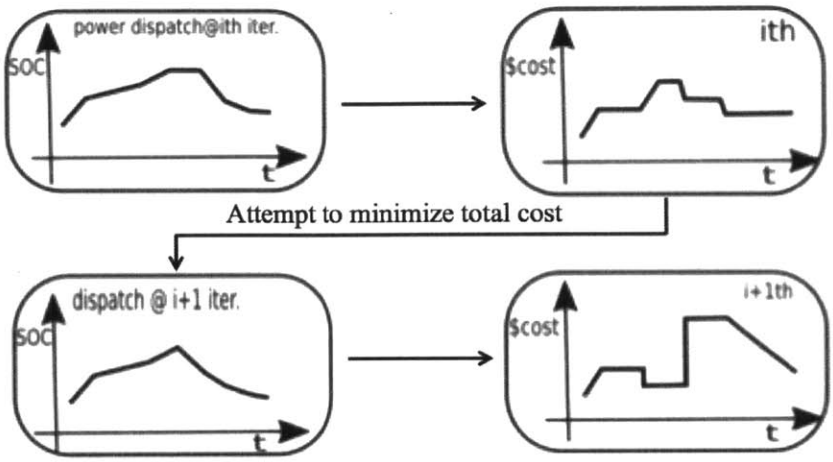


Figure 6. A simplified diagram showing the complication of the solving process.

One of the reasons is that the solver tends to jump around in the feasible region and it is quite difficult to actually control how it goes. Indeed, this optimization problem has very large dimensions and the fact that the objective function is highly nonlinear and non-smooth makes the solving process very complex and computational intensive. In addition, there exist a large number of local minima and it is almost impossible to find the global minimum. It is viable to finalize a relatively most optimal solution locally, but this can still take quite amount of time. Thus, the goal of this research is to improve the convergence accuracy and solving speed by making suitable modifications on the model. Suggestions are made in the end for saving computation time without sacrificing much accuracy of the result.

3 Numerical Methods on Solving the Problem

3.1 Optimization Algorithms in MATLAB

The nature of this optimization problem is described in the previous chapter. This chapter introduces the numerical methods that can be applied to solve it. Firstly, this problem is recognized as a constrained, large-scale, non-linear and non-smooth optimization problem. The process of solving this type of optimization problems is noted as Non-Linear Programming (NLP). A general form of the nonlinear minimization with equality and/or inequality constraints is shown below.

$$\begin{aligned} \text{minimize: } & f(x) \\ \text{subject to: } & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in X \end{aligned} \tag{2}$$

where $h: \mathbb{R}^n \rightarrow \mathbb{R}^r$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^r$

are the functions $h = (h_1, \ldots, h_n), g = (g_1, \ldots, g_m)$; n and m are positive integers; functions h and g represent the battery rated power constraints [KW] and SOC constraints [KW] respectively in this problem;

X is a subset of $\mathbb{R}^n$, standing for the optimal dispatch of the batter power in KW [6]; and the objective function f(x) simply is the total cost of energy and power usage in dollars.

There are a wide variety of common numerical methods available for solving different types of nonlinear optimization problems. In unconstrained nonlinear optimization, optimality conditions and gradient methods are generally used. Examples of gradient methods include Newton's method, gradient descent method, conjugate gradient method, and Gauss-Newton method. In constrained nonlinear optimization, I can also apply optimality conditions such as Karush-Kuhn-Tucker (KKT) conditions to solve for minima or maxima. Conditional gradient methods and gradient projection are also commonly used, and in more specific and complex situation, methods such as interior point method and cutting-plane method can be used [6].

The software MATLAB is chosen to solve my optimization problem and the built-in function 'fmincon' is used. There are mainly four numerical optimization algorithms that can be implemented: (1) Active-Set; (2) Sequential Quadratic Programming (SQP); (3) Interior Point method and (4) Trust Region Reflective. The general approach of the four methods is that they all tend to transform the original problem into simpler (most of the time quadratic) sub-problems and solve these sub-problems through an iterative process. Because they are solving the problem iteratively, an initial condition, which can also be treated as a possible solution, is required. The following paragraphs briefly summarize the main ideas of the four algorithms and their major characteristics.

Active-Set

As mentioned before, the original optimization problem is reformulated into quadratic sub-problems by approximation. A general active-set algorithm will start with a feasible solution to solve the approximated sub-problem defined by the active-set. The active-set at a feasible point x is composed of the active constraints at the current point. More specifically, given a point x in the feasible region, a constraint $g_i(x) \geq 0$ is said to be active at x if $g_i(x) = 0$ and inactive at x if $g_i(x) > 0$ [7]. In other words, the active-set algorithm solves a sequence of equality-constrained quadratic sub-problems iteratively [8].

The conventional active-set method is divided into two phases: (1) in phase one (for feasibility): a feasible point is found for the constraints $Ax = b$ and $Dx \geq f$;(2) in phase two (for optimality): the objective function f(x) is minimized through an iterative process while feasibility is maintained [3]. Note that phase one only concentrates on the feasibility and the objective function is not considered in this phase. In phase two, the optimal solution is usually obtained through an iterative process, during which the Lagrange multipliers are computed and the subset of the constraints with negative Lagrange multipliers will be removed. Then the infeasible constraints are searched and the process will be repeated until the results are optimal enough [9]. An optimal solution is often the solution that satisfies the optimality conditions by the optimization theories, which is discussed in section 4.1. In addition, a special type of active-set methods is called gradient projection method, which is used when the only constraints in the problem are bounds on the variables [7].

Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming (SQP) is known as one of the most effective iterative methods for nonlinear constrained optimization problem [7]. It is not only a generalization of Newton's method for unconstrained optimization problems, but also can be used in both line search and trust region frameworks [10]. In SQP algorithm, a quadratic programming (QP) sub-problem at the current iterate (x_k, λ_k) is generated by replacing the objective function with quadratic approximation. Consider the general form

of the nonlinear programming problem shown in equation (2), the corresponding quadratic sub-problem becomes

$$\begin{aligned} &\min_d \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, \mu_k) d \qquad (3) \\ &\text{subject to } h(x_k) + \nabla h(x_k)^T d = 0; \\ &\qquad g(x_k) + \nabla g(x_k)^T d \leq 0; \end{aligned}$$

Where the Lagrangian $\mathcal{L}(x, \lambda, \mu) := f(x) + \lambda^T h(x) + \mu^T g(x)$. [7][11]

By local convergence properties of the SQP method, it is assumed that the second condition holds when (x, λ) is close to the optimum (x^*, λ^*), and at the optimum solution, the second-order sufficient condition is satisfied [10]. Under this circumstance, the inequality constraints, which are also inactive at the solution x^* can be ignored, because when x_k is sufficiently close to x^*, the constraints should be the same and correctly identified [7]. Thus the previous formulation (3) can be restricted to the following form

$$\begin{aligned} &\min_d \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) d \qquad (4) \\ &\text{subject to } h(x_k) + \nabla h(x_k)^T d = 0. \qquad [10]\,[11] \end{aligned}$$

By solving this updated form of the sub-problem, the step d_k can be found. It can be treated as the updated search direction of the Newton's method, and is used to form the new iterate (x_k, λ_k). Therefore, let x_0 be the starting value. Then when $\|x_k - x^*\|$ is sufficiently small, the sequence of iterates converges quadratically to the optimal solution (x^*, λ^*) [11]. The SQP step computation can be too coarse/expensive in some cases, especially in early iterations, and it can be ineffective if the Hessian is not positive definite.

Note that the computation of Hessian matrix $\nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$ is required in this method and an approximation of such a positive definite matrix can be obtained by using quasi-Newton methods. Most implementations replace it with another matrix B_k, approximated and updated by using the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{5}$$

where $s_k = x_{k+1} - x_k; y_k = \nabla \mathcal{L}(x_{k+1}, \lambda_k) - \nabla \mathcal{L}(x_k, \lambda_k)$. [10]

Interior-Point Method

In interior point method, the original problem is transformed into a sequence of approximated minimization problems. The algorithm will start with iterating in the interior of the feasible set and then progressively approach the boundary (if the minimum does lie on the boundary). This is done by using a barrier function in the modified objective function that goes to infinity at the boundaries of the feasible region. The strength of this barrier function at each subsequent optimization will be reduced through iterations and consequently, the sequence of minima will approach arbitrarily closely toward the minimum of the original problem [12]. With the original form of the problem shown in equation (2), the approximate problem is

$$\begin{aligned} &\min_{x,s} f_\mu(x,s) = \min_{x,s} f_\mu(s) - \mu \textstyle\sum_i \ln(s_i) \\ &\text{subject to } h(x) = 0 \text{ and } g(x) + s = 0. \end{aligned} \tag{6}$$

where $\mu > 0$, and as $\mu \to 0$, the minimum of f_μ approaches to the minimum of f; s_i is the slack variable, which is restricted to be positive such that $\ln(s_i)$ is kept bounded; the number of s_i is equal to the number of inequality constraints g_i. Note that a slack variable is a variable added to the inequality constraint to transform it into an equality constraint [13].

This approximate problem stands for a sequence of equality-constrained problems with an added logarithmic term noted as the barrier function [13]. There are two main types of iteration steps for the algorithms provided by MATLAB: one is to take a direct Newton step in (x,s) by solving the KKT equations (mainly by linearized Lagrangian $\nabla_x \mathcal{L}(x, \lambda) = 0$; $\lambda_{g,i} g_i(x) = 0 \; \forall i$); and the other one is to take a Conjugate Gradient (CG) step using a trust region, subject to linearized constraints. By default, MATLAB will try to use the direct Newton step first, and then the CG step if the direct step fails. A special case is that when the approximate problem is not locally convex near the current iterate, MATLAB

will take the CG step [13]. Note that a merit function can be formulated by using a linear combination of the barrier function and the infeasibility, with a form such as $f_\mu(x,s) + v\|(h(x), g(x)+s)\|$. This merit function serves as a guidance of choosing step length at each iteration, where the value of this function should be decreased by the algorithm. In order to keep the solution feasible, the parameter v may increase with the iteration number. At each iteration, the algorithm decreases the merit function by the current attempted step. However, a different/new step will be started when the following two situation occur: (1) the current step fails to decrease the merit function and (2) the objective function returns a complex value, NaN, inf or an error at the current iteration [13].

Trust-Region Methods

In trust-region-reflective algorithm, a neighborhood around a feasible point is defined as the trust region if, in this neighborhood, an approximation of the objective function f can reflect the behavior of function f properly [13]. The trust-region sub-problem is to find the minima over the trust region, which can be defined as

$$\min_s q(s) \qquad (7)$$

$$\text{subject to } s \in N$$

where q is the approximated simpler function of the objective function f, and N is the trust region, a neighborhood around point x, in which q can reasonably reflect the behavior of the function f [13].

