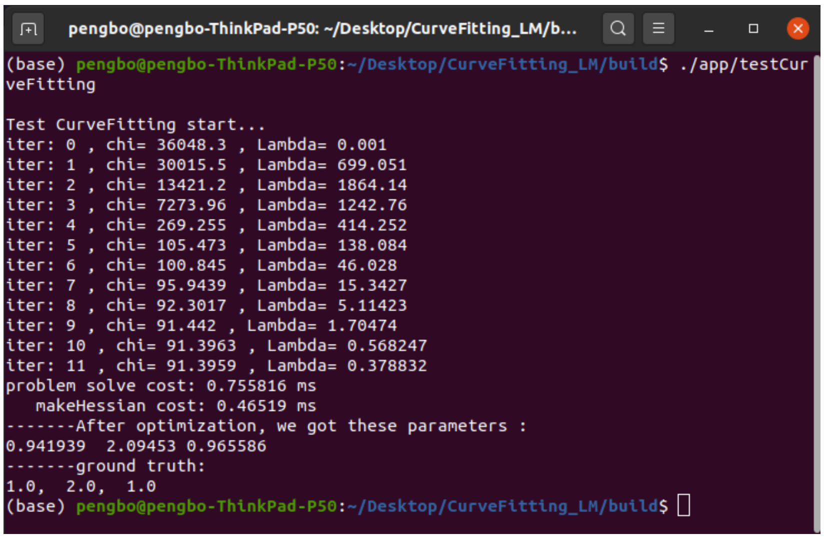
从零开始手写 VIO - 作业 3

peng00bo00

August 1, 2020

1. (a) 阻尼因子随迭代次数变化曲线如 Fig.1-Fig.2所示

   补充：检查了一下阻尼因子曲线没发现什么错误，这里把程序运行截图贴出来希望助教老师指正



```
(base) pengbo@pengbo-ThinkPad-P50:~/Desktop/CurveFitting_LM/build$ ./app/testCur
veFitting

Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 30015.5 , Lambda= 699.051
iter: 2 , chi= 13421.2 , Lambda= 1864.14
iter: 3 , chi= 7273.96 , Lambda= 1242.76
iter: 4 , chi= 269.255 , Lambda= 414.252
iter: 5 , chi= 105.473 , Lambda= 138.084
iter: 6 , chi= 100.845 , Lambda= 46.028
iter: 7 , chi= 95.9439 , Lambda= 15.3427
iter: 8 , chi= 92.3017 , Lambda= 5.11423
iter: 9 , chi= 91.442 , Lambda= 1.70474
iter: 10 , chi= 91.3963 , Lambda= 0.568247
iter: 11 , chi= 91.3959 , Lambda= 0.378832
problem solve cost: 0.755816 ms
   makeHessian cost: 0.46519 ms
-------After optimization, we got these parameters :
0.941939  2.09453 0.965586
-------ground truth:
1.0,  2.0,  1.0
(base) pengbo@pengbo-ThinkPad-P50:~/Desktop/CurveFitting_LM/build$
```

