# 最优化大作业

张晋

北京航空航天大学，数学与系统科学学院

2017 年 12 月 16 日

# 目录

# 1 5.6

$$G = \begin{bmatrix} 10 & -9 \\ -9 & 10 \end{bmatrix}, \qquad \lambda_1 = 19, \lambda_2 = 1, \qquad (\frac{\lambda_1 - \lambda_2}{\lambda_1 + x_2})^2 = 0.81$$

**Algorithm 1** Steepest-denscent method for problem(5.6)

Given $\boldsymbol{x}^{(0)}$ and $G$
Set $\boldsymbol{p}^{(0)} = -\boldsymbol{g}^{(0)}, k = 0$
**while** $\|\boldsymbol{g}^{(k)}\| > \epsilon$ **do**
  Set $\alpha_k = -\dfrac{\boldsymbol{p}^{(k)^T}\boldsymbol{g}^{(k)}}{\boldsymbol{p}^{(k)^T}\boldsymbol{G}\boldsymbol{p}^{(k)}}$
  Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}$
  Set $\boldsymbol{g}^{(k+1)} = g(\boldsymbol{x}^{(k+1)})$
  Set $\boldsymbol{p}^{(k)} = -\boldsymbol{g}^{(k)}$
  k=k+1
**end while**

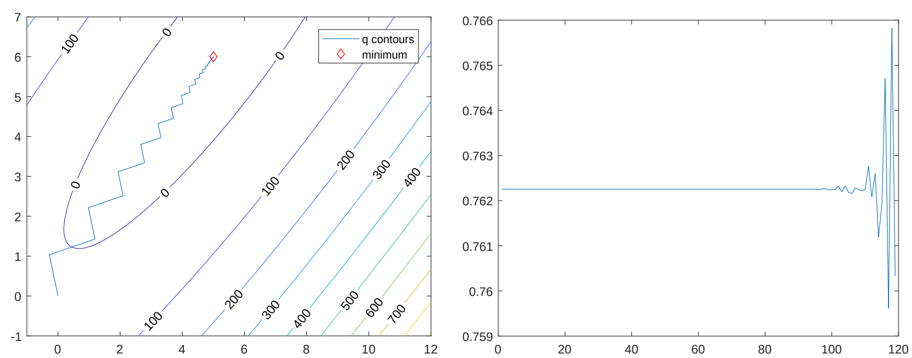



图 1: Steepest-denscent in (0,0)

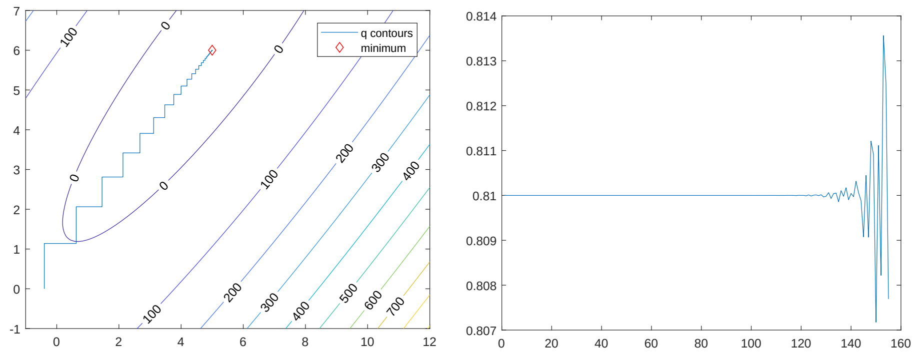


图 2: Steepest-denscent in (-0.4,0)

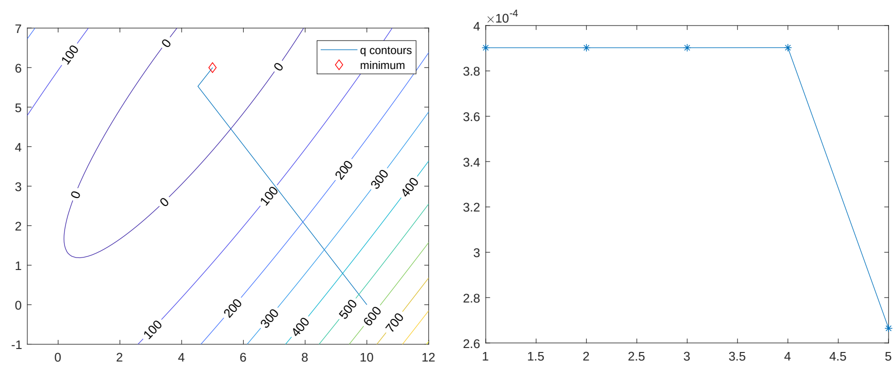


图 3: Steepest-denscent in (10,0)

