# Support Vector Machine with Line Search SQP using a flexible step acceptance strategy

Meijia Liu

April 6, 2022

# Contents

# 1 Introduction

The Support Vector Machine(SVM) is one of the machine learning techniques that can help to solve classification problems. It s a linear classifier that can be viewed as an extension of the perceptron, which guaranteed to find a hyperplane if it exists. However, if there is no separating hyperplane between two classes because of low dimension, the linear boundary is not feasible. As a result, we allow some misclassifications and use soft margin for classifier. This report will solve linear non-separable case and use Line Search Subsequence Quadratic Programming algorithm to find optimal hyperplane.

# 2 Support Vector Machine

## 2.1 Linear Separable SVM

The classification problem can be represented as follows. Let $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \in \boldsymbol{R}^n$ be a training set, in binary classification the set of label is $\{-1, 1\}$. A hyperplane means a set $H_{\boldsymbol{w},b} = \{\boldsymbol{x} \in \boldsymbol{R}^n : \boldsymbol{w}^T\boldsymbol{x} + b = 0\}$, where $\boldsymbol{w} \in \boldsymbol{R}^n$ and $b \in \boldsymbol{R}$. The value of each $w_i$ is a measure of the importance of each attribute for the classification problem and $b$ is a bias value.

The objective is to maximize the distance between hyperplane and the closest point within both classes, which also named 'margin', under the constraints that all data points lie on the correct side of the hyperplane. It solves optimization problem:

$$H(S) := \max_{\boldsymbol{w},b} \min_{1\le i\le m} \{\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{||\boldsymbol{w}||} : y_j(\boldsymbol{w}^T\boldsymbol{x}_j + b \ge 0), j = 1, \cdots, m\} > 0 \tag{1}$$

where $\frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}{||\boldsymbol{w}||}$ is the distance of a point $x$ from the hyperplane $H_{\boldsymbol{w},b}$. In this report we choose to work with Canonical hyperplane. Canonical hyperplane require that $min_{i=1}^m y_i(\boldsymbol{w}^T\boldsymbol{x}_i) = 1$ [5], we have:

$$H(S) = \max_{w,b}\{\frac{1}{||\boldsymbol{w}||} : \min_i\{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\} = 1, y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \ge 0\} \tag{2}$$

$$= \max_{w,b}\{\frac{1}{||\boldsymbol{w}||} : y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \ge 1\} \tag{3}$$

Then the optimal hyperplane can be constructed by solving the following constrained quadratic optimization primal problem:

$$Minimize \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{4}$$

$$subject\ to \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \ge 1, i = 1, \cdots, m \tag{5}$$

## 2.2 Linear non-separable SVM

If the data cannot be separated by a hyperplane perfectly, we can fix this by allowing some misclassification with the introduction of slack variables [3]:

$$Minimize \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i \tag{6}$$

$$subject\ to \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi, \quad \xi_i \geq 0 \tag{7}$$

$C$ is complexity constants and $\xi_i$ denotes the misclassification loss, here we replace $\xi_i$ by the hinge loss:

$$\xi_i = \begin{cases} 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 1 \\ 0 & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \end{cases} \tag{8}$$

Substitute (8) into equation (6), we obtain the following unconstrained version as loss function and regularizer:

$$\min_{w,b} \quad \boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n} max[1 - y_i(\boldsymbol{w}^T\boldsymbol{x} + b), 0] \tag{9}$$

### 2.2.1 New dual form

The solution of equation (6) combined with inequation (7) is equivalent to find the saddle point of the Lagrangian function

$$L(\boldsymbol{w}, \boldsymbol{\xi}, b; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i[y + i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \xi_i - 1] - \sum_{i=1}^{m}\beta_i\xi_i \tag{10}$$

where $\alpha_i,\ \beta_i \geq 0$ are the Lagrange multipliers. We minimize $L$ over $\boldsymbol{w},\ \boldsymbol{b}\ and\ \boldsymbol{\xi}$ and obtain:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} y_i\alpha_i = 0 \tag{11}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i = \mathbf{0} \tag{12}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{13}$$

Then dual problem shows [2]:

$$\min_{\alpha_1,\cdots,\alpha_n} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j < x_i, x_j > -\sum_{i=1}\alpha_i \tag{14}$$

$$s.t. \quad 0 \leq \alpha_i \leq C \tag{15}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{16}$$

### 2.2.2 Kernel Tricks

For a non-linear classification problem in the input space, it can be transformed into a linear classification problem in some dimensional feature space by means of a non-linear transformation to train a linear support vector machine in a high-dimensional feature space. Since both the objective function and the classification decision function involve only the inner product between instances in a pairwise problem learned by a linear support vector machine, it is not necessary to explicitly specify the non-linear transformation, but to replace the inner product with a kernel function.

According to Somla et al.(1998) [6], under general smoothness assumption, gaussian kernel tend to perform well and should be considered if no additional information could obtained from data. Gaussian kernel, also known as RBF kernel:

$$K(x,z) = exp(\frac{-||x-z||^2}{2\sigma^2}) \tag{17}$$

As a result, our new dual problem under kernel trick is:

$$\min_{\alpha_1,\cdots,\alpha_n} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} - \sum_{i=1}\alpha_i \tag{18}$$

$$s.t. \quad 0 \leq \alpha_i \leq C \tag{19}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{20}$$

where $\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \phi(x_i)$ and prediction:

$$h(x) = sign(\sum_{i=1}^{n}\alpha_i y_i k(\boldsymbol{x}_i, x) + b) \tag{21}$$

## 2.3 Support Vectors

Support vectors are those points satisfies:

$$y_i(\boldsymbol{w}^T\phi(xi) + b) = 1 \tag{22}$$

and the complementary KKT-conditions says $\alpha(y_i(\boldsymbol{w}^T\phi(xi)+b)-1)=0$ and $\alpha \geq 0$. Equation (22) shows that $y_i(\boldsymbol{w}^T\phi(xi)+b)-1 \neq 0$ for training inputs without support vectors, then to satisfy KKT conditions, $\alpha_i = 0$ has to be satisfied. As a result, $\alpha_i > 0$ for support vectors. This lead to a nice interpretation of the dual problem, where $h(x)$ is equivalent to the sum of all support vectors and all inputs $\boldsymbol{x}_i$ with nonzero alpha can be discarded after training.