If the approximated function q fits the original function well (or if f(x+s)<f(x)), the trust region will be enlarged(or current point x be updated to be x+s), otherwise contracted[13]. Proper decisions and modifications on the approximation q and trust region N are quire essential in this method. MATLAB follows the standard trust-region method that is to define q by taking the first two terms of the Taylor approximation to the objective function at x, and the corresponding trust region N usually has the shape of

spherical or ellipsoidal around x [13]. According to MATLAB documentation, the mathematical definition is stated as

$$\min_s \frac{1}{2} s^T H s + s^T g \tag{8}$$

$$subject\ to\ \|Ds\|_2 \leq \Delta$$

where g and H are respecitively gradient and Hessian matrix of f at x; D is diagonal scaling matrix and Δ is a positive scalar [13].

There are good existing algorithms that can be applied to direct solve this problem to obtain an accurate solution. These algorithms require the computation of a full Eigen-system and/or a newton process to be applied to the secular equation $\frac{1}{\Delta} = \frac{1}{\|s\|}$ [13]. In this case, the computation time would be somewhat proportional to the factorization of H; but for large-scale problems, different approaches are needed including several approximation and heuristic strategies [13]. Specifically in MATLAB, the trust-region sub-problem is restricted to a 2D subspace S, which is determined with the aid of a preconditioned CG process. Trust-region methods is said to be effective on large-scale, sparse problems.

3.2 Discussion on the Numerical Methods

It is instantly noticed that the trust-region-reflective algorithm is not applicable to the optimization problem considered here because, in order to use this algorithm in MATLAB, either of the following two conditions needs to be satisfied: (1) a gradient of the objective function is supplied by users; or (2) the optimization problem has only bounds/linear constraints. The microgrids optimization problem, as described in section 2.1, does not satisfy such requirements. Therefore, starting from this section, the other three algorithms are discussed.

Active-set algorithm has the potential of speeding up the process by take large steps; and the complexity of searching the optimal solution can be reduced because that the estimation of the active-set region can provide us a subset of inequalities to watch

during the search [9]. This method is suitable for problems with non-smooth constraints. However, this method is designed to be effective for mostly small-and medium-sized problems, thus it is not a large-scale algorithm [8].

Sequential Quadratic Programming is quite a good fit for solving problems with significant nonlinearities and is able to provide a relatively reliable indication of the infeasibility. It is one of the most popular NLP algorithms because of its fast convergence properties from Newton's methods and its potential of being modified to apply to various problem structures. Since this method does not require the starting point to be feasible, it has can be used to capitalize a good initial starting point, which will be discussed in the section 4.2 [14]. In addition, the SQP algorithm seems to be similar to the active-set method but according to the MATLAB documentation, there are several differences between them that needed to be addressed: (1) in SQP, the iterative step taken in the region is constrained by bounds which are not exactly strict; (2) the objective function may returns a value of Inf, NaN, or a complex value in SQP since this algorithm can attempt to take a step that fails during its iterations; (3) To solve the quadratic programming sub-problems, the linear algebra routines used in the SQP algorithm is more efficient in both memory usage and speed than that in active-set method; (4) when constraints are not satisfied, there will be two new approaches to reformulate feasibility routines for SQP, which can slow down to solution of the sub-problems[13]. Lastly, the SQP algorithm provided in MATLAB is appropriate for both small and medium-sized problems. However, it is not quite effective for large-scaled problem either.

The interior-point method is developed recently and has become quite popular back to 1990s [7]. Both interior-point and active-set methods iterate within the interior of the feasible region, but note that active-set method does not use any barrier term to keep the algorithm in the interior [13]. Furthermore, each iteration of interior-point method can be more expensive than active-set because it solves linear systems with all variables involved whereas the active-set method only deals with some solve subset of the variables [8]. Nevertheless, interior-point method is able to satisfy bounds at all iterations and recover from NaN or Inf results. This algorithm is well suited for large, sparse problems as well as small dense problems [13]. It seems that interior-point method might be the best choice for this microgrids optimization problem for its applicability to large,

sparse problems. Further testing and comparison of these methods are performed; the corresponding observations and conclusions are shown in the following section.

3.3 Choosing the Algorithm: Application of Algorithms to the Microgrid Dispatch Optimization Problem

In order to compare the performance of the three algorithms (described in section three) on this specific optimization problem, the problem is simplified first to decrease some unnecessary degrees of freedom. More specifically, it is assumed that that there is no excessive renewable generation, which means that the net load before considering battery power is always positive. The time-of-use pricing is made to be constant, and the sales contract to incentivize renewables is chosen to be feed-in-tariff ('FIT'), which gives a constant input vector as well. In addition, assumtions are made that the ACAC-efficiency of batteries is ideal (100%) and the microgrid assets dispatch is only optimized for seven days instead of a year. In this way some complexity and confusions can be removed the results.

3.3.1 Comparison of Performance of Algorithms

The three algorithms active-set, interior-point and SQP are applied to solve the simplified problem respectively. The initial condition is kept to be flat (zero's) and the convergence tolerances are gradually increased from a very small number. For active-set method, the tightest tolerance is chosen to be 1e-5 (or 1e-3%) because it is already taking a large amount of time to converge at this convergence level and the returned optimal cost is "optimal enough". The results obtained at tolerance level lower than that is not likely to give us any more useful information. For the other two methods, the lowest tolerance is set to be 1e-8 (or 1e-6%). Note that the convergence tolerance here refers to the step tolerance in MATLAB, which implies the upper bound of the percent change of the solution between the last two consecutive iterations at convergence. The details regarding tolerances are discussed in section 4.1. The time and CPU usage are tracked, the corresponding results of which are shown in Figure 7. It is observed that at each tolerance level, active-set algorithm takes the longest time to converge and the SQP solver uses the least time (or CPU). The computation time decreases as the step tolerance

increases and it seems that the two terms are linearly related for interior-point and SQP methods.

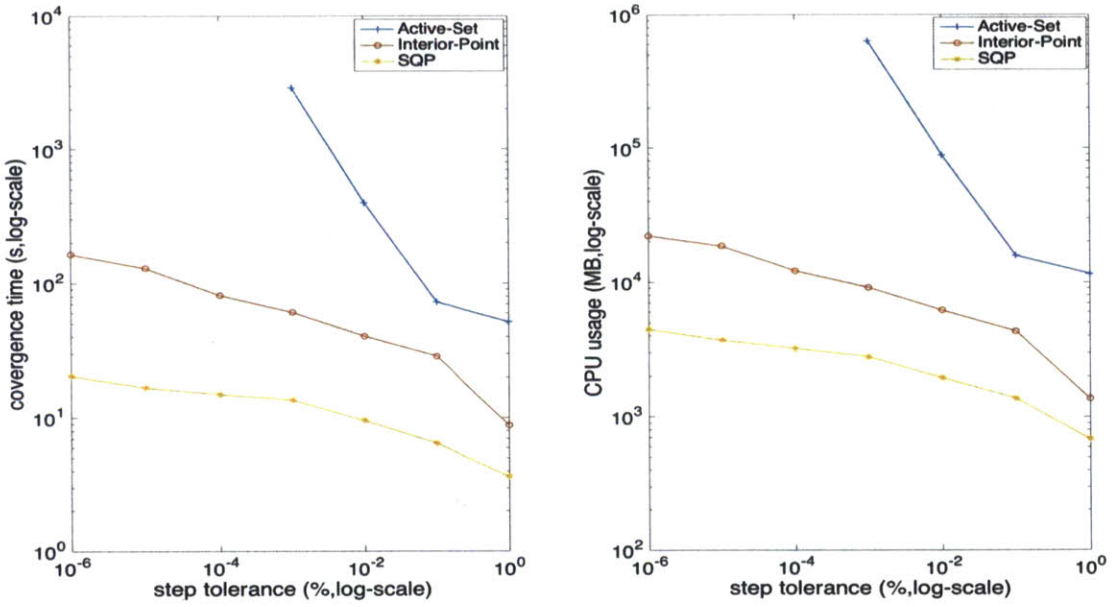


Figure 7. Plots of Time in seconds (left), CPU usage in MB (right) versus step tolerance (%); note that the step tolerance refers to the upper bound of the percent change of the solution between the last two consecutive iterations for convergence. More details about tolerances are introduced in section 4.1. Both x-axis and y-axis are in log-scale.

Moreover, the converged results, which is also the optimal cost obtained by using three methods under different tolerance settings are compared as well. At different tolerance levels for each algorithm, comparisons are made on the average daily optimal cost ($) and relative RMSE (relative root mean squared error in %) of the optimal costs obtained. The baseline for the relative RMSE results is the cost obtained with the tightest tolerance chosen here, which is at step tolerance equals to 1e-5(or 1e-3% for Active-Set algorithm) and 1e-8(or 1e-6% for the other two). This cost is considered as the most ideal cost I can obtain with each algorithm. The relative RMSE of the optimal cost obtained with step tolerances higher than that "ideal" result is computed and plotted as a function of the step tolerance. Both plots of the averaged daily optimal cost and relative RMSE of the cost as a function of the step tolerance are shown in Figure 8.

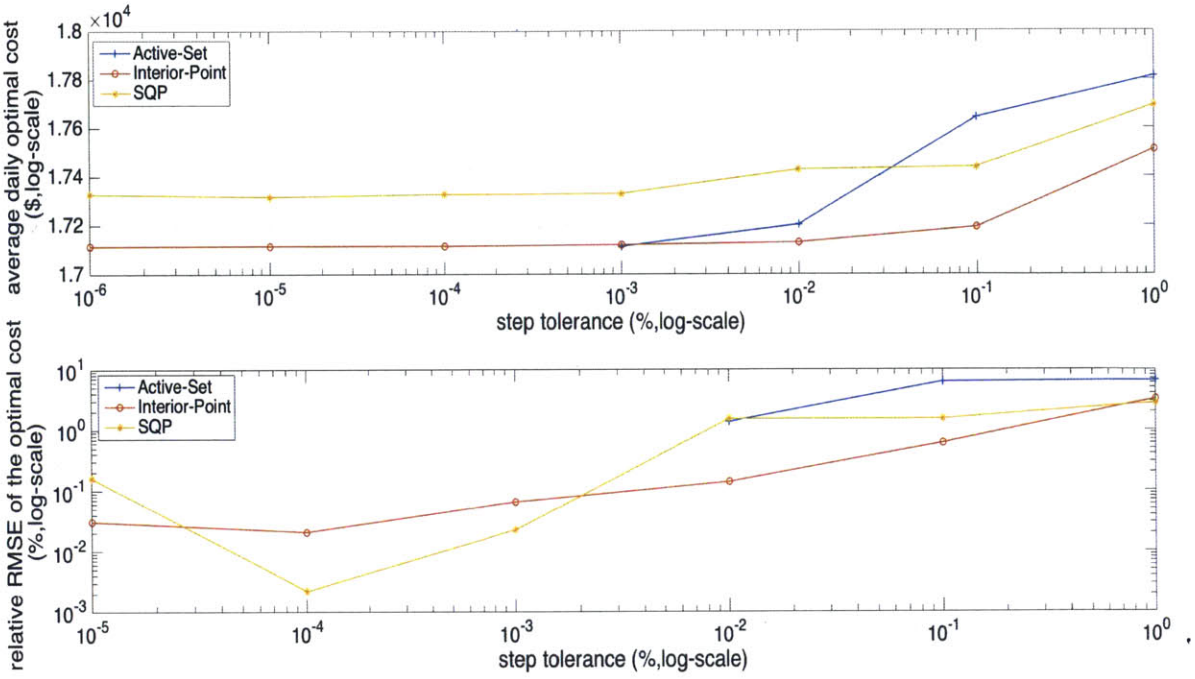


Figure 8. The graph at the top shows the plots of average daily optimal costs obtained by using each algorithm as a function of step tolerance; the bottom one shows the relative RMSE of the optimal cost as a function of step tolerance for each algorithm. Note that both x-axis and y-axis are in log-scale.