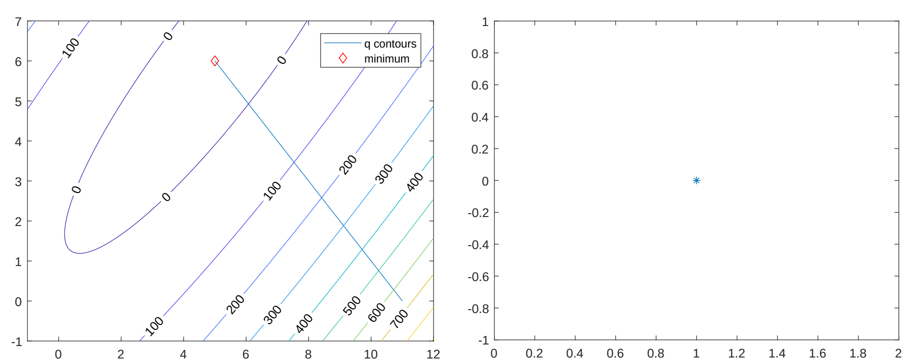


图 4: Steepest-denscent in (11,0)

表 1: 收敛因子比较

| 起始点 | $(0,0)$ | $(-0.4,0)$ | $(10,0)$ | $(11,0)$ |
|---|---|---|---|---|
| 收敛因子 | 0.762255 | 0.810000 | 0.000390 | 0 |

**分析：**

由于目标函数为凸函数，故使用梯度下降法从这四个不同的起始点出发都能收敛到全局最优点，然而线性收敛因子却互不相同.

由图像可知：在迭代开始后，函数值的收敛速度稳定在一个值左右，直到接近最优点时，收敛速度开始较大幅度波动。

而且，可以看出，初始点越接近等值线椭圆的狭长端，线性收敛因子越大，在点 $(-0.4,0)$ 处甚至达到了线性收敛因子的上界 0.81, 而离狭长端越远，收敛因子越小。这是由于梯度下降在构造搜索方向时没有充分利用到函数的二阶导数信息，在面临“峡谷”状的函数时，会反复震荡到“峡谷”的另一端，而不能直接向最优值方向前进。

容易看出，等值线椭圆的长轴端斜率为 9/10，且 $(-0.4,0)=(5,6)-0.6*(9,10)$，这说明点 $(-0.4,0)$ 刚好处在等值线椭圆的长轴上，因此也是震荡最剧烈的地方，收敛速度达到了最坏收敛速度。

# 2 5.7

**Algorithm 2** Newton method for problem(5.7)

1: Given $x^{(0)}$ and compute $g(x)=f'(x)$
2: Compute $G(x)=g'(x)$
3: Set $g^{(0)}=g(x^{(0)}), k=0$
4: **while** $|g^{(k)}|>\epsilon$ **do**
5: Set $s^{(k)}=-g^{(k)}/G^{(k)}$
6: Set $x^{(k+1)}=x^{(k)}+s^{(k)}$
7: Set k=k+1
8: **end while**

其中:

