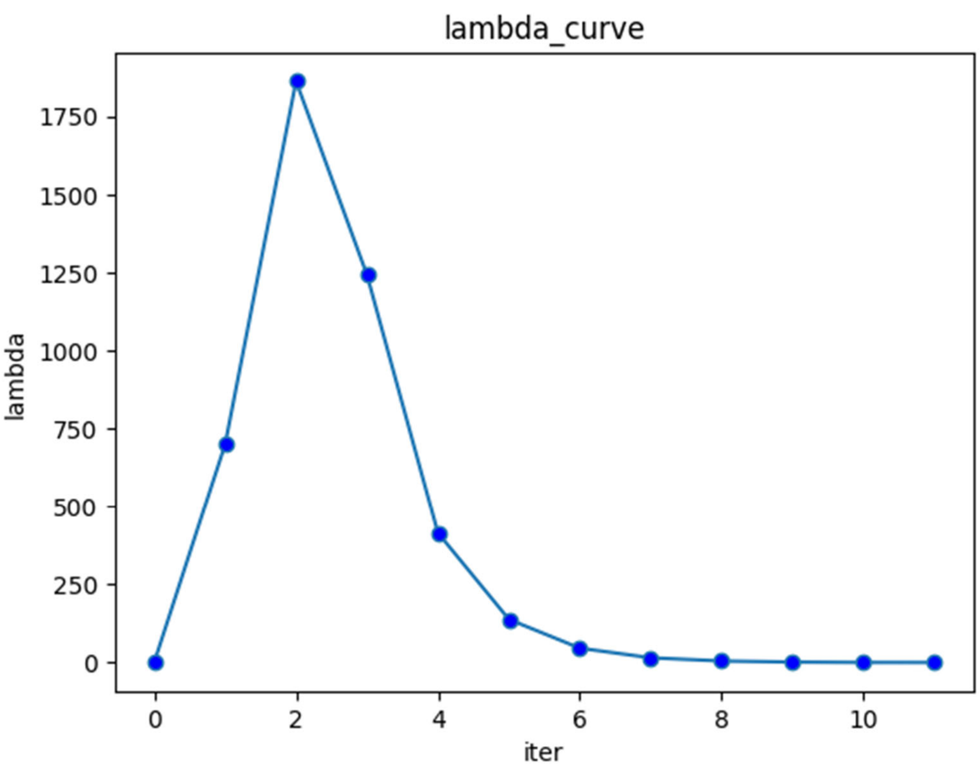
第一题

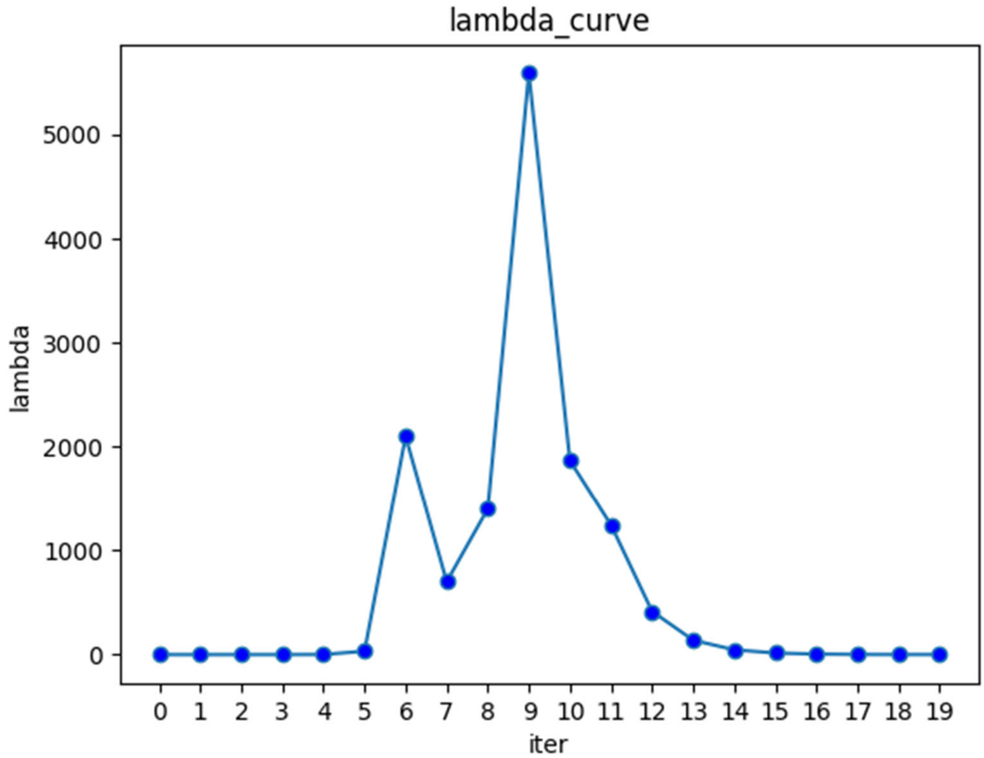
（1） 原程序运行结果如下图所示。

```
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: 20.51 ms
   makeHessian cost: 13.3646 ms
-------After optimization, we got these parameters :
0.941939  2.09453 0.965586
-------ground truth:
1.0,  2.0,  1.0
```