The plots of average daily costs versus step tolerance in Figure 8 show that the interior-point method gives us the lowest cost of all, and it seems that SQP algorithm is not doing as well as the other two for it has relatively bigger optimal costs regardless of the tolerance level. Furthermore, when the interior-point method was applied, as I increase the value of step tolerances, the relative RMSE of the optimal cost does not change (or increase) as rapidly as the other two and it appears to be linearly related to the step tolerance. Last but not the least, the results obtained from using SQP algorithm does not seem to be very stable, and it is observed that lower tolerance value does not always guarantee a better optimal solution.

3.3.2 Discussion and Conclusion

According to the results shown in the previous section, the SQP algorithm seems to take the shortest time. However, the optimal cost given by using SQP does not seem to be as optimal as those from the other two algorithms and the convergence accuracy seem to be quite unstable. Active-set algorithm takes the greatest amount of time to converge and

the optimal costs do not seem to get much better for taking that excessive amount time comparing to the other two algorithms. One reason for the slow convergence rate of active-set solver could be because of the nature of this algorithm. It tends to jump around in the feasible region formed by a combination of subsets of active constraints. . It may be a good algorithm for convex optimization problem, but does not seem to work well for this case (which is much more complex and has very high dimensions). After comparing the results, it is concluded that interior-point is the best fit for solving this optimization problem because of its relatively fast convergence speed and quite decent performance on minimizing the cost. This result also confirmed the guess from studying the corresponding theories in section 3.2.

4 Discussion on Convergence Rate and Accuracy

In the previous section, it is decided to use interior-point algorithm to solve this microgrid dispatch optimization problem. Recall that the problem is simplified to extract information for comparing algorithms previously. Perhaps here, to identify the effects of adding complexity to the problem, the simplifying assumptions will be removed iteratively. The Time-of-Use price is no longer constant and I may add some variations to the peak demand charge depending on the situation. In order to get more straightforward information about the behavior of the objective function itself, the renewable generation remains nonnegative and the battery conditions, especially the ACAC-efficiency and rated power, are still kept to be ideal. The time period is still seven days in most cases instead of a year for easier observation of the results, except that in section 4.2.3 and section 4.2.4, the optimal costs of 30 days are used for comparisons. Three aspects that may affect the convergence speed and accuracy of solving such a problem are studied in the following three sub-sections: (1) the definition and applications of different tolerances; (2) the influence of initial conditions; (3) the impact of behavior of the objective function. Results corresponding to each topic are summarized and discussed, and conclusions and suggestions are drawn in the end.

4.1 Tolerances and Stopping Criterion of the Solver

4.1.1 Definition of Different Tolerances

There are a number of parameters and different types of tolerances that are directly related to the termination of the iterative solving process. A list of the generation definition of each tolerance term from MATLAB is shown in Table 2. Note that the default values stated are just the recommended values for using interior-point algorithm according to MATLAB. It does not necessarily mean that they are the best choices for this microgrid optimization problem. Terms with similar functions are paired up and analyzed in the following paragraphs.

Table 1. The definitions of some parameters for solver options from MATLAB [15].

Parameters	MATLAB Definition	Default Value
MaxFunctionEvaluations	"The maximum number of function evaluations allowed, a positive integer"	3000 [no. of evaluations]
MaxIterations	"The maximum number of iterations allowed, apositive integer."	1000 [no. of iterations]
OptimalityTolerance	"Termination tolerance on the first-order optimality, a positive scalar"	1.00E-06 [$/KW]
ConstraintTolerance (TolCon)	"Tolerance on the constraint violation, a positive scalar. "	1.00E-06 [KW]
StepTolerance (TolX)	"Termination tolerance on x, a positive scalar."	1.00E-10 [% in step-change]
FunctionTolerance (TolFun)	"Termination tolerance on the function value, a positive scalar. "	1.00E-06 [$]

MaxFunctionEvaluations & MaxIterations

As shown in Table 2, the MaxFunctionEvaluations and MaxIterations are just the upper bounds for the numbers of function evaluations and iterations during the iterative process. The default values shown are for most of the commonly seen and nicely structured optimization problems. However, they are not quite suitable for this case of microgrids optimization. It is essential to understand that these two terms will limit the number of iterations and function evaluations, and inaccurate results may be obtained if they are too small. In other words, if they are not kept sufficiently big enough, the

algorithm will be stopped before it actually converges to the "true" solution because either the iteration numbers or the times the solver evaluates the functions may reach to the limits. In order to prevent this to happen and ensure the actual convergence of the solver, they are set to be large enough that the solver does not terminate due to either of these two limits. On the other hand, these two terms should not be set too large, which may cause the solver to run excessive number of iterations and take too much time to converge. Under this circumstance, I decide to gradually increase the values of MaxFunctionEvaluations and MaxIterations and record the termination messages provided by MATLAB. The termination messages tell us the reason why the solver stops, noted as exit message in MATLAB. The exit message is indicated by "exitflag" and returned as positive integers [16]. When the "exitflag" returns 0, it means that the algorithms stops due to the limitations of these two terms and the values of either or both terms should be increased. In this microgrids optimization problem, the model is tested with different values and both the MaxFunctionEvaluations and MaxIterations are decided to be 1e6, at which I am guaranteed to have converged results without spending extra time.

OptimalityTolerance & ConstraintTolerance

When a possible optimal solution is detected and the solver will then terminate. At this point, MATLAB will give two termination messages: (1) "local minimum possible" and (2) "local minimum found". The first case simply just means that the returned solution is most likely to be an optimal solution but may not be exact. In most situations, the solver stops because it cannot decrease the objective function or the step size anymore during the iterative process [17]. The second case is the most ideal case, which means that the analytical optimal conditions are satisfied within the specified tolerances. In this case, the "exitflag" given by MATLAB will return 1 and this is the only case that an (local) optimum solution is guaranteed [17]. This is where the OptimalityTolerance and ConstraintTolerance come into play. The local minimum can be found only if both of these tolerances are satisfied by certain conditions, which is also corresponding to the theories of optimality conditions.

The ConstraintTolerance is responsible for the feasibility of the solution, which in MATLAB is the upper/ lower bound on the magnitude of all the constraints [18]. This tolerance does not directly decide whether the solver stops or not. When it is violated, the MATLAB will make sure that the violation is less than this tolerance [16]. Therefore, even if the solution may violate this bound; the solution can still be treated to be feasible since the ConstraintTolerance is always set to be very small compared to zero, which means that the violation is so small that the solution can still be treated as feasible solution. The default value given is quite sufficient for the majority of the problems, so is good for this case of microgrids optimization. This is because that in my case the solution is in the range of about 50KW to 200KW, and the default value (1e-6 KW) of the ConstraintTolerance is approximately zero, which is sufficiently small for my solution.

The OptimalityTolerance is defined as the necessary first-order optimality conditions by either unconstrained or constrained optimality theories, depending on the type of the problem. For unconstrained optimization problem, the first-order optimality is measured by the infinite norm of the gradient of the objective function; and for constrained problems, the first-order optimality is defined by the Karush-KughlTucker (KKT) conditions, which uses the Lagrangian function (see equation (11)) [19]. Therefore, if the first-order optimality described is less than the OptimalityTolerance, the solver stops, and this is this case when the optimum solution is found. Since in order to achieve optimality conditions, the first-order optimality is required to be close/equal to zero, this OptimalityTolerance should always set to be very small compared to zero regardless of what problems are trying to solve. Similar to the value of ConstraintTolerance, the default MATLAB value (1e-6 $/KW) is already small enough compared to the scale of the gradient change of my solutions (~100$/KW). Therefore this value is kept the same for this microgrids optimization problem. Note that there is not much space here to vary the values of the ConstraintTolerance and OptimalityTolerance, because they should always be kept approximately zero to ensure an optimal feasible solution, unless there are special requirements. I do not consider changing the values of these two terms to achieve convergence speed because making them large may save large amount of time but will give very inaccurate or even false results.

StepTolerance & FunctionTolerance

StepTolerance and FunctionTolerance are highly related to the convergence rate and accuracy for the iterative solving process. Especially in this case of microgrids optimization, it is quite hard to satisfy both ConstraintTolerance and OptimalityTolerance conditions to obtain an exact numerical-approximated optimal solution. It is observed that most of the optimal solutions belong to case (1), where the solver stops because it cannot decrease either the objective function or the step size anymore [17]. In MATLAB, the StepTolerance is defined as the lower bound on the size of a step at each iteration, which is the infinite norm of $(\vec{x}_i - \vec{x}_{i-1})$. The solver stops when

$$\frac{|\vec{x}_i - \vec{x}_{i-1}|}{|\vec{x}_i|} \leq StepTolerance, \tag{8}$$

where $\vec{x}_i$ and $\vec{x}_{i-1}$ represent the solution obtained at two consecutive iterations i and i-1; $|\cdot|$ stands for infinite norm and this notation applies to all equations in this paper. Here a relative measure is used for comparisons [18]. Indeed, the detailed results are tracked and extracted for each iteration. After testing the interior-point algorithm with a simple function, it is found that the stopping criterion for the step size agrees with equation (9) without using any relative bound. Similarly, FunctionTolerance is a lower bound on the difference in objective function values during a step [4]. The stopping criteria related to this tolerance does not use a relative measure either, but simply follows equation (10) as

$$|f(\vec{x}_i) - f(\vec{x}_{i-1})| \leq FunctionTolerance, \tag{11}$$

where $f(\vec{x}_i)\ and\ f(\vec{x}_{i-1})$ represent the values of the objective function at two consecutive iterations i and i-1[18]. Note that the FunctionTolerance can also be treated as the upper bound of $\left|\frac{df(x)}{dx}\right|$, the infinite norm of the gradient of the objective function. This is usually applied to unconstrained optimization problems.