### 2.4 Recovering b

The bias term is indeed, a special term in SVM which controls whether hyperplane pass through the original point. The bias term should be treated separately because bias term is no longer part of our dual optimization problem. As we mentioned before, in the dual, support vectors are those with $\alpha_i > 0$. we can calculate $b$ from support vectors:

$$b = y_i - \sum_j y_j \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) \tag{23}$$

## 3 Line Search SQP

Recall last section, we need to optimize:

$$\min_{\alpha_1,\cdots,\alpha_n} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} - \sum_{i=1}\alpha_i \tag{24}$$

$$s.t. \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{25}$$

$$\forall i \ \alpha_i \geq 0 \tag{26}$$

$$\forall i \ C - \alpha_i \geq 0 \tag{27}$$

It is a constraint optimization problem with both equality and inequality linear constrains. The objective function is a quadratic programming and the problem is deterministic. The challenging for this problem is it has inequality constrains and more than one.

In this report we will use Line Search SQP method given by Wright and Nocedal(1999) [7] and modified it with a flexible step acceptance strategy raised by Zhu and Pu(2012) [8].

## 3.1 Equality Constraints

This subsection start from equality-constrained problem:

$$min \quad f(\boldsymbol{x}) \tag{28}$$

$$s.t \quad c(\boldsymbol{x}) = \mathbf{0} \tag{29}$$

Define $A(\boldsymbol{x}) = \bigtriangledown c(\boldsymbol{x}) = [\bigtriangledown c_1, \cdots, \bigtriangledown c_m] \in R^{n \times m}$, is a Jacobian matrix of $c(\boldsymbol{x})$

**SQP framework:**

At step $x_k$, assume $f(x_k)$ is not optimal, hence we expect that the value $f(x_k + p)$ should decrease with respect to the descent direction in k-th step $p$. Additionally, $x_k + p$ should still in feasible set $D$, then we get a new subproblem which minimize $f(x_k + p)$:

$$\min_p \quad f(x_k + p) \tag{30}$$

$$s.t. \quad c(x_k + p) = 0 \tag{31}$$

Through second-order and first-order Taylor expansion for (30) and (31), our subproblem is almost equivalent to:

$$\min_p \quad f_k + \bigtriangledown f_k^T p + \frac{1}{2} p^T \bigtriangledown^2 f(x) p \tag{32}$$

$$s.t. \quad c_k + \bigtriangledown c_k^T p = 0 \tag{33}$$

The Lagrangian for this problem is:

$$L(p, \lambda) = f_k + \bigtriangledown f_k^T p + \frac{1}{2} p^T \bigtriangledown^2 f(x) p - \lambda^T (c_k + \bigtriangledown c_k^T p) \tag{34}$$

where $\mu \in R^m$ is the Lagrangian multiplier for equality constraint. The KKT conditions for this problem are:

$$\begin{cases} \bigtriangledown_p L(p, \lambda) = 0 \\ c_k + \bigtriangledown c_k^T p = 0 \end{cases} \tag{35}$$

which is equivalent to:

$$F(p, \lambda) = \begin{bmatrix} \bigtriangledown_p L(p, \lambda) \\ c_k + \bigtriangledown c_k^T p \end{bmatrix} = \mathbf{0} \tag{36}$$

We then apply Newton-Raphson Iteration Method $\bigtriangledown^T F(x_k)p = -F(x_k)$ to calculate the zero points of (36). Then we have:

$$\begin{bmatrix} \bigtriangledown_{pp} L(p,\lambda) - \bigtriangledown c_k \\ \bigtriangledown c_k \end{bmatrix} \begin{bmatrix} p_{xk} \\ p_{\lambda k} \end{bmatrix} = \begin{bmatrix} - \bigtriangledown_p L(p,\lambda) \\ c_k + \bigtriangledown c_k^T p \end{bmatrix} \tag{37}$$

at one particular iteration step $p_k$

This is a sufficient and necessary condition for the following quadratic optimal problem:

$$\min_p \quad g_k^T p + \frac{1}{2} p^T H_k p \tag{38}$$

$$s.t. \quad A_k^T p + c_k = 0 \tag{39}$$

where $A_k = \bigtriangledown c_k$, $g_k$ is gradient Lagrangian and $H_k$ is a symmetric approximation to the current Lagrangian Hessian. Apply the subsequence quadratic optimal problem consist of objective function (36) and constraints (37) in each iteration to the new iterate $(x_{k+1},\ \lambda_{k+1})$ is the basic and simplest form of SQP method.

## 3.2 Inequality Constraints

According to our final SVM soft margin dual optimization problem (24)-(27), we can write the constraint problem in the following general form:

$$\min_\alpha \quad f(\alpha) \tag{40}$$

$$s.t. \quad c_\epsilon(\alpha) = 0 \tag{41}$$

$$c_I(\alpha) \geq 0 \tag{42}$$

where $\alpha \in R^n$, $\epsilon = \{i | i = 1, 2, \cdots, m_e\}$ and $I = \{i | i = m_e + 1, \cdots, m\}$, $f : R^n \to R$ and $c : R^n \to R^m$ are smooth functions. The Lagrangian for this constraint problem is:

$$L(x, \lambda, \mu) = f(\alpha) - \lambda^T c_\epsilon(\alpha) - \mu^T c_I(\alpha) \tag{43}$$

and its KKT conditions are:

$$\begin{cases} \bigtriangledown f(\alpha) - A_\epsilon^T \lambda - A_I^T \mu = 0 \\ c_\epsilon(\alpha) = 0 \\ c_I(\alpha) \geq 0 \\ \mu \geq 0 \\ \mu^T c_I(x) = 0 \end{cases} \tag{44}$$

Extended from (38) and (39) in equality constraints, the new iterate $(x_{k+1}, \lambda_{k+1})$ can therefore be defined as he solution of the quadratic program:

$$\min_p \quad g_k^T p + \frac{1}{2} p^T H_k p \tag{45}$$

$$s.t. \quad A_{\epsilon k}^T p + c_{\epsilon k} = 0 \tag{46}$$

$$A_{Ik}^T p + c_{Ik} \geq 0 \tag{47}$$

According to Wright and Nocedal(1999), this ensures local convergence since we can always convergent to optimal solution if the calculated $p_k$ is a descent direction. However, if $p_k$ is not a acceptable descent direction, our basic algorithm will not convergent to the optimal solution.