$$g(x) = f'(x) = 9 - \frac{4}{x-7}$$

$$G(x) = g'(x) = \frac{4}{(x-7)^2}$$

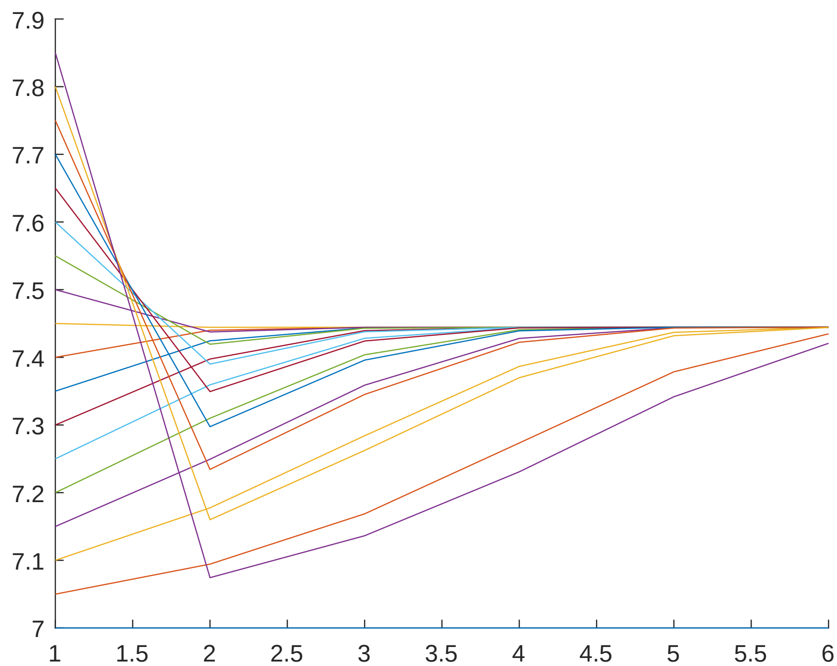
$$x^{(k+1)} = x^{(k)} - \frac{1}{4}(x^{(k)} - 7)(9x^{(k)} - 67)$$

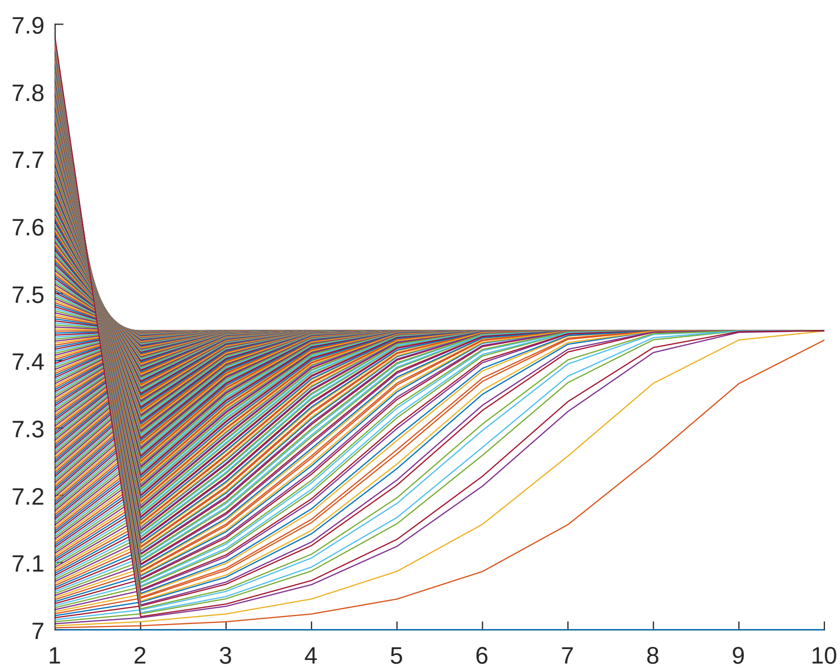
表 2: 迭代 5 次过程

| $x^{(0)}$ | 7.4 | 7.2 | 7.01 | 7.80 | 7.88 |
|---|---|---|---|---|---|
| $x^{(1)}$ | 7.44 | 7.31 | 7.019775 | 7.16 | 7.0176 |
| $x^{(2)}$ | 7.4444 | 7.403775 | 7.038670136 | 7.2624 | 7.03450304 |
| $x^{(3)}$ | 7.44444444 | 7.440722936 | 7.073975668 | 7.36987904 | 7.066327546 |
| $x^{(4)}$ | 7.444444444 | 7.444413283 | 7.135638438 | 7.431934445 | 7.122756569 |
| $x^{(5)}$ | 7.444444444 | 7.444444442 | 7.229881858 | 7.444092319 | 7.211607493 |
| $f(x^{(5)})$ | 70.24372086 | 70.24372086 | 70.94969577 | 70.24372212 | 71.11655611 |

由 MATLAB 函数 fminsearch 求得最优解为:

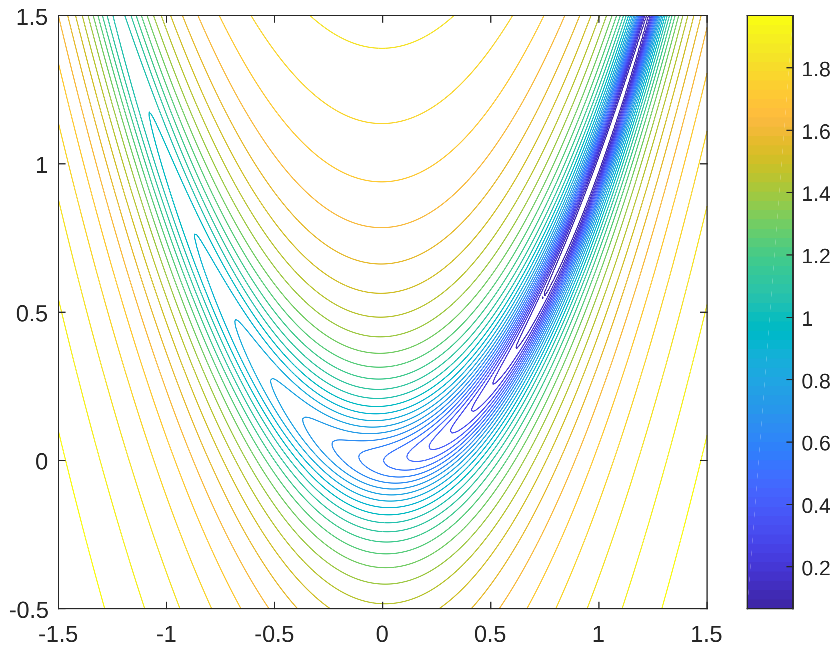
$$x^\star = 7.444421386718750, \quad f(x^\star) = 70.243720870248540$$


7.9
7.8
7.7
7.6
7.5
7.4
7.3
7.2
7.1
7
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

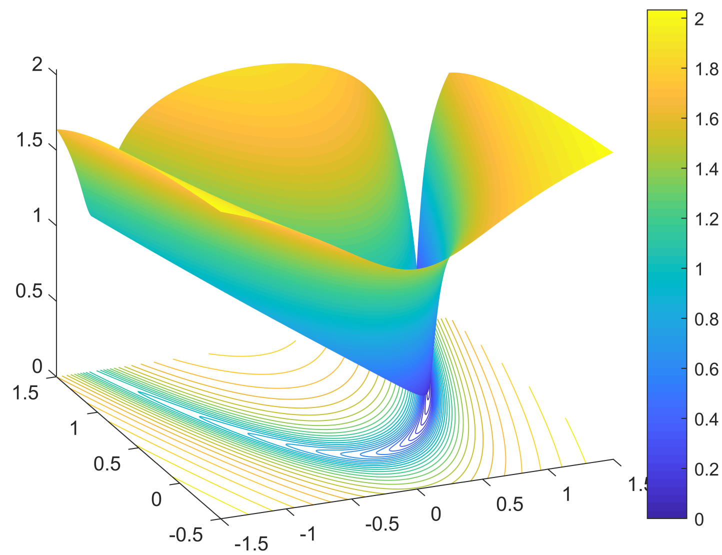

7.9
7.8
7.7
7.6
7.5
7.4
7.3
7.2
7.1
7
1
2
3
4
5
6
7
8
9
10

# 3 5.9

首先画出 Rosenbrock 函数的图像及等值线如下[1]:




[1] 由于 Rosenbrock 函数过于陡峭，因此对原函数进行了取对数处理，以便于观察其特点。




本题中 Armijo 线搜索的参数为 $\gamma = 0.5, \rho = 0.01$.

然后分别以梯度下降法和牛顿法迭代，并画出等高线、运动轨迹、迭代值如下:

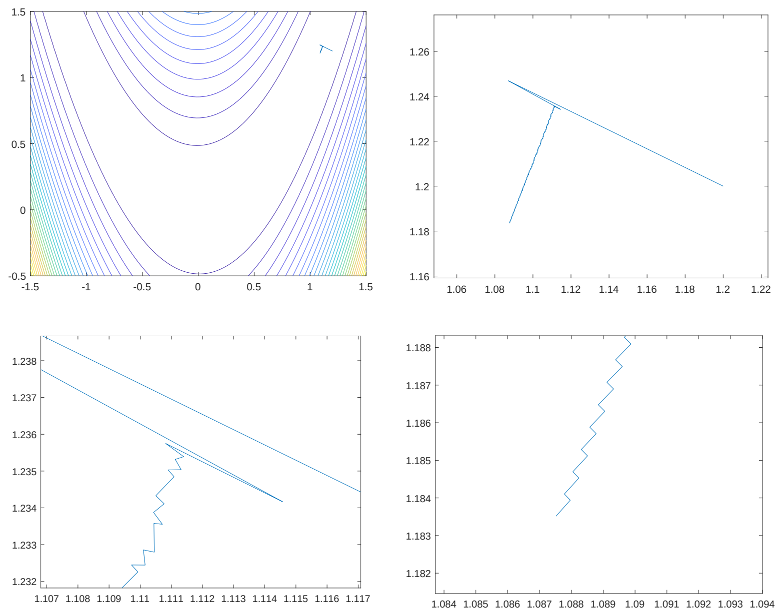


图 5: Steepest-denscent in (1.2,1.2)

Step[190]: x=[ 1.088846 1.186477 ] optim_fx=0.007973
Step[191]: x=[ 1.089052 1.186303 ] optim_fx=0.007937
Step[192]: x=[ 1.088577 1.185882 ] optim_fx=0.007924
Step[193]: x=[ 1.088779 1.185710 ] optim_fx=0.007889
Step[194]: x=[ 1.088310 1.185288 ] optim_fx=0.007874
Step[195]: x=[ 1.088507 1.185118 ] optim_fx=0.007841
Step[196]: x=[ 1.088044 1.184696 ] optim_fx=0.007825
Step[197]: x=[ 1.088236 1.184529 ] optim_fx=0.007793
Step[198]: x=[ 1.087780 1.184104 ] optim_fx=0.007776
Step[199]: x=[ 1.087965 1.183941 ] optim_fx=0.007745
Step[200]: x=[ 1.087517 1.183515 ] optim_fx=0.007727
最速下降法,共迭代 200 步
结果：
x=[ 1.087517e+00 1.183515e+00 ] optim_fx=0.007727

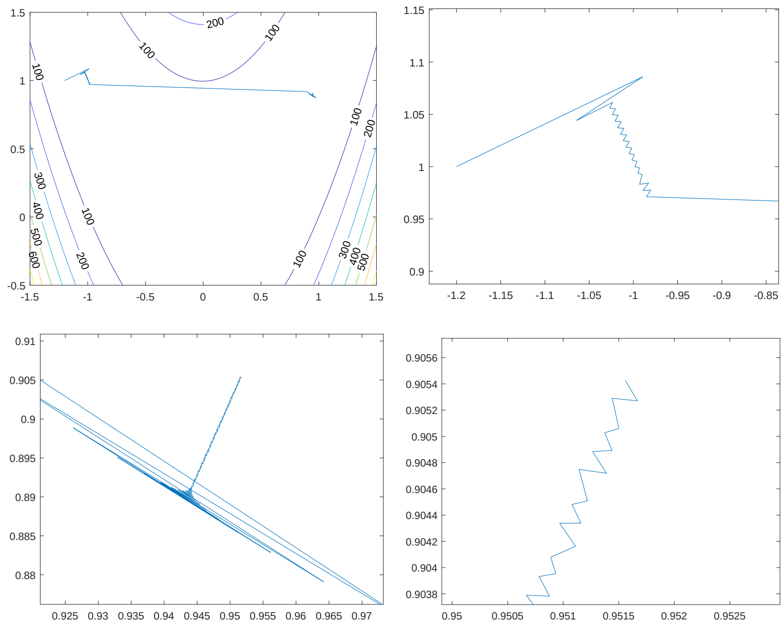


图 6: Steepest-denscent in (-1.2,1)

Step[190]: x=[ 0.951079 0.904481 ] optim_fx=0.002394
Step[191]: x=[ 0.951218 0.904509 ] optim_fx=0.002389
Step[192]: x=[ 0.951143 0.904748 ] optim_fx=0.002388
Step[193]: x=[ 0.951390 0.904719 ] optim_fx=0.002381
Step[194]: x=[ 0.951265 0.904884 ] optim_fx=0.002375
Step[195]: x=[ 0.951440 0.904892 ] optim_fx=0.002370
Step[196]: x=[ 0.951373 0.905027 ] optim_fx=0.002365
Step[197]: x=[ 0.951501 0.905060 ] optim_fx=0.002361
Step[198]: x=[ 0.951442 0.905290 ] optim_fx=0.002358
Step[199]: x=[ 0.951668 0.905271 ] optim_fx=0.002352
Step[200]: x=[ 0.951559 0.905427 ] optim_fx=0.002347
最速下降法,共迭代 200 步
结果：
x=[ 9.515588e-01 9.054274e-01 ] optim_fx=0.002347