In addition, for more complex constrained optimization problems, the FunctionTolerance is also compared to the first-order optimality, which is the maximum absolute values of

the following two terms, obtained by applying KKT condition to the Lagrangian function [19],

$$|\nabla_x \mathcal{L}(x,\lambda)| := |\nabla f(x) + \lambda^T \nabla g(x) + \mu^T \nabla h(x)|, \text{and } |\lambda^T g(x)| \qquad (12)$$

Where $\mathcal{L}(x,\lambda)$is the auxiliary Lagrangian function; g(x) corresponds to the inequality constraints; h(x) represents the equality constraints; λ and μ are simply Lagrange multipliers.

Considering this microgrids optimization problem, unlike the other four types of tolerances, the choices of the StepTolerance, as well as the FunctionTolerance are largely related to the convergence time and the accuracy of the optimal solution. The relationship between FunctionTolerance and StepTolerance are determined below. The discussion about choices of StepTolerances and the trade-offs between the convergence rate and accuracy is carried out in the next section.

4.1.2 Results and Discussion on the Application of Tolerances

The StepTolerance represents the relative change of the solution x at last two consecutive iterations, and FunctionTolerance can be treated as the infinite norm of the gradient of the objective function. In order to discover the impact of these tolerances on this microgrid dispatch optimization problem, it is significant to understand the physical meaning of the stopping criteria to the actual problem.

It is known that the goal is to solve for the dispatch of the battery power by minimizing the energy cost. Therefore, here $\vec{x}$, represents the battery power in KW and the energy cost stands for the objective function f(x) that is to be minimized, with units in dollars. By referring to the Mathematical definitions, it is quite straightforward that the StepTolerance is comparable to the percent change in battery power during one step in the iterative process. Since the cost is calculated by summing up the cost of energy use and the power usage, the relationship between the StepTolerance (TolX) and FunctionTolerance (TolFun) in this optimization problem can be defined as

$$Tolfun = TolX * (\alpha * |EnergyPrice| * dt + \beta * |PowerPrice|) \qquad (13)$$

Where TolX and TolFun are MATLAB notations for FunctionTolerance and StepTolerance with units dollar [$] and percent [%] in change of power KW repectively; EnergyPrice is simply the price for energy-use in [$/KWh]; PowerPrice is the price for use of power in [$/KW]; dt is the step time period, which equals to 0.25 h (or 15 minutes); α and β are just positive constants such that $\alpha + \beta = 1$.

After the relationship between the two tolerances and most importantly, the physical connections between tolerances and this microgrids optimization problem is discovered, different tests can be performed with different tolerance value set-ups. Since the FunctionTolerance is dependent on the StepTolerance and can be obtained by using equation (13) in this case, I can reduce one variable and only look at the StepTolerance. The value of StepTolerance is increased from 1e-8 (1e-6% KW) to 1e-2 (1% KW). The corresponding computation time and the obtained optimal cost are recorded. The plots of optimal cost versus number of days for different StepTolerance level is shown in Figure 9, which can give us a sense of the behavior and magnitude of the optimal cost profile.

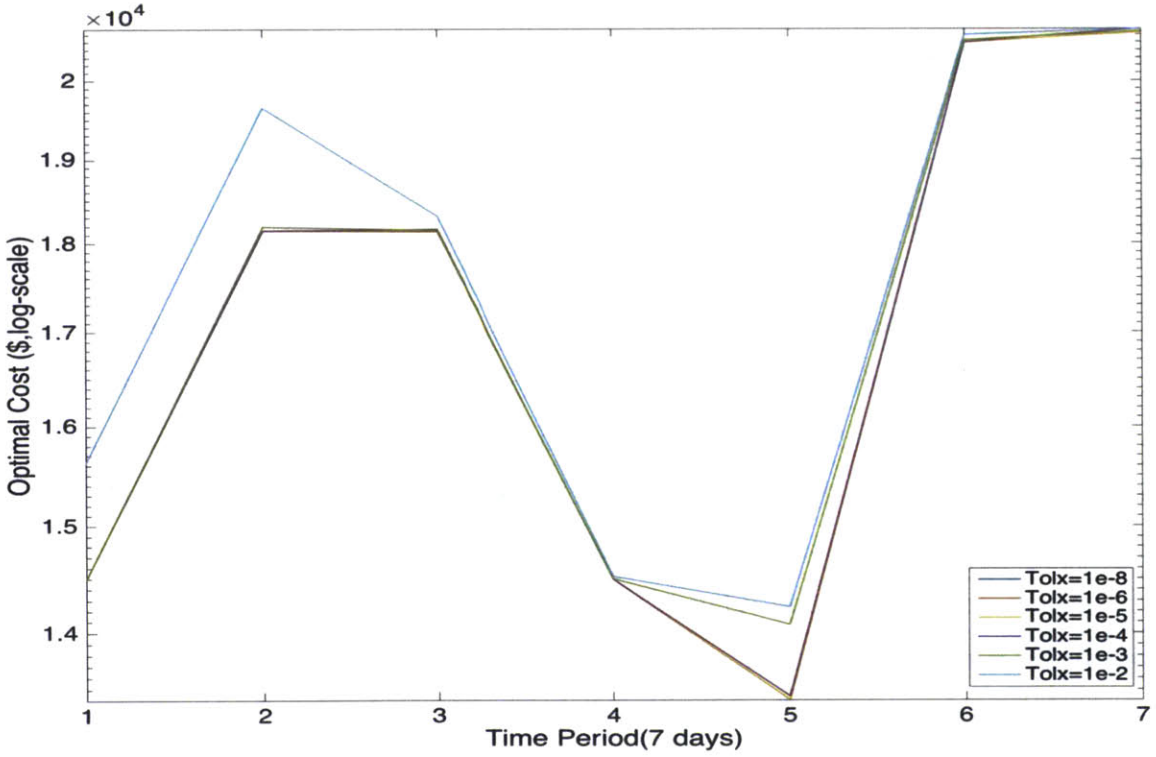


Figure 9. The plot of optimal cost obtained at different StepTolerance level versus time period in 7 days.

In order to better observe the trends and obtain more insightful information, the relative RMSE (Root Mean Squared Error) of the cost is estimated. Note that the optimal cost obtained at StepTolerance equals to 1e-8 (1e-6%) is set to be the baseline for calculating relative RMSE here, and this is considered as the most ideal optimal solution. A plot of the computation time as a function of the relative RMSE of optimal cost is created and shown in Figure 10.

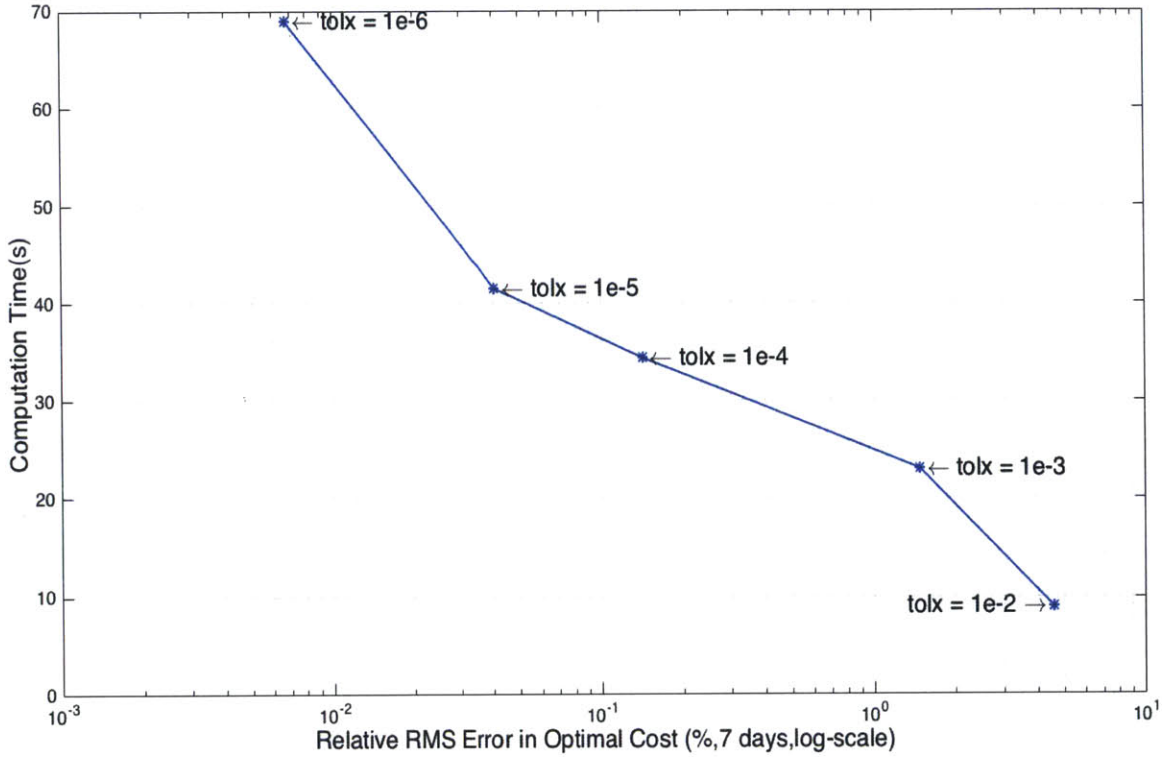


Figure 10. The plot of computation time as a function of relative RMSE of the optimal cost indicating the trade-off between computation time and convergence accuracy. The StepTolerance value for each plot is indicated and the optimal cost is optimized for 7 days with 15 minutes power supply intervals.

It is observed that as an increase in StepTolerance value results in a decrease in computation time and also an increase in relative RMSR of the optimal cost. It indicates that if I want to save computation time, I will need to sacrifice some accuracy in the optimal cost. Indeed, the actual baseline values of the optimal cost that is used to comparison have a range of about 1.44e4~2.06e4 [$]. By observing the plot, estimations are made that for saving about every 10 seconds of the computation time, an sacrifice of approximately 0.8% in relative RMSE of the optimal cost compared to the most ideal

solution, is to be made. Note that 0.8% of the ideal cost is equivalent to about 137 dollars in RMSE of the ideal solution, which has an average daily cost of 1.71e4 dollars. A good choice of step tolerance allows a reduction of computation time while does not sacrifice much accuracy. Judging from Figure 10, I may decide the step tolerance to be at 1e-4 (or 0.01%), where a save of about 66% of computation time can be achieved by sacrificing about 5.4% relative RMSE of the optimal cost compared to the ideal situation. This is equivalent to approximately 924 dollars in RMSE between the optimal costs obtained at this selected tolerance level 1e-4 (or 0.01%) and the most ideal tolerance level 1e-8 (or 1e-6%).