## 3.3 Algorithm Development

To avoid insufficient decrease descent $p_k$ in each sub-quadratic program, this report use Damped BFGS Updating [7] to approximate and ensure positive definite Hessian and merit function step acceptance strategy with feasible penalty functions [8] to generate step length $\alpha_k$

### 3.3.1 Merit Function

In most common SQP methods, the next iterate step length is obtained by a line search backtracking along the search direction, here we use a merit function to decide whether a trial step should be accepted. A widely used merit function is the $l_1$ penalty function [8]:

$$\phi(x; \pi_k) := f(x) + \pi_k h(x) \tag{48}$$

where $\pi_k$ is the current penalty parameter. Extended to inequality case [9], the flexible penalty function is:

$$\phi(x; \pi_k) = f(x) + \pi_k ||c_\epsilon(x)||_1 + \pi_k ||c_I(x)^-||_1 \tag{49}$$

where $c_I(x)^- = min\{c_I(x), 0\}$. Then the directional derivative of $\phi$ in the direction $p_k$ satisfies:

$$D(\phi(x;\pi_k);p_k) = \nabla f(x) + \pi_k||\nabla c_\epsilon(x)||_1 + \pi_k||max\{-\nabla c_I(x), 0\}||_1 \tag{50}$$

In order to ensure that $p_k$ is a descent direction, we choose $\pi_k$ such that:

$$\pi_k = \begin{cases} \pi_{k-1} & \pi_{k-1} \geq r_k \\ r_k + \epsilon & otherwise, \end{cases} \tag{51}$$

where $r_k = max\{||\lambda_k||_\infty, ||\mu_k||_\infty\}$, $\epsilon > 0$ is a small constant.
According to Zhu and Pu(2012) [8], if we choose penalty parameter too low, we may produce an infeasible point while slow convergence when it is large.
According to Boggs and Tolle(1995) [?], it must be possible to reduce $\phi$ by taking an appropriate step in the direction $p_k$ generated by solving the quadratic subproblem. This appropriate step length that decreases merit function can be computed by a backtracking procedure of choosing smaller step length until a suitable one is obtained. In a line search method, a step $\alpha_k p_k$ will be accepted if:

$$\phi(x_k + \tau^j p_k;\pi_k) \leq \phi(x_k;\pi_k) + \eta\tau^j D(\phi(x;\pi_k);p_k) \tag{52}$$

where j is the j-th iteration when doing line search. After finding an optimal next iterate step length, we set $\alpha_k = \tau^{j_k}$ and update $x_{k+1} = x_k + \alpha_k p_k$.

### 3.3.2 Damped BFGS Updating

The Hessian of the Lagrangian is made up of the second derivatives of the objective function and constraints, in some situation it is hard to get enough information to compute. Besides, normal BFGS method could not ensure a positive definite approximated hessian at each step [7]. That is, in BFGS method, we update $H_k$ with vectors $s_k$ and $y_k$:

$$s_k = x_{k+1} - x_k \tag{53}$$

$$y_k = \nabla_x L(x_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \lambda_{k+1}) \tag{54}$$

may not satisfy the curvature condition $s_k^T y_k > 0$. Damped BFGS Updating ensures that the updated Hessian always positive definite through modifying the definition of $y_k$.
Define:

$$r_k = \theta_k y_k + (1 - \theta_k) H_k s_k \tag{55}$$

where $\theta_k$ is defined as:

$$\theta_k = \begin{cases} 1 & s_k^T y_k \geq 0.2 s_k^T H_k s_k \\ (0.8 s_k^T H_k s_k)/(s_k^T H_k s_k - s_k^T y_k) & s_k^T y_k < 0.2 s_k^T H_k s_k \end{cases} \tag{56}$$

According to Wright and Nocedal(1999) [7], the choice of $\theta_k$ makes new approximation to stay close to the current approximation thus positive definiteness could be guaranteed.
Update $H_k$ as follows:

$$H_{k+1} = H_k - \frac{H_k s_k s_K^T H_k}{s_k^T H_k s_k} + \frac{r_k r_k^T}{s_k^T r_k} \tag{57}$$

The formula (57) ensures positive definite of $H_{k+1}$ through replacing $y_k$ with $r_k$

# 4 Numerical Results

In this report we use generated spiral data as classification problem:

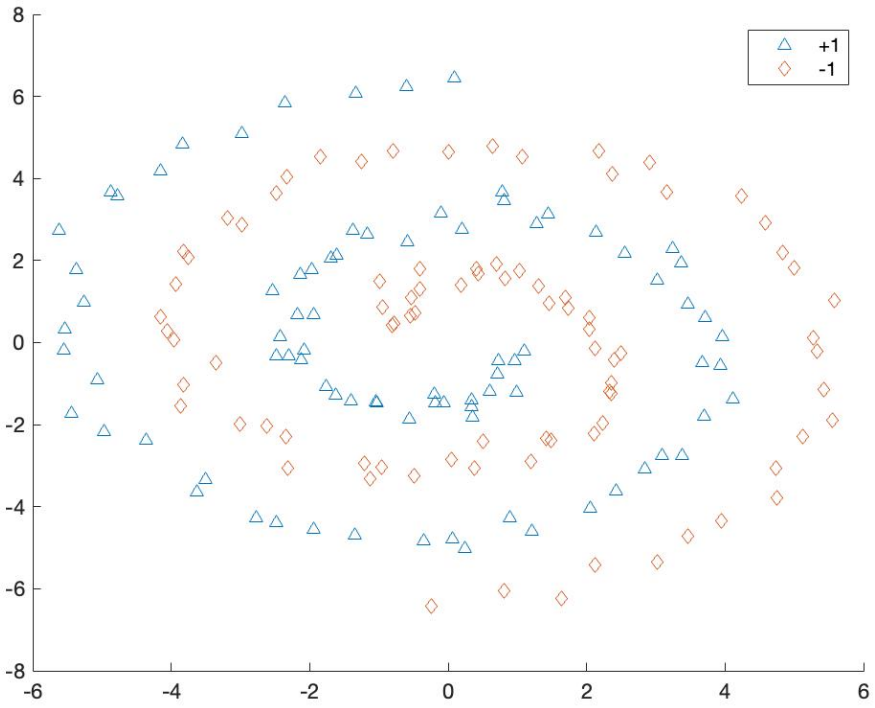



Fig1: Two spiral data sets (training)

