

# 逐步二次规划算法实现实验报告

内容：技术文档与算法文档

姓名：付乐豪

学号：PB20071456

时间：2023 年 4 月 30 日

目录	1
----	---

## 目录

<b>1 技术文档</b>	<b>1</b>
1.1 运行环境	1
1.2 包含文件	2
1.3 参数配置	2
1.4 调用方法	2
1.5 输出结果	3
<b>2 算法文档</b>	<b>4</b>
2.1 算法流程	4
2.2 收敛性分析	7
2.2.1 收敛性保证	7
2.2.2 收敛速率分析	7
2.3 实验结果	8
2.3.1 实验设置	8
2.3.2 实验结果	8
2.3.3 实验收敛性	8
<b>3 总结</b>	<b>9</b>

## 1 技术文档

### 1.1 运行环境

本程序使用 python3.9 版本实现，主要依赖于一下库：

- import pandas as pd

- `import numpy as np`
- `from sympy import *`
- `from sympy import symbols, Matrix, diff`
- `from cvxopt import solvers, matrix`
- `import cvxopt`
- `import matplotlib.pyplot as plt`

## 1.2 包含文件

- `SQP main.py`: 主要实现 SQP 算法的脚本文件。
- `迭代过程记录.csv`: 记录了迭代过程中的迭代点坐标以及当前迭代点的目标函数值。
- `gap.png`: 绘制了迭代过程中目标函数的下降情况。

## 1.3 参数配置

- `input_fun()`: 用户可以通过是否注释此函数, 选择在运行端手动输入目标函数还是直接修改代码运行。
- `run(func, eq_cons, ineq_cons, iters=100, epsilon=1e-5)`: 运行 SQP 算法的主函数。其中 `func` 为目标函数, `eq_cons` 为等式约束, `ineq_cons` 为不等式约束, `iters` 为最大迭代次数, `epsilon` 为迭代过程中的容忍度。其中 `iters` 默认值为 100 次, 如果需要更改必须为 `int` 类型, `epsilon` 默认值为 `1e-5`。

## 1.4 调用方法

本程序提供了两种使用办法: (1) 直接在程序的 155-161 行定义目标函数, 约束和起始点, 并注释 162 行后直接运行。如下图:

```

155 func = (1 - x[0]) ** 2 + 100 * (x[1] - x[0] ** 2) ** 2
156 ineq_cons = [-(1 - x[0] - 2 * x[1]),
157             -(1 - x[0] ** 2 - x[1]),
158             -(1 - x[0] ** 2 + x[1])]
159 eq_cons = [2 * x[0] + x[1] - 1]
160 x0 = 1.0; x1 = 3.0
161 start = Matrix([[x0, x1]]).T
162 # func, eq_cons, ineq_cons, start = input_fun()

```

图 1: 用户输入部分代码

(2) 取消 162 行的注释, 在运行段手动输入目标函数, 约束和起始点, 其中目标函数和约束中  $x$  的两个分量分别以  $x_0$  和  $x_1$  表示, 不等式约束默认为输入的函数  $\leq 0$ ; 如果已经输入了全部的不等式约束或者等式约束, 请输入空格以结束, 如下图:

```

请输入目标函数, 其中分量用x_0和x_1表示: (1 - x_0) ** 2 + 100 * (x_1 - x_0 ** 2) ** 2
如果已经输入了所有不等式或者等式约束, 请输入空格。
请逐个输入不等式约束(<=0): -(1 - x_0 - 2 * x_1)
请逐个输入不等式约束(<=0): -(1 - x_0 ** 2 - x_1)
请逐个输入不等式约束(<=0): -(1 - x_0 ** 2 + x_1)
请逐个输入不等式约束(<=0):
请逐个输入等式约束: 2 * x_0 + x_1 - 1
请逐个输入等式约束:
请输入x0的初始值: 1
请输入x1的初始值: 3

```

图 2: 手动输入范例

## 1.5 输出结果

SQP 运行过程中, 会输出每一步的迭代点和目标函数在当前点的取值。如果算法收敛, 在运行结束后会输出最优值和最优值点, 将记录的迭代过程保存在“迭代过程记录.csv”文件中, 并自动绘制迭代曲线。如果在达到设置的最大迭代次数后, 仍然没有满足终止条件, 也会输出结果, 并提示用户应该修改迭代次数和容忍度。如下图:

```
第0次迭代: 当前迭代点 x0 = 1.00, x1 = 3.00 | 迭代点目标函数值: 400.00
第1次迭代: 当前迭代点 x0 = 0.75, x1 = -0.50 | 迭代点目标函数值: 112.95
第2次迭代: 当前迭代点 x0 = 0.50, x1 = 0.00 | 迭代点目标函数值: 6.17
第3次迭代: 当前迭代点 x0 = 0.44, x1 = 0.13 | 迭代点目标函数值: 0.73
第4次迭代: 当前迭代点 x0 = 0.42, x1 = 0.17 | 迭代点目标函数值: 0.35
第5次迭代: 当前迭代点 x0 = 0.41, x1 = 0.17 | 迭代点目标函数值: 0.34
第6次迭代: 当前迭代点 x0 = 0.41, x1 = 0.17 | 迭代点目标函数值: 0.34
一共迭代了6次
取到最小值的点为: Matrix([[0.414944389633917], [0.170111220732166]])
目标函数的最小值为: 0.342717574847723
```