4.2 Impact of the Initial Condition

4.2.1 Introduction

As is described in section 3.1, regardless of types of algorithms, an initial condition is always required to start the iterative solving process. The initial condition can be treated as any feasible solutions to the optimization problem. In this microgrids optimization problem, it is equivalent to the possible State-of-Charge of the batteries, which is the battery power [KW] as a function of time [hour] in a 15 minutes (0.25 hour) increment. A good choice of initial condition can effectively reduce the computation time of the algorithm, however, it may also mislead the solver to converge to a local optimal that is not as optimal as other local optima.

Several approaches have been proposed to make a judicious guess such an initial condition. The most direct and simplest way is to visualize the objective function and get a sense of a region where the optimal could be located, and guess a point belongs to that region. However, this may not apply to objective functions that have very high dimensions and/or those that cannot be plotted on a graph for visualization. Another way is to implement some other algorithms to guess a feasible solution that may be close to local minima (or maxima). This is similar to the phase one of active-method discussed in section 3.1. A couple of methods have been studied and discussed for different situations including Machine Learning methods, sampling and clustering methods, simulated annealing and stochastic programming (i.e. BRST algorithm [20]). Note that some of these methods are quite analytical and ideal, which are not very applicable when it comes

to real world problems. Moreover, the adaptability of these methods can be highly dependent on the types and formulation of the objective function, which can be very difficult to apply to this microgrids optimization problem. In addition, implementing such an algorithm may actually add complexity to the solving process and lead to even longer computation time.

In this case of the microgrids optimization, concentrations are mostly on having a flat starting point (all zeros) with some perturbations and studying how this can affect the solving process. In spite of that, experiments with feeding in a guessed ideal solution and the solution obtained from SQP method are conducted. The guessed solution is simply the ideal solution expected by judging from the real situation. The reason why the SQP is chosen to be tested as the pre-condition algorithm, is because it seems to converge the fastest even if it lacks accuracy (see section 3.3). Its solution is then treated as a starting point and improved by using interior-point algorithm. The total time-use of the computation and accuracy of the results are tracked and compared.

4.2.2 Results and Discussion: Perturbation of Flat Initial Condition

The solution is equivalent to the State-of-Charge of the battery power [KW] as a function of time [hour]. I first start with flat initial condition and obtained the optimal battery power dispatch. Then I add white noise with incremental magnitude of the noise size [KW] that is comparable to the optimal solution, and compare the corresponding results. There are three types of noise considered: Gaussian, sine and sawtooth waves. Note that the step tolerance is chose to be 1e-4 (0.01%), which stands for the relative step change of the battery power (see section 4.1). The corresponding "ideal" solution with such step tolerance and flat initial condition are plotted and shown in Figure 11. The plots can give us a general sense of the behavior of the optimal cost and battery power dispatch. This will serve as a baseline for generating noise and comparing results obtained from subsequent perturbed initial guesses.

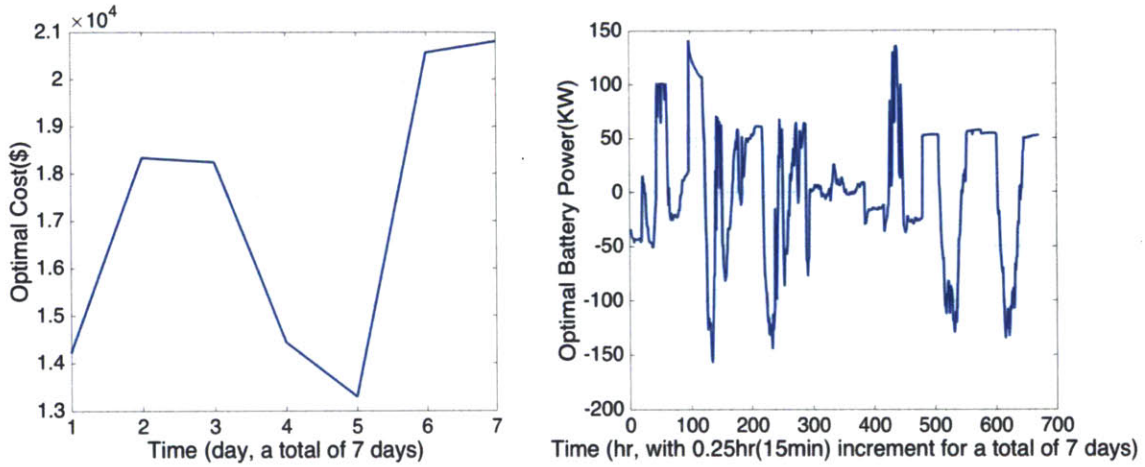


Figure 11. Plots of optimal cost (left) and corresponding battery power dispatch (right) as functions of time.

As is shown in Figure 11, the magnitude of the optimal battery power lies in a range of about [0, 200] kilowatts. Therefore, I choose to increase the noise size from 0 to 1000 kilowatts for all three types of noise. Note that noise size (in [KW]) represents the standard deviation value for Gaussian noise or the magnitude of sinewave and sawtooth noise. Each noise type with different noise size is directly added to the flat initial condition.

$$x'_{(\varepsilon)} = x + A * \varepsilon, \tag{15}$$

where

x is the original flat initial condition and here it is a vector of zeros;

ε is for the noise with unit magnitude (i.e. 1 [KW]) : for Gaussian noise, it is simply the white noise drawn from zero-mean normal distribution (~N(0,1)); for sinewave and sawtooth noise, they are generated respectively by sin(t), and sawtooth(t) in MATLAB; Figure 16 in section 4.3 shows an example of how such noise look like.

A is the noise size, a positive number, which represents the standard deviation for Gaussian noise, and the magnitude for sinewave and sawtooth noise.

The solver is run with each perturbed initial condition. In order to find the impact of noise on the optimal results, the relative root mean squared error (%) of the optimal cost

as a function of the noise size (KW) for all three types of noise is plotted and shown in Figure 12.

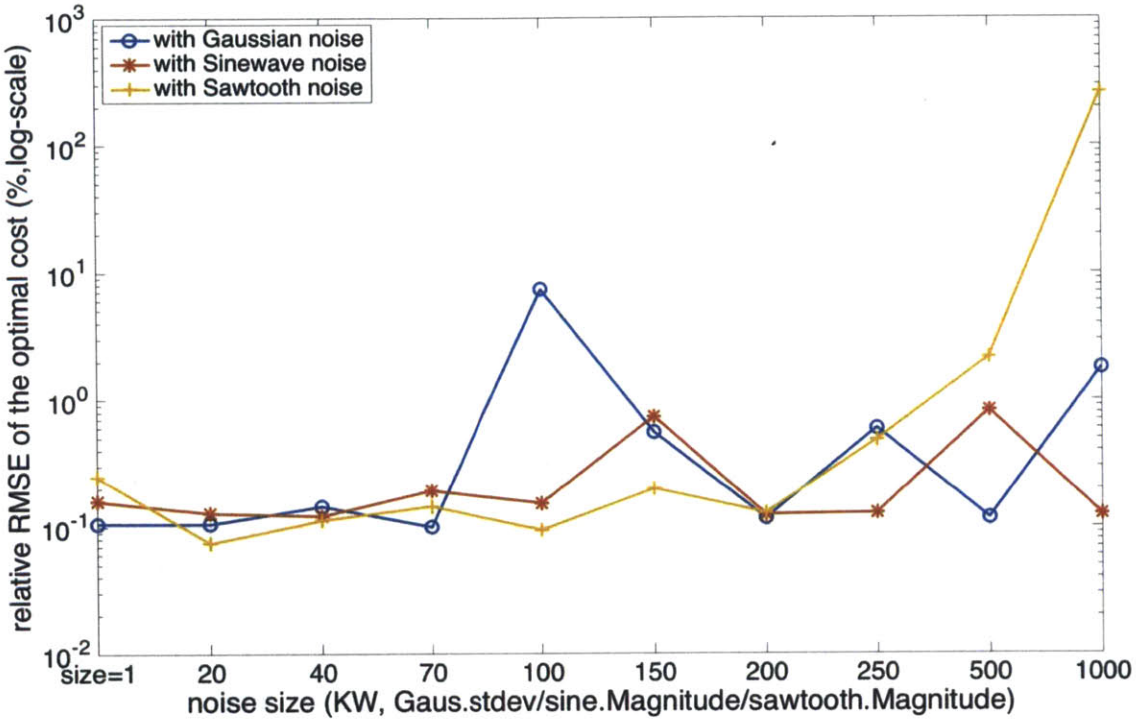


Figure 12. Plots of relative RMSE (root mean squared error) of the optimal cost as a function of noise size. The RMSE is computed against the ideal solution shown in Figure 11. Three types of noise are compared: Gaussian, sinewave and sawtooth noise.

The baseline for the relative root mean squared error is the "ideal" solution shown in Figure 11, which is the optimal cost obtained with flat initial condition. It seems that adding noise to the initial condition does not have obvious effect on the optimal cost. There may be an increasing trend when sawtooth noise was added but within the range where the "ideal solution" lies ([0,200] KW), yet not much influence on the converged cost can be observed. In addition, the computation time as a function of the noise size for three types of noise is plotted and shown in Figure 13.

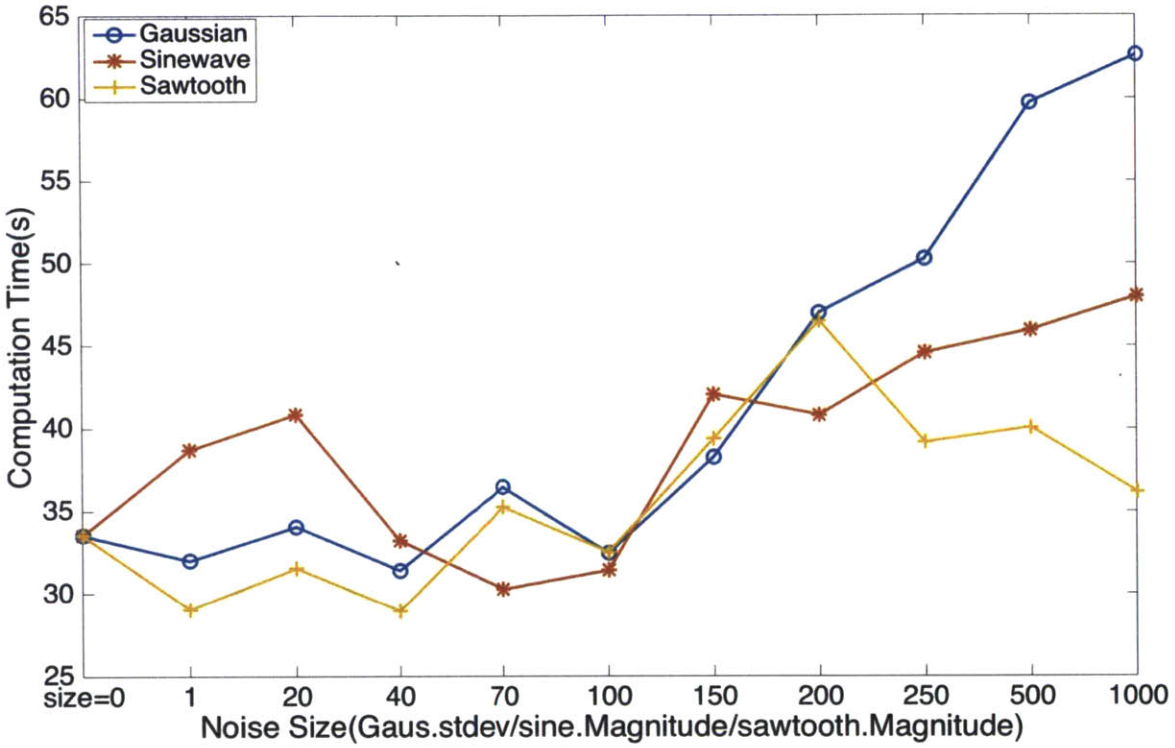


Figure 13. Plots of computation time versus the noise size added to the flat initial condition. Three types of noise are compared: Gaussian noise, sinewave noise and sawtooth noise.