As we mentioned in Section 2, our optimization problem is:

$$\min_{\alpha_1,\cdots,\alpha_n} \quad \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} - \sum_{i=1}\alpha_i \tag{58}$$

$$s.t. \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{59}$$

$$\forall i \ \alpha_i \geq 0 \tag{60}$$

$$\forall i \ C - \alpha_i \geq 0 \tag{61}$$

In this section, some numerical results will be reported to verify the efficiency of our algorithm. We follow the terminate idea suggested by Wang and Pu(2013) [9] if:

$$||\nabla f_k - \nabla c_\epsilon(x_k)^T\lambda_k - \nabla c_I(x_k)^T\mu_k||_2 \leq 10^{-6} \tag{62}$$

$$||(c_\epsilon(x_k), max\{-c_I(x_k), 0\})||_2 \leq 10^{-6} \tag{63}$$

The parameters are chosen as: $\tau = 0.5$, $\eta = 0.1$, $\pi_0 = 1$, $\epsilon = 10^{-4}$ and we initialize $\boldsymbol{\alpha}_0 = (0, 0, \cdots, 0)^T$, $\lambda_0 = 0.5$ and $\boldsymbol{\mu}_0 = (0.5, 0.5, \cdots, 0.5)^T$ where length of $\boldsymbol{\mu}$ is 2×numbers of data. We then use Line Search SQP algorithm with flexible penalty functions mentioned in Section 3 to get optimized $\boldsymbol{\alpha}$s and classify 170 testing data:

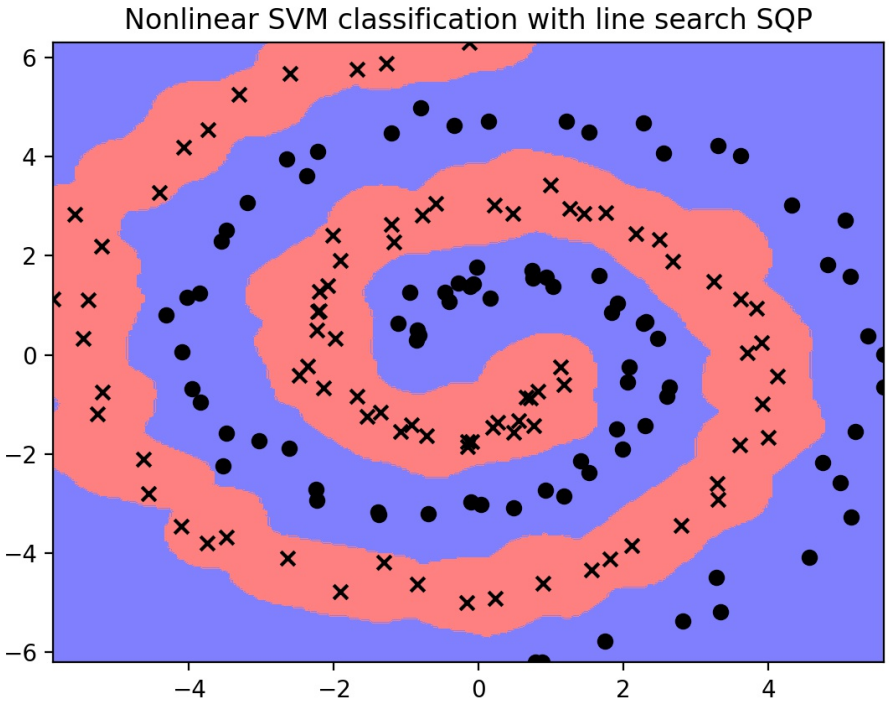


Fig2: Two spiral data sets (testing)

As it is shown in Fig2, we classify the testing data with 0 testing error which shows succeed in optimizing $\alpha$ in our dual SVM problem.

## 4.1 Convergence Result

According to Huang and Madden(2015) [4], global convergence can be expected if following assumptions are satisfied:

1. $\{x_k\}$ and $\{x_k + \alpha p_k\}$ are contained in a compact and convex set $D$.
2. The objective function $g$ and the constraint functions $c_\epsilon$, $c_I$ are twice continuously differentiable on an open set containing $D$.
3. There exists an $M > 0$ such that the Hessian matrices $H_k$ satisfy $||H_k||_2 \leq M$ for all k.

The first condition means each subsequence of compact set is convergenct, that is, $\{x_k\}$ generated by algorithm should be converged.