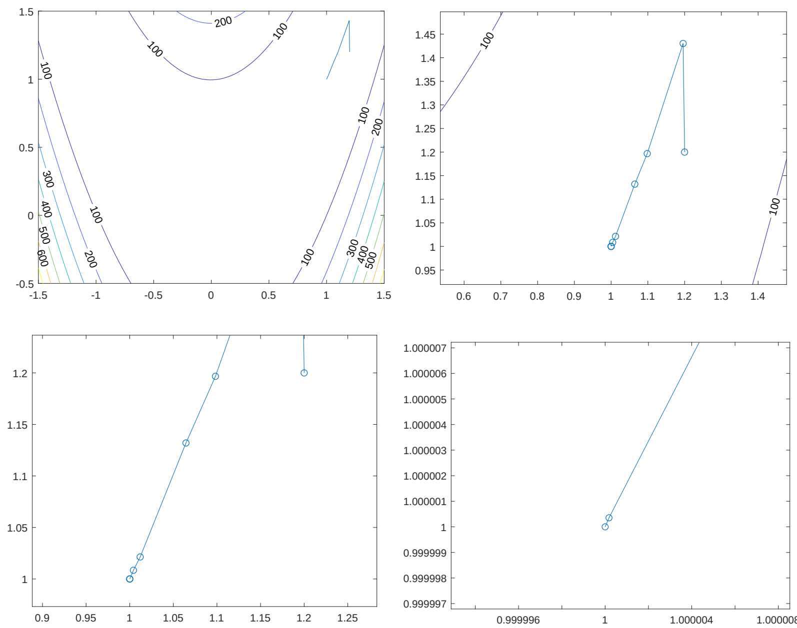


图 7: Newton-Armijo in (1.2,1.2)

Step[1]: x=[ 1.200000 1.200000 ] optim_fx=5.800000
Step[2]: x=[ 1.195918 1.430204 ] optim_fx=0.038384
Step[3]: x=[ 1.098284 1.196688 ] optim_fx=0.018762
Step[4]: x=[ 1.064488 1.131993 ] optim_fx=0.004289
Step[5]: x=[ 1.011992 1.021372 ] optim_fx=0.000903
Step[6]: x=[ 1.004261 1.008481 ] optim_fx=0.000019
Step[7]: x=[ 1.000050 1.000083 ] optim_fx=0.000000
Step[8]: x=[ 1.000000 1.000000 ] optim_fx=0.000000
Step[9]: x=[ 1.000000 1.000000 ] optim_fx=0.000000