Figure 1: 阻尼因子变化

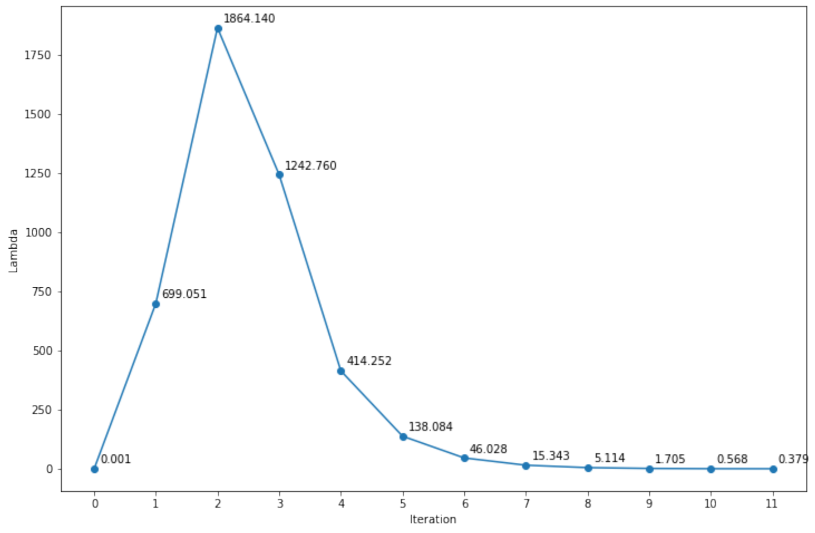
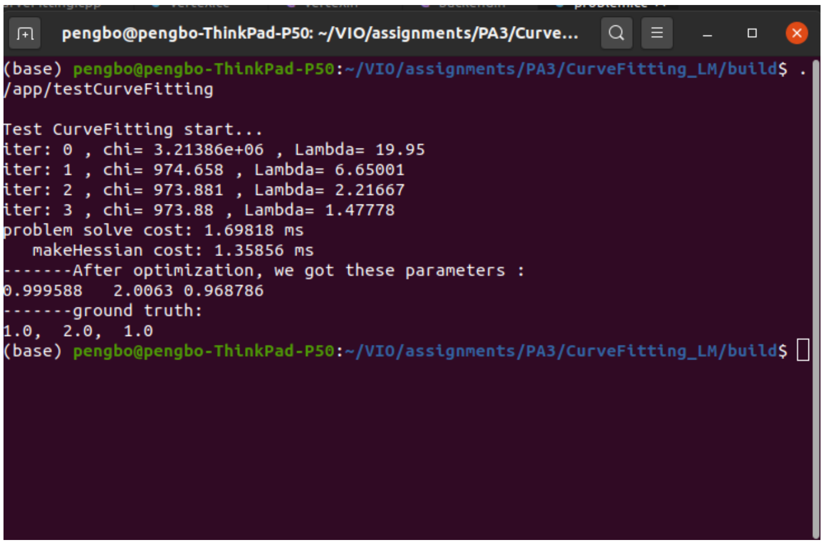


Figure 2: 阻尼因子曲线

   (b) 修改代码后对曲线进行参数估计得到程序运行截图如 Fig.3所示，可以看出估计值与真实值是非常接近的

   (c) 首先尝试使用 Marquardt 策略如下，程序运行截图如 Fig.4所示



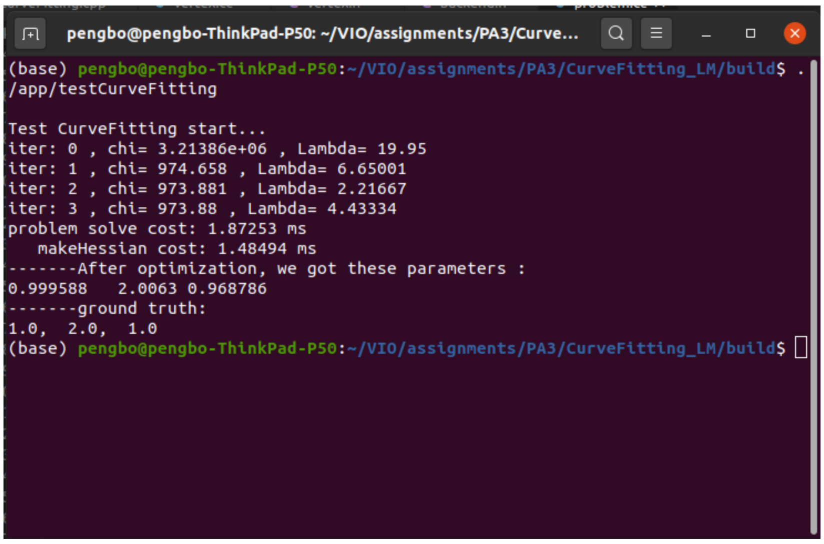
```
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$ ./app/testCurveFitting

Test CurveFitting start...
iter: 0 , chi= 3.21386e+06 , Lambda= 19.95
iter: 1 , chi= 974.658 , Lambda= 6.65001
iter: 2 , chi= 973.881 , Lambda= 2.21667
iter: 3 , chi= 973.88 , Lambda= 1.47778
problem solve cost: 1.69818 ms
   makeHessian cost: 1.35856 ms
-------After optimization, we got these parameters :
0.999588   2.0063 0.968786
-------ground truth:
1.0,  2.0,  1.0
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$
```