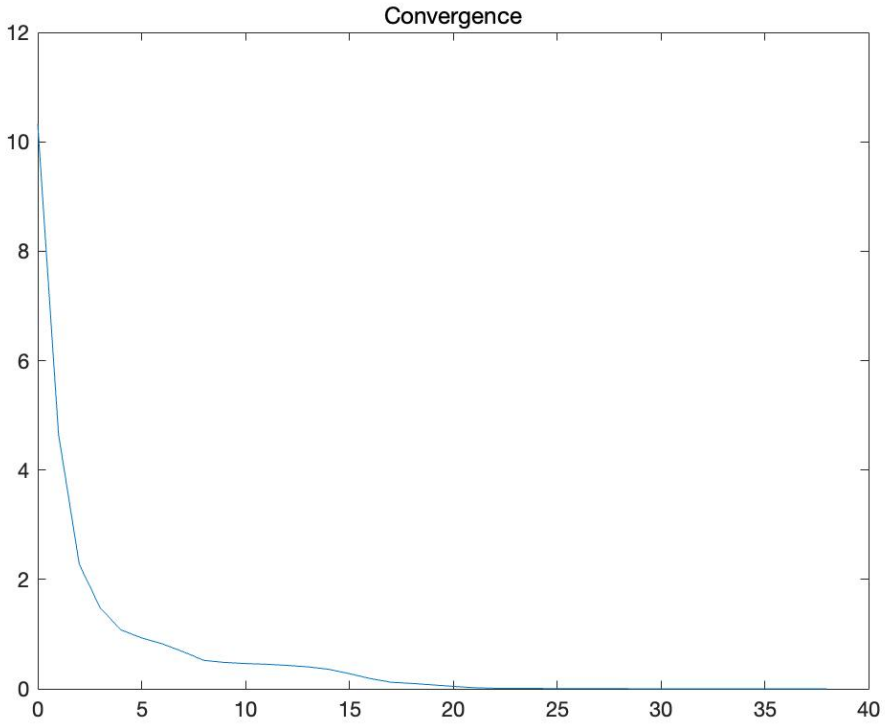



Fig3: Convergence of $\{x_k\}$

Fig3 proved that our algorithm generates converged $\{x_k\}$, which satisfy the first condition. As to the second condition, from function (58)-(61) we know that both $c_\epsilon$ and $c_I$ are twice continuously differentiable. Damped BFGS algorithm ensures each approximated hessian matrix is positive definite and finally close to the real hessian, which satisfy the third condition. To conclude, global convergence can be expected for our problem.

Besides, according to Boggs and Tolle(1995) [1], to ensure global convergence, a merit function $\phi$, which measures progress, whose reduction implies the reduction to an optimal point, must used throughout the iteration process in SQP algorithm. Here in our flexible line search SQP algorithm, we use $l_1$ penalty function as our merit function in each iteration to check whether the trial step should be accepted, thus global convergence could be ensured.

Boggs and Tolle [1] combined line search SQP with BFGS method, where the choice of matrix $H_k$ is a finite difference approximation of the Hessian of the Lagrangian in each iteration, and named this algorithm as "PSB-SQP" algorithm. Another scheme, which they called the Powell-SQP method, is to use damped BFGS to ensure a positive definite approximated Hessian in each iteration. To make sure the convergence rate, Boggs and Tolle [1] has given a theorem of local convergence properties.

To discuss convergence of line search SQP algorithm, the asymptotic rate of convergences play an important role. The convergence can be investigated with Q-convergence, the general Q-convergence is defined as:

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||^p} \leq M,\ M > 0,\ \forall k\ sufficiently\ large,\ p > 1 \tag{64}$$

Q-linear convergence is defined:

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||^p} < M,\ r \in (0,1),\ \forall k\ sufficiently\ large \tag{65}$$

We will also encounter a measure of the average rate of convergence called the $R-rate$. A sequence $x_k$ converges $R-linearly$ if the sequence $\{||x_k - x^*||\}$ is bounded by a Q-linearly convergence sequence.

There are two relationships between $Q-rate$ and $R-rate$. First, $m$-step $Q-linear$ convergence implies an $R-linear$ rate of convergence. Secondly, a sequence of $Q-rate$ vectors indicates the same $R-rate$ of convergence of its components(Boggs and Tolle, 1995).

**Theorem 4.1.1** Suppose that $H_k$ is positive definite and let $H_0$ be an initial positive definite matrix.

Suppose that the sequence $\{(x_k, \lambda_k)\}$ is generated by the SQP algorithm with sequence of matrix approximated by the BFGS update(57). Then, if $||x_0 - x^*||$ and $||H_0 - H^*||$ are sufficiently small and $\lambda_0$ is appropriate, the sequence $\{H_k\}$ is uniformly continuous and if

$$\lim_{k\to\infty} \frac{||P^k(H_k - H*)(x_{k+1} - x_k)||}{||(x_{k+1} - x_k)||} = 0 \tag{66}$$

where

$$P(x) = I - \nabla h(x)[\nabla h(x)^T \nabla h(x)]^{-1} \nabla h(x)^T \tag{67}$$

The iterates $(x_k, \lambda_k)$ converge superlinearly to the optimal solution. In addition the **x**-iterates converge superlinearly and the multipliers converge R-superlinearly.

Based on our numerical result generated by MATLAB, we plot Q-convergence of generated $\{x_k\}$. When p=1:

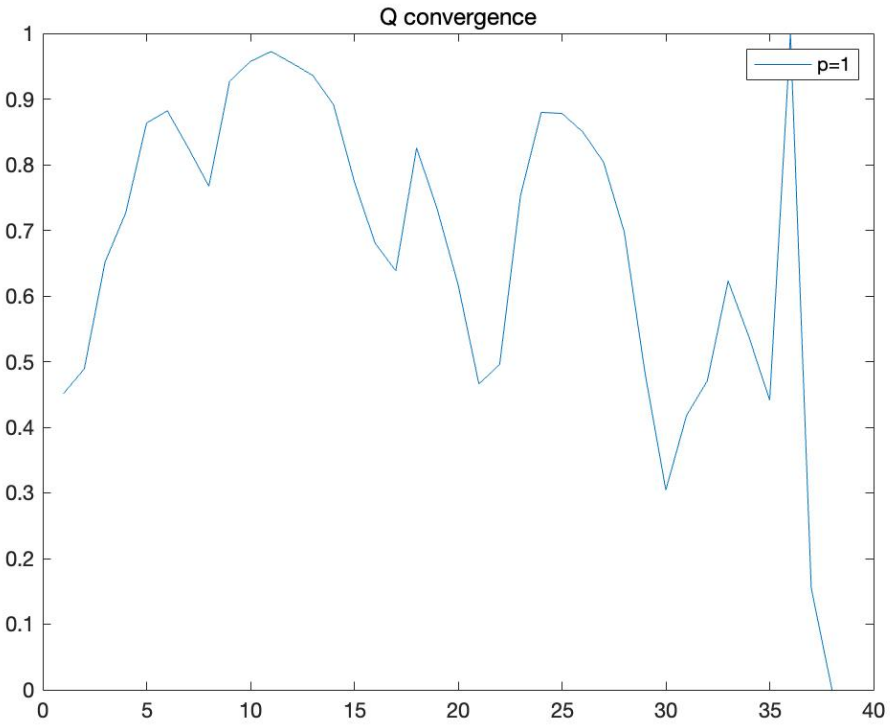
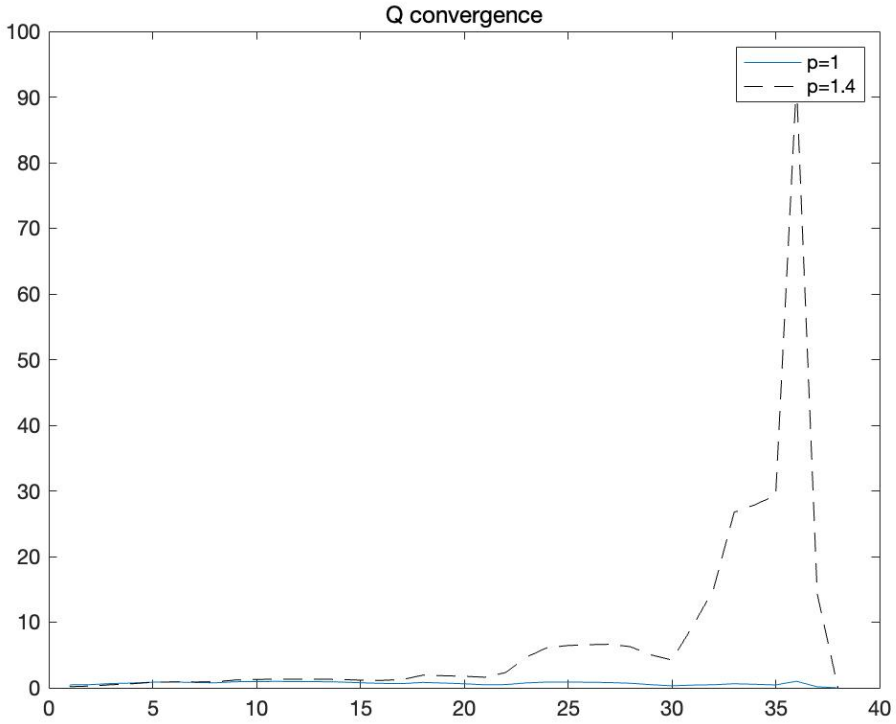



Fig4: Q-linear convergence

From Fig4 we can observe that all Q-convergence rate is from 0 to 1, which shows the algorithm converges at least a linear rate. When p=1.4 and p=2, we have:



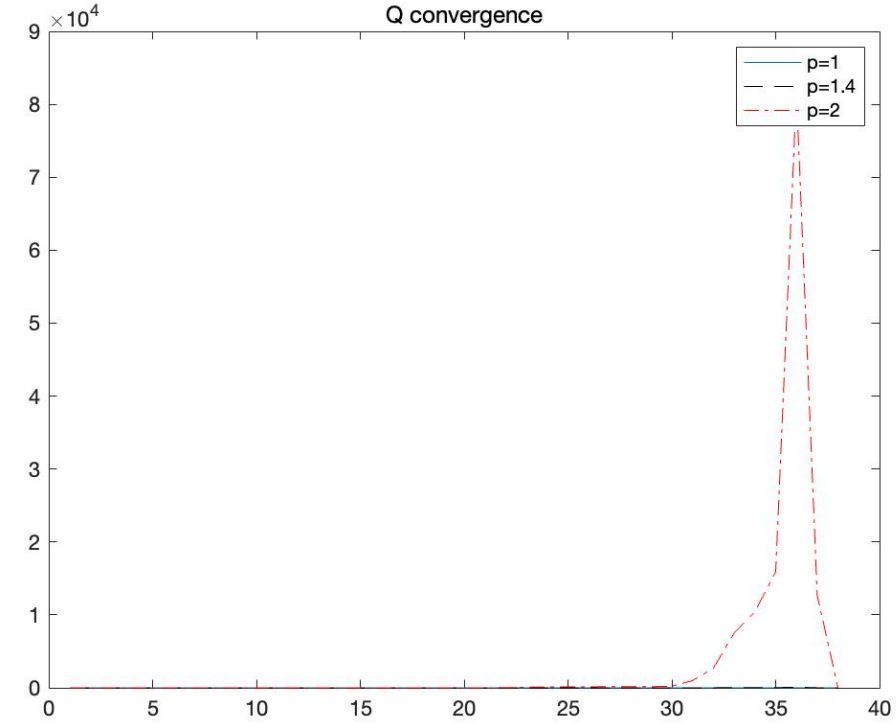




Fig5: Q-convergence

As we can see in Fig5, when p=1.4 and p=1.5, Q-convergence could not be bounded by a fixed M. To conclude, the algorithm converges linearly.

Combined with theorem mentioned before, we could observe that **x**-iterates has a local Q-superlinear convergence rate and the Lagrange multipliers has a local R-superlinear convergence rate. Through our numerical result and the assumption raised by Huang and Madden, $\{x_k\}$ converges globally and linearly.

### 4.2 Performance complexity

The CPU time of our algorithm costs 14.56 seconds and the memory used is 7591537.56 kb. In the algorithm we use two nested while loop, one is update variable until max iterations or satisfies stopping criteria, the other one is in each iteration, we backtracking to find an acceptable step length. Set we iterate n times and in each iteration, we backtracking m times, then the time complexity of our algorithm is O(nlog2m).

## 5 Summary

This report classifies random spiral data with support vector machine, where a flexible line search SQP algorithm used to do the optimization.