It is found that as the size of noise increases, the corresponding computation time increases. Combining the observations from Figure 12 and Figure 13, it may be concluded that with flat initial condition, adding noise does not improve the solver efficiency in this microgrids optimization problem. The algorithm actually takes longer time to converge with perturbed initial condition and there seems to be no improvements on the optimal solution either.

4.2.3 Results and Discussion: The Effect of Initial Guess on Optimality and Efficiency

Since it is known that most ideal battery power dispatch should flatten out the daily load profile, the analytical ideal solution is calculated and used as the initial condition. In other words, I am using this ideal optimal solution as the starting point for the iterative process. Comparison of the optimal cost against the result obtained with flat initial condition is made. Each optimal cost and the difference of between them as

functions of time are plotted shown in Figure 14. Note that in order to obtain more generalized results, the solver is run for 30 days instead of 7 days.

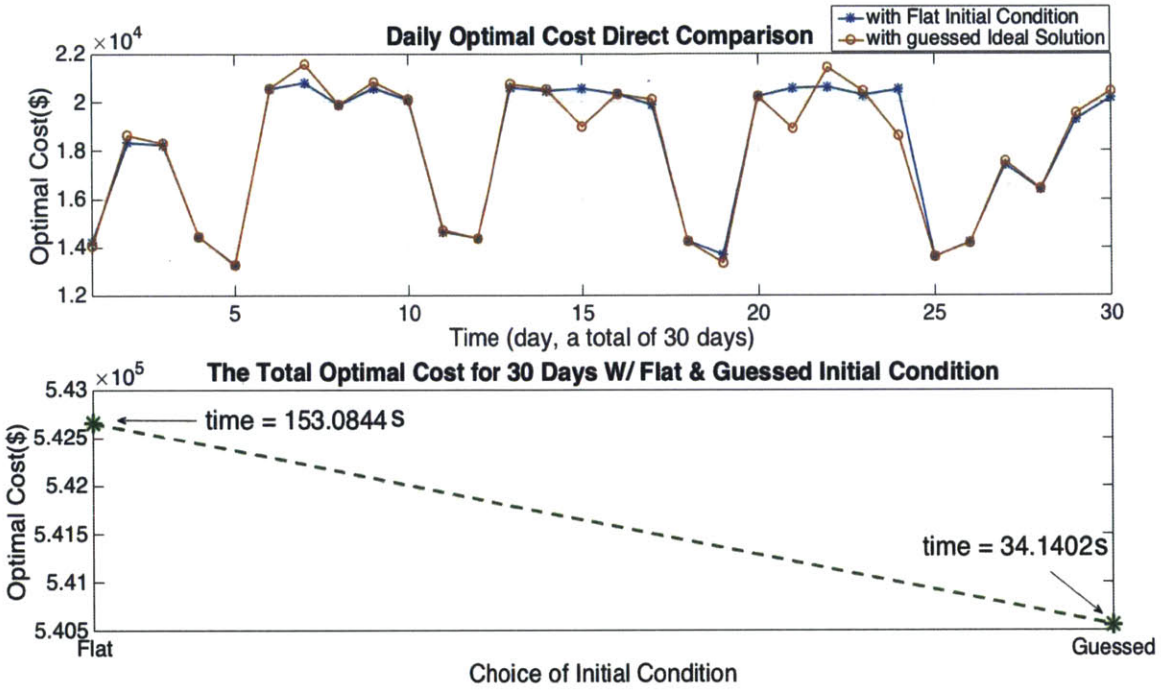


Figure 14. The plots of the optimal cost obtained by using guessed solution and flat data as initial conditions as a function time (Top); and the plot of the total optimal cost for 30 days by using guessed solution and flat data as initial condition, with corresponding computation time indicated (Bottom).

The optimal cost is improved with guessed initial condition by looking at the second plot since both the computation time and the total cost for 30 days is largely reduced after using the guessed initial condition. In particular, the 30-day total optimal cost is reduced by 0.39% and the computation time is reduced by 77.7% after the guessed solution is used as the initial condition. This result indicates that starting with an initial condition that is close to the ideal solution should save a large amount time because it will guide the solver to sink to the optimal solution in a very fast speed. For instance, the solver may only need a few iterations to converge in this case. Differently in the previous section, perturbing flat initial condition does not necessarily lead to the optimal solution, but instead, the behavior of the solver will be just similar to having flat initial condition. This is why perturbing initial condition does not save as much time compared to using

guessed initial condition. However, in real world problems, the expected ideal solution is less likely to be estimated all the time, and simply applying an ideal solution is not very applicable. On the other hand, the results indicate that the solver is minimizing the optimal cost.

4.2.4 Results and Discussion: Approximated Optimal Solutions Obtained by Using SQP Algorithm as initial guess.

According to section 3.3, SQP algorithm converges the fastest of all three algorithms but has a lack of accuracy. Therefore, it is decided to use the optimal solution obtained by using SQP algorithm as the initial condition and re-solve the problem with interior-point solver. Similar to section 4.2.3, comparisons are made with this optimal cost with the solution starting with flat initial condition, as well as the computation time. The optimal costs and the differences between two results are plotted as a function of time shown in Figure 15.

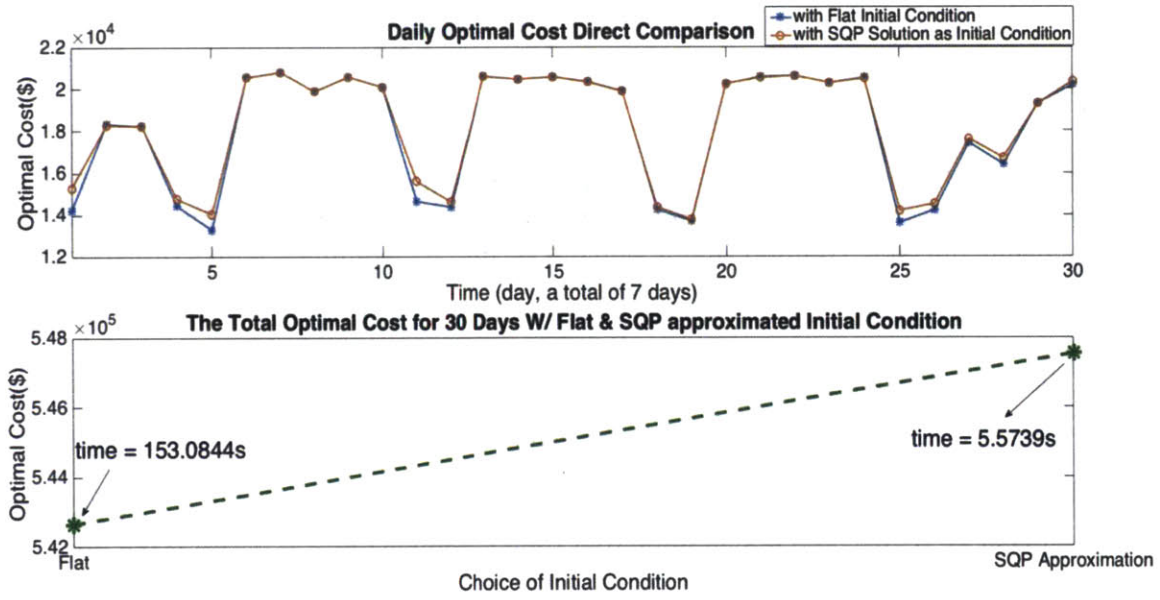


Figure 15. The plots of the daily optimal cost obtained by using SQP solution and flat data as initial conditions as a function time (Top); and the plot of the total optimal cost for 30 days by using SQP optimal solution and flat data as initial condition, with corresponding computation time indicated (Bottom).

As is shown in the bottom plot Figure 15, the optimal cost actually gets larger (by about 0.9%) after the SQP solution is used as the initial condition. It seems that using

SQP solution as the initial condition leads the solver to converge to another local minimum that is worse than what is obtained by using flat initial condition. This is contrary to the hypothesis that preconditioning the initial guess is beneficial. However, the total computation time gets improved by 77.5% (decreased from 153s to 34.4s) after I feed in the SQP solution. This result implies that using the solution from SQP algorithm may not improve the optimal cost but it can save a large amount of cpu time. In this microgrids optimization problem, the SQP algorithm seems to converge fast and does not lead to significant errors in results. However, it is not guaranteed to obtain similar results in other cases.

4.3 Smooth Objective Function by Perturbing Peak Demand Charge Data

4.3.1 The Impact of the Type and Size of the Noise

The peak demand charge (PDC) data plays an important role in estimating the total cost. This data represents the power charge, in $/kw-month, for consumers' energy demand at peak hours. In PJM region, about 26% of the electricity bill is comprised of demand charge. In the numerical studies presented above, the peak demand charge was set to be constant (24 $/kw-month) through out the whole year. Mathematically, it gives the objective function a flat slope, which results in some discontinuity in the objective function. In addition, it is not very likely to have flat peak demand charge (PDC) for the entire year in most electricity markets. Therefore, I decided to perturb this data and compare the corresponding results. Similar to the perturbation of initial condition, three types of noise are added to the original flat peak demand charge data. An example of such noise is shown in Figure 16.