牛顿 Armijo 回溯法,,共迭代 9 步
结果：
x=[ 1.000000e+00 1.000000e+00 ] optim_fx=0.000000

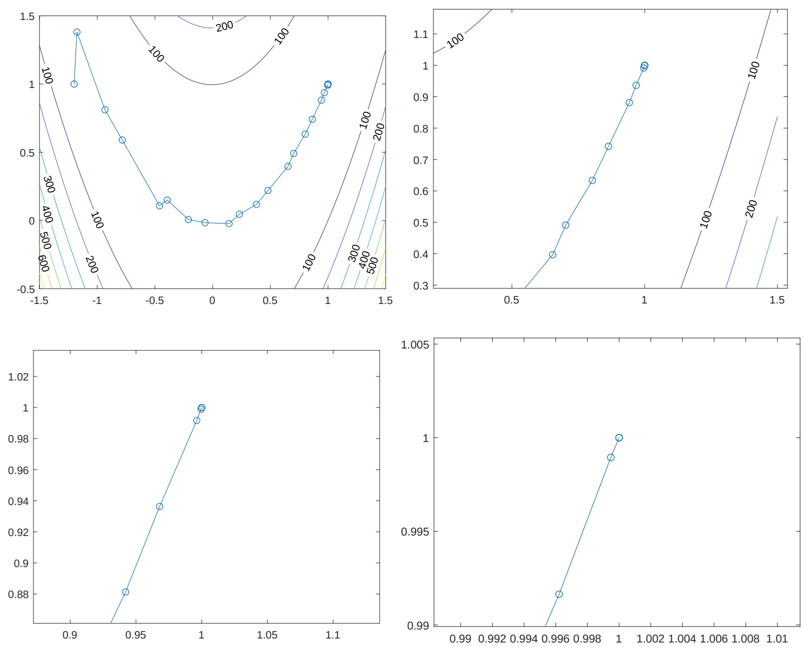


图 8: Newton-Armijo in (-1.2,1)

Step[15]: x=[ 0.802786 0.633221 ] optim_fx=0.051535
Step[16]: x=[ 0.863491 0.741931 ] optim_fx=0.019993
Step[17]: x=[ 0.942079 0.881336 ] optim_fx=0.007169
Step[18]: x=[ 0.967992 0.936337 ] optim_fx=0.001070
Step[19]: x=[ 0.996210 0.991639 ] optim_fx=0.000078
Step[20]: x=[ 0.999479 0.998948 ] optim_fx=0.000000
Step[21]: x=[ 0.999999 0.999998 ] optim_fx=0.000000
Step[22]: x=[ 1.000000 1.000000 ] optim_fx=0.000000
牛顿 Armijo 回溯法,,共迭代 22 步
结果：
x=[ 1.000000e+00 1.000000e+00 ] optim_fx=0.000000

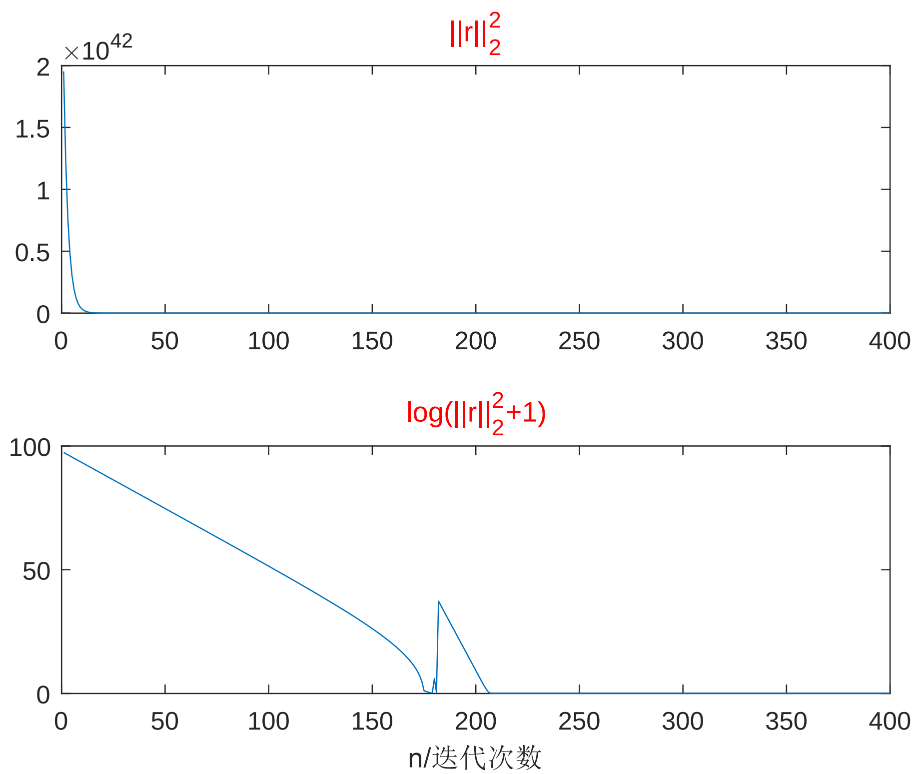
# 4 5.19

迭代次数和求解的值如下:

| **n=5** | **n=8** | **n=12** | **n=20** |
|---|---|---|---|
| k=6 | k=19 | k=35 | k=66 |
| x | x | x | x |
| 5.00E+00 | 5.90E-11 | -9.61E+00 | -1.10E+01 |
| -1.20E+02 | -6.97E-11 | 8.15E+02 | 1.05E+03 |
| 6.30E+02 | -5.16E-10 | -1.65E+04 | -2.40E+04 |
| -1.12E+03 | 1.12E-09 | 1.36E+05 | 2.20E+05 |
| 6.30E+02 | 3.17E-10 | -5.36E+05 | -9.65E+05 |
| | -6.55E-10 | 1.03E+06 | 1.99E+06 |
| | -6.57E-10 | -6.43E+05 | -1.25E+06 |
| | 4.59E-10 | -6.58E+05 | -1.34E+06 |
| | | 8.04E+05 | 8.83E+05 |
| | | 6.63E+05 | 1.69E+06 |
| | | -1.24E+06 | 3.88E+05 |
| | | 4.66E+05 | -1.31E+06 |
| | | | -1.71E+06 |
| | | | -5.28E+05 |
| | | | 1.21E+06 |
| | | | 2.00E+06 |
| | | | 9.45E+05 |
| | | | -1.43E+06 |
| | | | -2.65E+06 |
| | | | 1.89E+06 |

# 5　5.27

此题采用线搜索确定步长时，得到的结果误差极大，因为 $\phi'(0)$ 在离稳定点较远时数量级高达 $10^{40}$ 量级，导致线搜索得到的步长极小，几乎为 0，无法收敛, 经反复调整参数都没能取得好的结果，最后只好手动确定步长 $\alpha_k = 0.05$，此时效果良好，残量的 2-范数随迭代次数的下降情况如下：（由于数量级巨大，为了更好的显示残差的波动，将原图中将残差取对数处理后并排参考）




又采用 MATLAB 中优化工具箱中的 lsqnonlin 函数进行拟合，得到的结果比我的程序算出来的略好，将两者进行比较，比较结果如下：

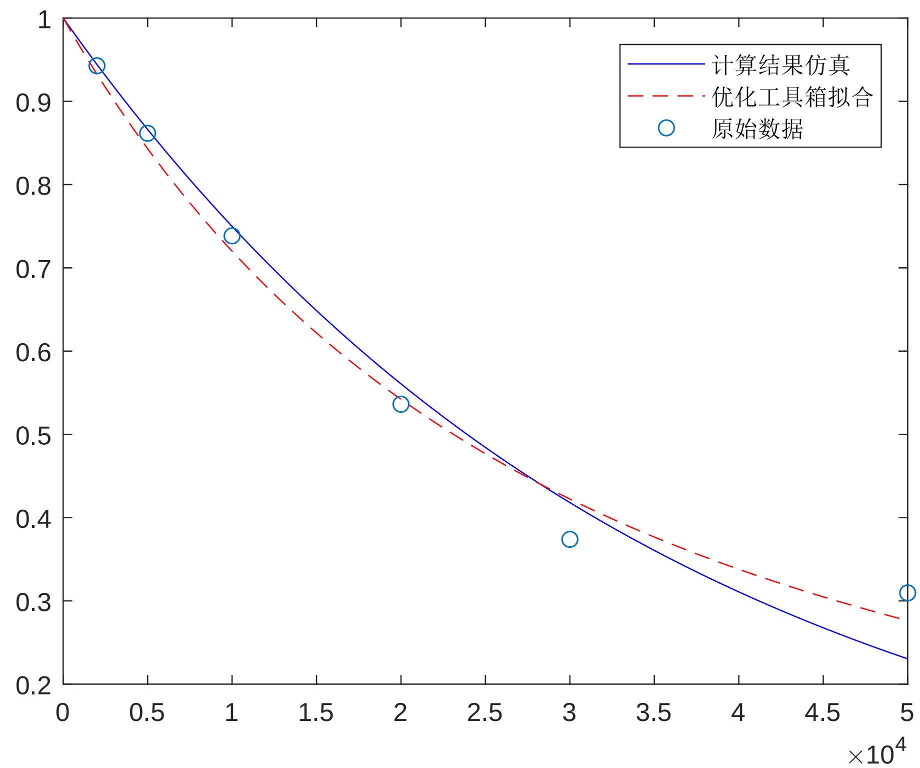



表 3: 结果比较

| | 程序计算 | 工具箱拟合 |
|---|---|---|
| stv | 0.125639950119876 | 0.104420208306470 |
| $x_1$ | 3.323336983976929e-04 | -0.009615612533368 |
| $x_2$ | 3.516673367231929e+02 | -19.446505801429495 |