The basic idea of SQP algorithm is to construct a quadratic programming subproblem in each iteration, and then to use the solution to this subproblem to construct a better approximation of $x_{k+1}$. SQP may share the characteristics of Newton-like methods, converge rapidly when the iterates are close to the solution. To decide whether the trial step is acceptable, a $l_1$ penalty function used as merit function in this report. Besides, this report use BFGS to approximate $H_k$ in each step, which is improved by Damped BFGS to ensure a positive definite Hessian. This algorithm named Powell-SQP by Boggs and Tolle in 1995.

This algorithm performs well since the trained classifier classified the testing data with 0 testing error.

# 6 Appendix

Appendix shows the MATLAB code of implemented optimization method and classification problem.

**no_project.m:**

We generate two different types of random spiral data as training data and testing data, use Line Search SQP algorithm to optimize the constraint optimization problem in SVM and use the trained alphas to classify testing data in this package.

```
%%
clc; clear;
load seamount
C=10; % regularzation parameter

%% SVM
% generate data
[xTr, yTr, xTe, yTe]=sprialdata(170);

figure;
scatter(xTr(yTr==1,1),xTr(yTr==1,2),'^')
hold on
scatter(xTr(yTr≠1,1),xTr(yTr≠1,2),'d')
legend('+1','-1');

% compute gaussian kernel matrix: K
sigma=10;
K=gaussK(xTr,xTr,sigma);

yTr=yTr'; % now yTr:n*1
% dual optimization problem
A=(yTr*yTr').*K; % n*1 1*n .* n*n --> n*n

F.f=@(alpha) 0.5*alpha'*A*alpha-sum(alpha);
F.df=@(alpha) A*alpha-ones(size(yTr,1),1);
F.d2f=@(alpha) A;

c.h=@(alpha) alpha'*yTr;
c.dh=@(alpha) yTr';

alpha0=zeros(size(xTr,1),1);
lambda0=0.5;
mu0=0.5*ones(2.*size(xTr,1),1);
```

```
34
35 C_soft=C*ones(size(alpha0,1),size(alpha0,2)); % use soft margin
36
37 I.h=@(alpha) [alpha;C_soft-alpha]';
38 I.dh=@(alpha) [eye(size(alpha0,1)),-1*eye(size(alpha0,1))]';
39
40 % optimization alphas
41 [xMin,fMin,info,iter,mu_k]=sqpLine(F,c,I,alpha0,lambda0,mu0);
42
43
44 %%
45 [bias]=recoverBisas(K,yTr,xMin,C);
46 svmclassify=@(x) (xMin.*yTr)'*gaussK(xTr,x,sigma)+bias;
47
48 predsTr=svmclassify(xTr);
49 pred_err=length(yTr(yTr≠sign(predsTr')))/length(yTr)
50
51 predsTe=svmclassify(xTe);
52 pred_err_te=length(yTe(yTe≠sign(predsTe)))/length(yTe)
53
54 figure;
55 scatter(xTe(sign(predsTe)==1,1),xTe(sign(predsTe)==1,2))
56 hold on
57 scatter(xTe(sign(predsTe)≠1,1),xTe(sign(predsTe)≠1,2))
58 legend('+1','-1');
59
60 % visclassifier(svmclassify,xTe,yTe)
61
62 %% convergence
63
64 con_SQP=convergenceHistory(info,[],F,2);
65 figure;
66 plot(0:length(con_SQP.x)-1,con_SQP.x);hold on;title('Convergence');
67
68 p=1;
69 rs1=(con_SQP.x(2:end))./((con_SQP.x(1:end-1)).^p);
70 ps=1.4;
71 rss=(con_SQP.x(2:end))./((con_SQP.x(1:end-1)).^ps);
72 p=2;
73 rs2 = (con_SQP.x(2:end))./((con_SQP.x(1:end-1)).^p);
74 figure, plot(rs1); hold on; title('Q convergence'); legend('p=1')
75 figure, plot(rs1); hold on; plot(rss,'k--'); title('Q convergence'); ...
       legend('p=1',['p=' num2str(ps)])
76 figure, plot(rs1); hold on; plot(rss,'k--'); plot(rs2,'r-.'); title('Q convergence'); ...
```

```
        legend('p=1',['p=' num2str(ps)],'p=2')
```

**spiraldata.m:**

In this package we generate random spiral data.

```
function [xTr, yTr, xTe, yTe]=sprialdata(N)
rng(24) %24
r=linspace(1,2*pi,N);
xTr1=[sin(2.*r).*r;cos(2.*r).*r]';
xTr2=[sin(2.*r+pi).*r;cos(2.*r+pi).*r]';
xTr=[xTr1;xTr2];
xTr=xTr+normrnd(0,1,size(xTr,1),size(xTr,2))*0.2;
yTr=[ones(1,N),-1.*ones(1,N)];

xTe=xTr(1:2:end,:);
yTe=yTr(:,1:2:end);
xTr=xTr(2:2:end,:);
yTr=yTr(:,2:2:end);
```