Figure 3: 曲线参数估计

Algorithm 1 Marquardt 策略

```
1: if $\rho < 0.25$ then
2:     $\mu = 2 \times \mu$
3: else if $\rho > 0.75$ then
4:     $\mu = \mu/3$
5: end if
```



```
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$ ./app/testCurveFitting

Test CurveFitting start...
iter: 0 , chi= 3.21386e+06 , Lambda= 19.95
iter: 1 , chi= 974.658 , Lambda= 6.65001
iter: 2 , chi= 973.881 , Lambda= 2.21667
iter: 3 , chi= 973.88 , Lambda= 4.43334
problem solve cost: 1.87253 ms
   makeHessian cost: 1.48494 ms
-------After optimization, we got these parameters :
0.999588   2.0063 0.968786
-------ground truth:
1.0,  2.0,  1.0
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$
```

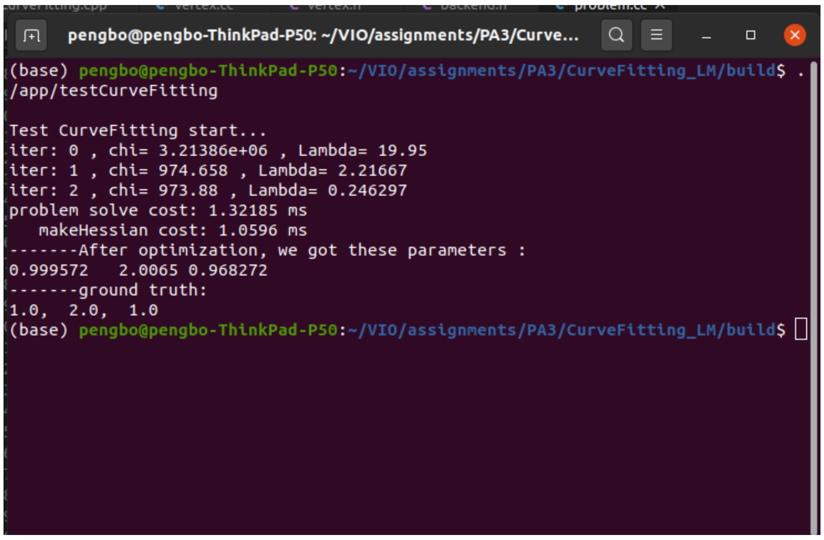
Figure 4: Marquardt 策略曲线参数估计

然后考虑论文中提出的策略如下，其中取 $\lambda_{\downarrow} = 9, \lambda_{\uparrow} = 11$,，程序运行截图如 Fig.5所示

Algorithm 2 论文策略

```
1: if $\rho > 0$ then
2:     $\mu = \max\{\mu/\lambda_{\downarrow}, 10^{-7}\}$
3: else
4:     $\mu = \min\{\mu\lambda_{\uparrow}, 10^{7}\}$
5: end if
```

对比 Fig.3-Fig.5可以发现 3 种策略都可以实现对曲线的参数估计，其中原始策略与 Marquardt 策略的效率接近而论文策略的效率要略高一些，而误差方面原始策略与 Marquardt 策略计算得到的结果具有相同的误差且该误差要略低于论文策略的误差。



```
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$ ./app/testCurveFitting

Test CurveFitting start...
iter: 0 , chi= 3.21386e+06 , Lambda= 19.95
iter: 1 , chi= 974.658 , Lambda= 2.21667
iter: 2 , chi= 973.88 , Lambda= 0.246297
problem solve cost: 1.32185 ms
   makeHessian cost: 1.0596 ms
-------After optimization, we got these parameters :
0.999572   2.0065 0.968272
-------ground truth:
1.0,  2.0,  1.0
(base) pengbo@pengbo-ThinkPad-P50:~/VIO/assignments/PA3/CurveFitting_LM/build$
```