图 3: 算法收敛的输出结果

```
第0次迭代: 当前迭代点 x0 = 1.00, x1 = 3.00 | 迭代点目标函数值: 400.00
第1次迭代: 当前迭代点 x0 = 0.75, x1 = -0.50 | 迭代点目标函数值: 112.95
第2次迭代: 当前迭代点 x0 = 0.50, x1 = 0.00 | 迭代点目标函数值: 6.17
一共迭代了3次, 未找到最优值, 请尝试更改迭代次数iter和容忍度epsilon
```

图 4: 算法不收敛的输出结果

## 2 算法文档

### 2.1 算法流程

序列二次规划算法 (SQP) 针对带等式和不等式的非二次规划问题, 在每一步迭代点构造一个二次规划 (QP) 的子问题, 通过子问题的最优解决定原始问题的迭代方向, 再基于 goldstein condition 的一维搜索得到步长  $\alpha_k$ , 同时, 通过 BFGS 迭代算法计算每一步的 Hessian 矩阵。如下图:

非线性约束最优化  
逐步二次规划法

下面的算法是Han(1977)提出的逐步二次规划方法：

(0) 给定 $x^{(0)}$ ,  $W_0 \in \mathbb{R}^{n \times n}$ ,  $\sigma > 0$ ,  $\rho \in (0, 1)$ ,  $\varepsilon \geq 0$ , 令 $k := 0$ .

(1) 求解子问题(41)给出 $d^{(k)}$ , 如果 $\|d^{(k)}\| \leq \varepsilon$ , 则停止；否则求 $\alpha_k \in [0, \rho]$ 使得

$$P(x^{(k)} + \alpha_k d^{(k)}, \sigma) \leq \min_{0 \leq \alpha \leq \rho} P(x^{(k)} + \alpha d^{(k)}, \sigma) + \epsilon_k.$$

(2) 置 $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ , 计算 $W_{k+1}$ , 令 $k := k + 1$ , 返回第(1)步。

YZW (USTC)
Optimization Algorithms
162 / 467

图 5: 外层迭代算法

其中原始问题和二次规划子问题的表达式如下：

非线性约束最优化  
逐步二次规划法

现考虑一般的非线性约束最优化问题

$$\begin{aligned}
 \min \quad & f(x) \\
 \text{s.t.} \quad & c_i(x) = 0, i \in \mathcal{E} = \{1, \dots, m_e\}, \\
 & c_i(x) \geq 0, i \in \mathcal{I} = \{m_e + 1, \dots, m\}.
 \end{aligned} \tag{40}$$

类似地, 在第 $k$ 次迭代里求解子问题

$$\begin{aligned}
 \min \quad & \frac{1}{2} d^T W_k d + g^{(k)T} d \\
 \text{s.t.} \quad & c_i(x^{(k)}) + a_i(x^{(k)})^T d = 0, i \in \mathcal{E}, \\
 & c_i(x^{(k)}) + a_i(x^{(k)})^T d \geq 0, i \in \mathcal{I}.
 \end{aligned} \tag{41}$$

非线性约束最优化  
逐步二次规划法

在这里,  $W_k$ 是原问题Lagrange函数的Hesse阵或其近似,  
 $g^{(k)} = \nabla f(x^{(k)})$ ,  $A(x^{(k)}) = (a_1(x^{(k)}), \dots, a_m(x^{(k)}))^T = [\nabla c(x^{(k)})]^T$ .