**sqpLine.m:**

In this package, we use Line Search SQP with a flexible penalty function to solve the optimization problem in SVM.

```
function [xMin,fMin,info,iter,mu_k]=sqpLine(F,c,I,x0,lambda0,mu0)
% F: objective function
% c: equality constraints
% i: inequality constraints
% x0: initial points
% lambda0: initial lambda
% k: iteration numbers
maxIter=500;
tau=0.5;
eta=0.1;
sigma=1;
pi_k=1/sigma;
x_k=x0;
% equality
lambda_k=lambda0;
l=length(lambda0);
```

```
% inequality
mu_k=mu0;
m=length(mu0);
epsilon1=1e-6; epsilon2=1e-6;
% objective
f_k=F.f(x_k);
df_k=F.df(x_k);
% equality
h_k=c.h(x_k);
Ae_k=c.dh(x_k);
% inequality
v_k=I.h(x_k);
Ai_k=I.dh(x_k);
A_k=[Ae_k;Ai_k];
% initial H0 for BFGS
H_k=eye(length(x0));
% initial information
info.xs=x0;
info.lambdas=lambda0;

iter=0;
while(iter<maxIter)
    % slove subsequence quadratic programme
    [pk,lambda,mu]=qpsubp(df_k,H_k,Ae_k,h_k,Ai_k,v_k);
    lambda_k=lambda;
    mu_k=mu;
    mp1=norm(h_k,2)+norm(max(-v_k,0),2);
    dL=dLagrange(lambda_k,mu_k,df_k,Ae_k,Ai_k);

    if (norm(dL,2)<epsilon1)&&(mp1<epsilon2)
        break;
    end

    % penalty parameter update, set mu=1/sigma
    mu_eps=1e-4;
    r_k=max(norm(lambda_k,inf),norm(mu_k,inf));

    if (pi_k>r_k)
        pi_k=pi_k;
    else
        pi_k=r_k+mu_eps;
    end

    % backtracking line search with algorithm 18.3 method
```

```
61      Phi.phi=@(x,pi_k) F.f(x)+pi_k*(norm(c.h(x),1)+norm(max(-I.h(x),0),1));
62      Phi.dphi=@(x,pi_k,pk) F.df(x)'*pk+pi_k*(norm(c.h(x),1)+norm(max(-I.h(x),0),1));
63      i=0;
64      while(i≤20)
65          if(Phi.phi(x_k+tau^i*pk,pi_k)-Phi.phi(x_k,pi_k)<eta*tau^i*Phi.dphi(x_k,pi_k,pk))
66              mk=i;
67              break;
68          end
69          i=i+1;
70          if (i==20)
71              mk=10;
72          end
73      end
74      alpha_k=tau^mk;
75      x_k_1=x_k+alpha_k*pk;
76      df_k_1=F.df(x_k_1); % new gradient objective function-not update yet
77      Ae_k_1=c.dh(x_k_1); % new gradient equality constrains-not update yet
78      Ai_k_1=I.dh(x_k_1); % new gradient inequality constrains-not update yet
79
80      h_k=c.h(x_k_1); % update equality constrains
81      v_k=I.h(x_k_1); % update inequality constrains
82      A_k=[Ae_k_1;Ai_k_1]; % combine equality with inequality
83
84      p_lambda=pinv(A_k)'*df_k_1;
85
86      lambda_k=p_lambda(1:l);
87      mu_k=p_lambda(l+1:l+m);
88
89      % damped BFGS updating
90      s_k=x_k_1-x_k;
91      y_k=dLagrange(lambda_k,mu_k,df_k_1,Ae_k_1,Ai_k_1)-dLagrange(lambda_k,mu_k,df_k,Ae_k,Ai_k);
92
93      if(s_k'*y_k>0.2*s_k'*H_k*s_k)
94          theta=1;
95      else
96          theta=0.8*s_k'*H_k*s_k/(s_k'*H_k*s_k-s_k'*y_k);
97      end
98      rk=theta*y_k+(1-theta)*H_k*s_k;
99      H_k=H_k+rk*rk'/(s_k'*rk)-(H_k*s_k)*(H_k*s_k)'/(s_k'*H_k*s_k);
100
101     % update
102     x_k=x_k_1;
103     df_k=df_k_1;
104     Ae_k=Ae_k_1;
```

```
105      Ai_k=Ai_k_1;
106      info.xs=[info.xs,x_k];
107      info.lambdas=[info.lambdas,lambda_k];
108      iter=iter+1;
109
110  end
111  xMin=x_k;
112  fMin=F.f(x_k);
113
114  %==================== gradient Lagrange nabla L(x,lambda,mu) ===============
115  function dL=dLagrange(lambda,mu,df,Ae,Ai)
116  m1=size(Ai,1); l1=size(Ae,1);
117  if(l1==0)
118      dL=df-Ai'*mu;
119  end
120  if(m1==0)
121     dL=df-Ae'*lambda;
122  end
123  if(l1>0&&m1>0)
124      dL=df-Ae'*lambda-Ai'*mu;
125  end
```

**recoverbias.m:**

In this package, we calculate bias term of hyperplane after optimized alphas obtained.

```
1  function [bias]=recoverBisas(K,yTr,alpha,C)
2  %  function bias=recoverBias(K,yTr,alpha,C);
3  %  Solves for the hyperplane bias term, which is uniquely specified by the
4  %  support vectors with alpha values 0<alpha<C
5  %  This is most stable if we pick an alpha_i that is furthest from C and 0
6  %
7  %  INPUT:
8  %  K : nxn kernel matrix
9  %  yTr : nx1 input labels
10 %  alpha  : nx1 vector of alpha values
11 %  C : regularization constant
12 %
13 %  Output:
14 %  bias : the scalar hyperplane bias of the kernel SVM specified by alphas
15 gap=abs(alpha-0.5*C);
16 alpha_index=find(gap==min(gap));
17 bias=yTr(alpha_index)-(yTr'.*alpha')*K(:,alpha_index);
```

# References


[1] Boggs, P. T., & Tolle, J. W. (1995). Sequential quadratic programming. Acta numerica, 4, 1-51.

[2] Cristianini, N. & Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines. Cambridge University Press

[3] Howley, T., Madden, M.G. The Genetic Kernel Support Vector Machine: Description and Evaluation. Artif Intell Rev 24, 379–395 (2005). https://doi.org/10.1007/s10462-005-9009-3

[4] Huang, M., & Pu, D. (2015). A line search SQP method without a penalty or a filter. Computational and Applied Mathematics, 34(2), 741-753.

[5] Trevor, H., Robert, T., & Jerome, F. (2009). The elements of statistical learning: data mining, inference, and prediction.

[6] Smola, A. J., Schölkopf, B., & Müller, K. R. (1998). The connection between regularization operators and support vector kernels. Neural networks, 11(4), 637-649.

[7] Wright, S., & Nocedal, J. (1999). Numerical optimization. Springer Science, 35(67-68), 7.

[8] Zhu, X., & Pu, D. (2012). Sequential quadratic programming with a flexible step acceptance strategy. Applied Mathematical Modelling, 36(9), 3992-4002.

[9] Wang, B., & Pu, D. (2013, September). Flexible penalty functions for SQP algorithm with additional equality constrained phase. In Proceedings of the 2013 International Conference on Advanced Mechatronic Systems (pp. 22-27). IEEE.