Figure 5: 论文策略曲线参数估计

Table 1: 阻尼策略对比

	运行时间 (ms)	迭代次数	L2 误差
原始策略	1.698	3	0.0318
Marquardt 策略	1.872	3	0.0318
论文策略	1.322	2	0.0323

2. (a)

$$
\begin{aligned}
\mathbf{f}_{15} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta b_k^g} &= \frac{\partial}{\partial \delta b_k^g} \frac{1}{2} (\frac{1}{2} R_{b_i b_k} \exp\{[(\omega - \delta b_k^g)\delta t]_\times\}(a^{b_{k+1}} - b_k^a))\delta t^2 \\
&= \frac{1}{4}\delta t^2 \frac{\partial}{\partial \delta b_k^g} R_{b_i b_k} \exp\{[\omega \delta t]_\times\} \exp\{[-J_r \delta b_k^g \delta t]_\times\}(a^{b_{k+1}} - b_k^a) \\
&= -\frac{1}{4}\delta t^2 \frac{\partial}{\partial \delta b_k^g} R_{b_i b_{k+1}} (a^{b_{k+1}} - b_k^a)_\times [-J_r \delta b_k^g \delta t] \\
&\approx \frac{1}{4}\delta t^2 R_{b_i b_{k+1}} (a^{b_{k+1}} - b_k^a)_\times \delta t
\end{aligned}
\tag{1}
$$

(b)

$$
\begin{aligned}
\mathbf{g}_{12} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta n_k^g} &= \frac{\partial}{\partial \delta b_k^g} \frac{1}{2} (\frac{1}{2} R_{b_i b_k} \exp\{[(\omega + \frac{1}{2}\delta n_k^g)\delta t]_\times\}(a^{b_{k+1}} - b_k^a))\delta t^2 \\
&= \frac{1}{4}\delta t^2 \frac{\partial}{\partial \delta n_k^g} R_{b_i b_k} \exp\{[\omega \delta t]_\times\} \exp\{[\frac{1}{2} J_r \delta n_k^g \delta t]_\times\}(a^{b_{k+1}} - b_k^a) \\
&= -\frac{1}{4}\delta t^2 \frac{\partial}{\partial \delta n_k^g} R_{b_i b_{k+1}} (a^{b_{k+1}} - b_k^a)_\times [\frac{1}{2} J_r \delta n_k^g \delta t] \\
&\approx -\frac{1}{8}\delta t^2 R_{b_i b_{k+1}} (a^{b_{k+1}} - b_k^a)_\times \delta t
\end{aligned}
\tag{2}
$$

3. 对 J^TJ 进行特征值分解并带入 LM 算法的正则方程得到：

$$\begin{aligned}(J^TJ+\mu I)\Delta x &= (V\Lambda V^T+\mu I)\Delta x \\ &= V(\Lambda+\mu I)V^T\Delta x \\ &= -J^Tf \\ &= -F'^T\end{aligned} \tag{3}$$

求解得到

$$\Delta x = -V(\Lambda+\mu I)^{-1}V^TF'^T \tag{4}$$

记 V 的各列向量为 v_i，由于 $\Lambda+\mu I$ 为对角阵，有

$$\begin{aligned}V(\Lambda+\mu I)^{-1}V^T &= \begin{bmatrix} v_1 & v_2 & ... \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda_1+\mu} & & \\ & \frac{1}{\lambda_2+\mu} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \end{bmatrix} \\ &= \begin{bmatrix} v_1 & v_2 & ... \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda_1+\mu}v_1^T \\ \frac{1}{\lambda_2+\mu}v_2^T \\ \vdots \end{bmatrix} \\ &= \sum_{i=1}^{n} \frac{1}{\lambda_1+\mu} v_i v_i^T\end{aligned} \tag{5}$$

带入上式得到

$$\begin{aligned}\Delta x &= -V(\Lambda+\mu I)^{-1}V^TF'^T \\ &= \sum_{i=1}^{n} \frac{1}{\lambda_1+\mu} v_i v_i^T F'^T \\ &= \sum_{i=1}^{n} \frac{v_i^T F'^T}{\lambda_1+\mu} v_i\end{aligned} \tag{6}$$