记子问题(41)的解为 $d^{(k)}$ , 相应Lagrange乘子向量为 $\bar{\lambda}^{(k)}$ , 故有

$$\begin{cases} W_k d^{(k)} + g^{(k)} = A(x^{(k)})^T \bar{\lambda}^{(k)}, \\ \bar{\lambda}_i^{(k)} \geq 0, i \in \mathcal{I}, \\ c(x^{(k)}) + A(x^{(k)}) d^{(k)} = 0. \end{cases}$$

YZW (USTC) Optimization Algorithms 159 / 467

YZW (USTC) Optimization Algorithms 160 / 467

图 6: (40) 为原始问题, (41) 为子问题

## 2.2 收敛性分析

### 2.2.1 收敛性保证

SQP 算法的收敛性由以下定理保证：

**非线性约束最优化  
逐步二次规划法**

可证明前述逐步二次规划法的收敛性结果如下：

**定理：**假定  $f(x)$  和  $c_i(x)$  连续可微，且存在常数  $M_1, M_2 > 0$  使得

$$M_1 \|d\|^2 \leq d^T W_k d \leq M_2 \|d\|^2, \forall k \in \mathbb{N}, \forall d \in \mathbb{R}^n,$$

如果  $\|\lambda^{(k)}\|_\infty \leq \sigma$  均成立，则 Han(1977) 算法产生的点列  $\{x^{(k)}\}$  的任何聚点都是问题(40)的 K-T 点。

YZW (USTC) Optimization Algorithms 163 / 467

图 7: 算法收敛性

### 2.2.2 收敛速率分析

在程序实现过程中使用 BFGS 拟牛顿法来近似原问题的拉格朗日函数的 Hessian 阵，故该算法具有拟牛顿算法的收敛速率，即局部二次收敛性。



## 2.3 实验结果

### 2.3.1 实验设置

为了测试算法的效果，这里选择了一个优化问题来实现，问题如下：

$$\begin{aligned}
 \min_{x \in \mathbb{R}^{2 \times 2}} \quad & (1 - x_0)^2 + 100(x_1 - x_0^2)^2 \\
 s.t. \quad & x_0 + 2x_1 - 1 \leq 0 \\
 & x_0^2 + x_1 - 1 \leq 0 \\
 & x_0^2 - x_1 - 1 \leq 0 \\
 & 2x_0 + x_1 = 1
 \end{aligned} \tag{1}$$

### 2.3.2 实验结果

以下列出了尝试的初始点和最终迭代到的目标函数最小值：

表 1: 初始点和最终迭代函数值

第一坐标分量	第二坐标分量	最终函数值	迭代次数
$x_0$	$x_1$	$f(x)$	iterations
0	0	0.343	6
0	1	0.343	6
0	2	0.343	5
1	0	0.343	5
2	0	0.343	6
3	0	0.343	6
100	0	0.343	6
0	100	0.343	7
100	100	0.343	6

### 2.3.3 实验收敛性

在程序运行完之后，会自动保留迭代过程并作出收敛性的图，此处仅仅以其中一次迭代的图像作为示例：

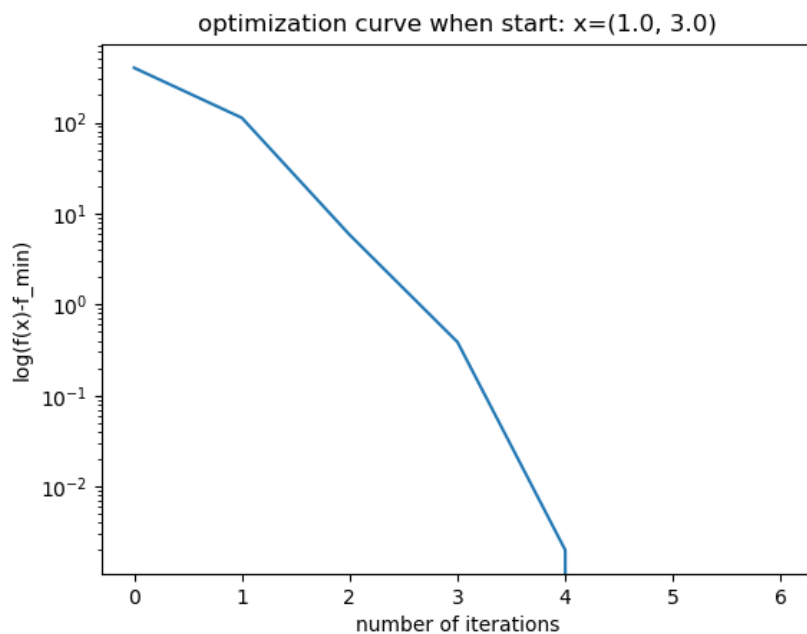


图 8: 外层迭代算法

此时一共迭代了 6 次，但由于目标函数取值在第四次之后几乎不变化，所有取对数之后 y 轴取值趋于负无穷，图中不显示。

### 3 总结

本程序实现了 SQP 算法，在内层循环时调用了 python 自带的 QP 算法，实现了算法的基本功能，并对于多个起始迭代点进行了测试，结果良好。同时本程序还提供了一定的用户接口，可以让用户自主选择使用方式。