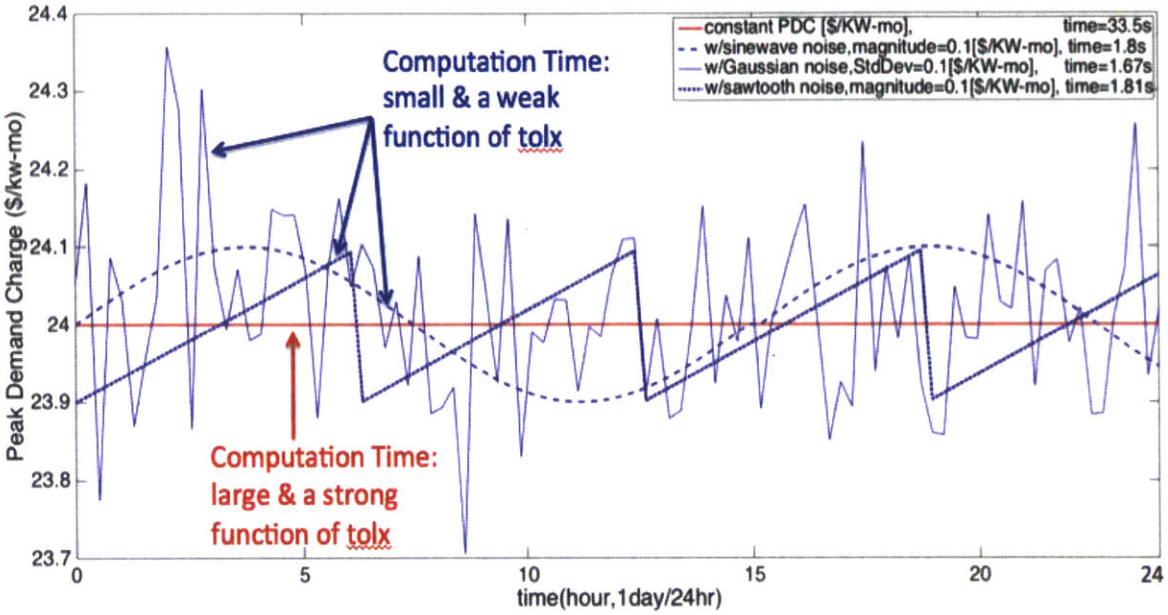


Figure 16. Sample plots of three types of noise as a function of time in one day. The increment for the time period is 0.25 hour (15 mins). Tolx stated in the graph stands for the step tolerance (see section 4.1).

In Figure 16, the effect of step tolerance (tolx) is compared against the variations of peak demand charge. It is found that when peak demand charge is a constant vector, the computation time tends to be very large and it acts as a strong function of the step tolerance. This case is discussed in section 4.1.2. When the peak demand charge is not constant, the computation time is largely reduced and becomes a very weak function of the step tolerance. This observation can be explained mathematically as well, as mentioned before, constant peak demand charge adds discontinuity to the objective function, which makes the estimation of its gradient to be very difficult. However, when the noise is added to the peak demand charge, even if it is very small, "some gradients is provided" to the objective function. This result, that a variable peak demand charge improves convergence, is useful as it suggests further strategies for numerical efficiency. Due to the very high sensitivity of the solver, the optimal cost may change compared to the optimal cost with flat peak demand charge. Figure 17 shows the plots of the optimal costs obtained after adding the sample noise (shown in Figure 16) to the peak demand charge.

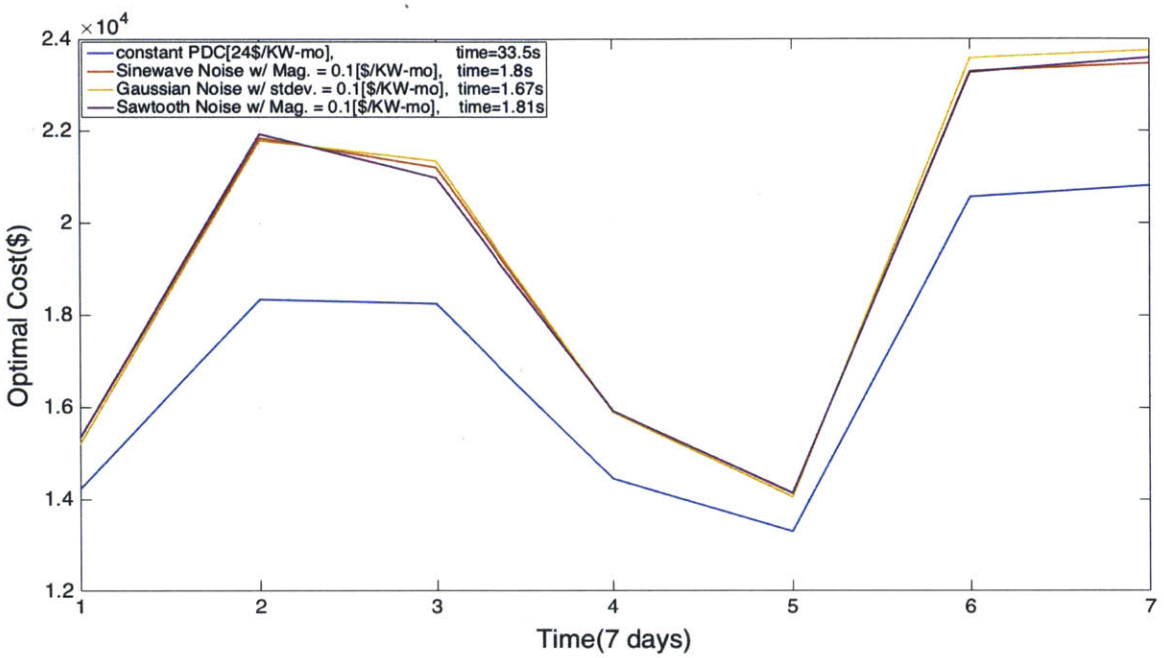


Figure 17. Plots of the optimal cost as a function of time, corresponding to the sample noise shown in Figure 15.

As is shown in Figure 17, while I achieve a large reduction of the computation time (about 94% compared to the case without noise), the corresponding optimal cost becomes larger after the noise is added to the peak demand charge. The corresponding relative RMSE (root mean squared error) is around 14% compared to the cost obtained with flat peak demand charge, which is about 2400 dollars in RMSE. This does not really suggest that adding noise will result in worse optimal cost, since the objective function is slightly changed. The solver is minimizing the cost based on a new objective function. The costs obtained with perturbed peak demand charge seem to be quite similar regardless of the noise type, which suggests that the solver is doing its job. It is to discover the relationship between the noise size and the computation time and accuracy of the cost.

4.3.2 Trade-offs between Computation Time and Convergence Accuracy

The goal is to find the threshold of the noise size that will reduce the computation time while maintain an acceptable cost accuracy. The step tolerance is still kept at 0.01% as discussed in section 4.1.2. At this tolerance level, the noise size is gradually increased and the corresponding computation time is tracked. A plot of computation time as a

function of noise size in percent of the original flat demand charge (24$/kw-month) is shown in Figure 18. All three types of noise are compared.

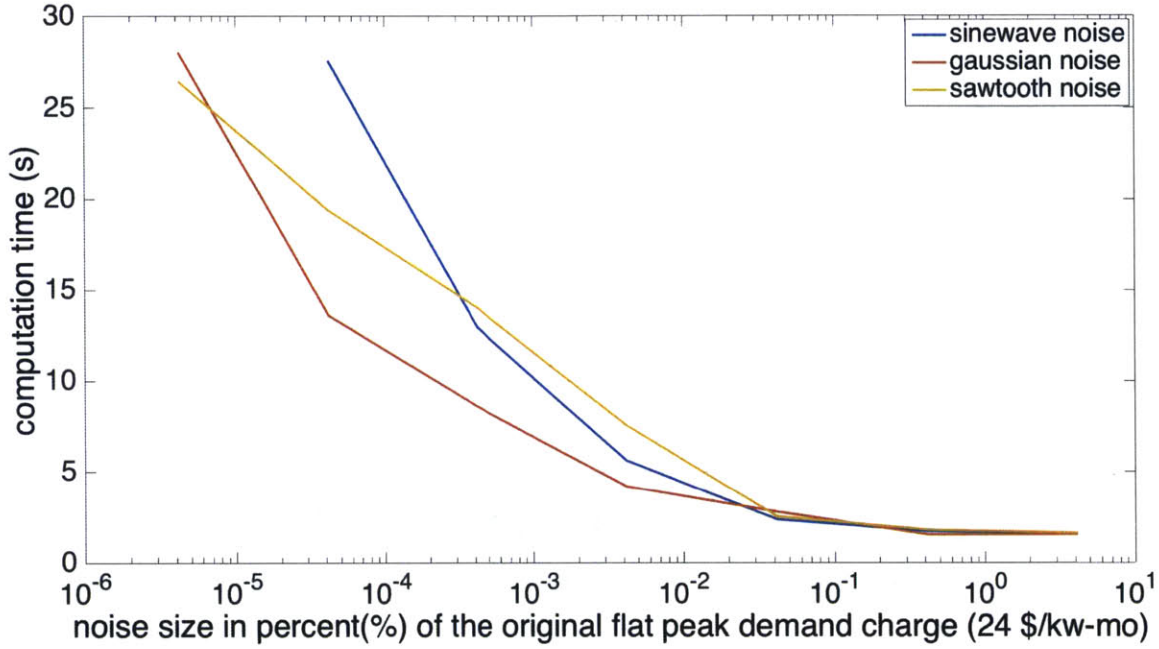


Figure 18. Plots of computation time(s) as a function of noise size in percent of the original flat peak demand charge level (24$/kw-mo), at tolx = 1e-4 (unit).

It is observed that the computation time decreases as the noise size increases. There exists a certain range of tolerance size that can influence computation time, which is between about 1e-5% and 1% (in $/Kw-mo) of the flat peak demand charge (24$/kw-month). In other words, the computation time will not be decreased when noise size is beyond of this range. This sensitive range of tolerance range can be applied to the solver and eventually find out a threshold of the noise size. Within this threshold, it is expected that the computation time should be reduced without affecting the accuracy cost much. Therefore, the noise size is gradually increased in that sensitive range shown in Figure 18, and the corresponding optimal cost is tracked. A plot of the relative RMSE (root mean squared error) of the optimal cost versus the noise size (in percent) is shown in Figure 19. Note that the percentage of the noise size is computed against the flat peak demand charge value, 24 $/kw-month; and the relative RMSE is obtained by comparing the optimal cost obtained by using the flat peak demand charge. Since all three types of

noise yield similar computation time at each noise size level, the average time of three of them is calculated and show then at the corresponding spot in the graph.

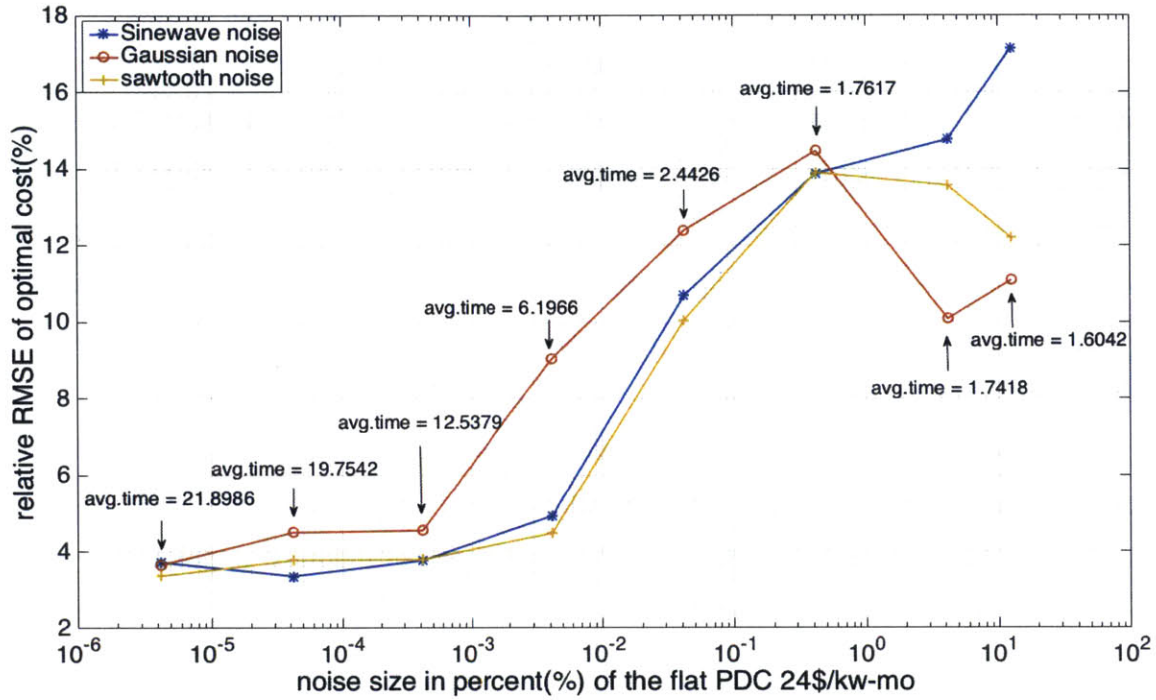


Figure 19. Plots of relative RMSE of the optimal cost as a function of noise size in percent of the flat peak demand charge (PDC), 24 $/kw-mo. The baseline for the relative optimal cost is the optimal solution obtained with flat PDC. The corresponding average computation time of three noise at each noise size is shown as well.

It is observed from Figure 18 that as noise size increases, the relative RMSE of the optimal cost tend to increase. The threshold for the magnitude of the noise is found to be around (4.17e-4)% of the constant peak demand charge level (24 $/kw-mo), which is equivalently 1e-4 $/kw-hour. At this threshold, the optimal time is largely reduced by about 61% (decreased from 33s to 13s) compared to the time used when constant peak demand charge is applied. This is to sacrifice about 4% in relative RMS of the optimal cost ($), which is approximately 685 dollars in RMS of the optimal cost compared to the average daily optimal cost (1.7e4 dollars) obtained with constant PDC. Depending on personal requirements of accuracy and computation time, one can always choose different noise level with different combination of computation time and accuracy.

Comparing this results to that of perturbing flat initial condition in section 4.2.2, it is essential to notice the difference between the perturbation of flat peak demand charge

and flat initial condition. Previously perturbing flat initial condition does not seem to improve the optimization results, whereas perturbing peak demand charge can save computation time without sacrificing much accuracy. This is because that adding noise to the flat initial condition only changes the starting point of the iterative solving process, which does not affect the formulation of the optimization problem. Only if the perturbed initial condition happens to be close to an optimal solution, may the solving process take less time to converge (see section 4.2.3 and section 4.2.4). However, since peak demand charge is one of the two major components of the calculation of total cost, also noted as the objective function, perturbing the flat peak demand charge is essentially bringing gradients to the objective function. With such smoothed objective function, the solving process will take much less time to converge compared to having the original objection function with flat peak demand charge, equivalently large discontinuity. Note that estimation of the gradient of the objective function is one of the most important steps in solving for an optimum (see section 3).

5 Conclusions and Future Work

This microgrid dispatch optimization problem is found to have very large dimensions, and the objective function is highly nonlinear and non-smooth. The process of optimizing this objective function is quite complex and computational intensive. Among all four optimization algorithm (trust-region reflective, active-set, interior-point and SQP methods) provided by MATLAB, interior-point method is chosen to be the best fit for solving this optimization problem not only because it is designed to solve large-scale problems, but also it can provide relatively good convergence speed and accuracy among all three algorithms.

Different tolerances are studied and discussed. The StepTolerance (tolx) and the FunctionTolerance (tolfun) are found to be highly related to the convergence time and accuracy. Both the mathematical and physical definitions of the tolerances are defined for this specific optimization problem; and an equation of tolx as a function of tolfun is extracted based on the nature of this problem. A suitable value of StepTolerance is

determined to reduce computation time by about 66% whilst sacrificing only about 5.4% relative RMSE of the optimal cost, compared to the ideal case.

After I choose the StepTolerance level, I am to find the impact of choices of initial condition on the solving this optimization problem. The initial condition was set to be zeros originally. Firstly, the flat initial condition data is perturbed with three types of noises: Gaussian noise, sinewave noise, and sawtooth noise. Different sizes of the noise are added and the corresponding results are compared to that obtained with flat initial condition. It is observed that perturbing initial condition does not necessarily improve the optimal solution but results in an increase the computation time. Secondly, I feed in the initial condition with an ideal guess, which is estimated by the understanding of this optimization problem. The results show a great reduction on computation time and the optimal cost is slightly improved, compared to the time and cost obtained with flat initial condition. However, this is not quite applicable in real world problems, since I do not always know the expected ideal solution. Nevertheless, the results indicate that my solver is actually doing well on minimizing the optimal cost. Last but not the least, an optimal SQP solution is fed in as the initial condition to the interior-point solver. Compared to the results by using flat initial condition, the optimization time is reduced in this case, but the optimal solution becomes slightly worse. This result implies that using the solution from SQP algorithm may not improve the optimal cost but can effectively reduce computation time. Using SQP algorithm as a pre-condition method maybe a good choice for my simplified case, but it is not guaranteed once I add complexity to the problem to simulate the situations in real world.

Perturbing the peak demand charge data is found to be a good way to smooth the objective function and it is very effective in terms of reducing computation time. This is mainly because that by adding noises to the constant peak demand charge data, I am providing some gradients to the objective function, which as a result, can largely facilitate the solving process. Similar to the perturbation of the initial condition, the same types of noises are added to the peak demand charge data. It is observed that increasing noise size in a certain range can largely reduce the computation time. However, by doing this, the accuracy of the optimal results may get worse. A threshold of a noise size is determined by testing the solver with different noise sizes, below which I can save a

maximum of approximated 61% of the computation time with a sacrifice of no more than 4% relative RMS of the optimal cost compared to the solution with flat peak demand charge. In addition, this result indicates that for electricity market with relatively flat peak demand charge, adding certain noises can be effective in improving the efficiency of solving for the optimal microgrid assets dispatch, whereas for electricity markets that has obvious variations in peak demand charge (i.e. in California), the model might already work quite fast and there is no need to perform such perturbation anymore.

For future research, it will be interesting to discover some advanced pre-condition methods such as Machine Learning methods, and sampling and clustering methods. It is also worth working on searching for other approaches of smoothing objective function including curve fitting and convolution methods. More complexity can be added to the optimization problem in order to simulate real world problems, and more insightful information can be obtained by applying such model in different electricity markets.

Bibliography

[1] Peng, Peng. "Microgrids: A Bright Future." *Huawei Communicate* 63 (2011): 6-8. *HUAWEI.* HUAWEI, 2011. Web. 20 Apr. 2016.

[2] "Power Restoration." *Power Restoration.* ONCOR, n.d. Web. 20 April 2016.

[3] Crawford, Mark. "Microgrids May Power Entire U.S. in Future." *Asml.org.* ASML, Jan. 2013. Web. 20 Apr. 2016.

[4] "What Are the Benefits of the Smart Microgrid Approach?" *Galvin Electricity Initiative.* The Galvin Project, Inc, n.d. Web. 20 Apr. 2016.

[5] Black, Jon. "Update on Solar PV and Other DG in New England." *Update on Solar PV and Other DG in New England* (n.d.): n. pag. *ISO New England Planning Advisory Committee.* ISO New England, 19 June 2013. Web. 23 Apr. 2016.

[6] "Constrained Nonlinear Optimization Algorithms." *Mathworks.com.* MathWorks, n.d. Web. 23 Apr. 2016.

[7] Bertsekas, Dimitri P. *Nonlinear Programming.* Belmont, MA: Athena Scientific, 1999. Print.

[8] Wong, Elizabeth. "Active-Set Methods for Quadratic Programming." *A Dissertation Submitted in Partial Satisfaction of the Requirements for the Degree Doctor of Philosophy in Mathematics* (2011): n. pag.*Ccom.ucsd.edu.* University of California, San Diego. Web. 24 Apr. 2016.

[9] "Active Set Method." *Wikipedia.* Wikimedia Foundation, n.d. Web. 25 Apr. 2016.

[10] "Sequential Quadratic Programming." *Neos-guide.org.* NEOS Guide, n.d. Web. 25 Apr. 2016.

[11] Boggs, Paul T., and Jon W. Tolle. "Sequential Quadratic Programming." *Acta Numerica* (1996): 1-48. *People.duke.edu.* Web. 25 Apr. 2016.

[12] Hauser, Kris. "B553 Lecture 10: Convex Optimization and Interior-Point Methods." *Convex Optimization* (2012): 1-4. *Homes.soic.indiana.edu.* Web. 25 Apr. 2016.

[13] "Fmincon Interior Point Algorithm; Fmincon Trust Region Reflective Algorithm." *Constrained Nonlinear Optimization Algorithms.* MathWorks, n.d. Web. 25 Apr. 2016.

[14] Gill, Phillip E., and Elizabeth Wond Wong. "Sequential Quadratic Programming Methods." *UCSD Department of Methematics Technical Report* (2010): 1-63. *Ccom.ucsd.edu*. University of California, San Diego. Web. 25 Apr. 2016.

[15] "Fmincon Description." *Find Minimum of Constrained Nonlinear Multivariable Function*. MathWorks, n.d. Web. 26 Apr. 2016.

[16] "Output Arguments." *Find Minimum of Constrained Nonlinear Multivariable Function*. MathWorks, n.d. Web. 26 Apr. 2016.

[17] "When the Solver Might Have Succeeded." *Optimization Toolbox*. MathWorks, n.d. Web. 26 Apr. 2016.

[18] "Tolerances and Stopping Criteria." *Optimization Toolbox*. MathWorks, n.d. Web. 26 Apr. 2016.

[19] "First-Order Optimality Measure." *Optimization Toolbox*. MathWorks, n.d. Web. 26 Apr. 2016.

[20] Boender, C.G.E., A.H.G. Rinnooy Kan, L. Strougie and G.T. Timmer (1982). "A stochastic method for global optimization". *Mathematical Programming* 22: 125–140. 27 Apr. 2016.