将 lambda 值输出到 lambda.txt 中，使用 draw_lambda.py 来画出曲线。



迭代修改，lambda 曲线（包含舍去的）：



（2） 修改的残差和雅可比计算等函数如下：

```
virtual void ComputeResidual() override
{
    Vec3 abc = verticies_[0]->Parameters();  // 估计的参数
    //residual_(0) = std::exp( abc(0)*x_*x_ + abc(1)*x_ + abc(2) ) - y_;  // 构建残差
    residual_(0) = abc(0)*x_*x_ + abc(1)*x_ + abc(2) - y_; // 构建残差
}

// 计算残差对变量的雅克比
virtual void ComputeJacobians() override
{
    Vec3 abc = verticies_[0]->Parameters();
    //double exp_y = std::exp( abc(0)*x_*x_ + abc(1)*x_ + abc(2) );

    Eigen::Matrix<double, 1, 3> jaco_abc;  // 误差为1维，状态量 3 个，所以是 1x3 的雅克比矩阵
    //jaco_abc << x_ * x_ * exp_y, x_ * exp_y , 1 * exp_y;
    jaco_abc<<x_ * x_, x_, 1.0;
    jacobians_[0] = jaco_abc;
}
```

数据点 N 取 1000，运行结果如下图所示：

```
Test CurveFitting start...
iter: 0 , chi= 7114.25 , Lambda= 0.01
iter: 1 , chi= 973.88 , Lambda= 0.00333333
iter: 2 , chi= 973.88 , Lambda= 0.00222222
problem solve cost: 28.6714 ms
   makeHessian cost: 23.5795 ms
-------After optimization, we got these parameters :
0.958923  2.06283 0.968821
-------ground truth:
1.0,  2.0,  1.0
```

以下为迭代加做的选做题

（3） 其他阻尼因子更新策略

参考文献：Gavin, H.P. (2013). The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems c ©.

$\lambda_0 = \lambda_o$; λ_o is user-specified [8].
use eq'n (13) for $\boldsymbol{h}_{\mathsf{lm}}$ and eq'n (16) for ρ
if $\rho_i(\boldsymbol{h}) > \epsilon_4$: $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{h}$; $\lambda_{i+1} = \max[\lambda_i / L_\downarrow, 10^{-7}]$;
otherwise: $\lambda_{i+1} = \min[\lambda_i L_\uparrow, 10^7]$;

修改 problem.cc 中阻尼因子更新部分为：

```
if (rho > 0 && isfinite(tempChi))    // last step was good, 误差在下降
{
    double alpha = 1. - pow((2 * rho - 1), 3);
    alpha = std::min(alpha, 2. / 3.);
    double scaleFactor = (std::max)(1. / 3., alpha);
    currentLambda_ = (std::max)(currentLambda_ / 9., 1e-7);
    ni_ = 2;
    currentChi_ = tempChi;
    return true;
} else {
    currentLambda_ = (std::min)(currentLambda_ * 11., 1e7);
    ni_ *= 2;
    return false;
}
```

程序运行结果如下：

```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 28146.3 , Lambda= 196.84
iter: 2 , chi= 23642.6 , Lambda= 2646.41
iter: 3 , chi= 17267.9 , Lambda= 3234.5
iter: 4 , chi= 7600.73 , Lambda= 359.388
iter: 5 , chi= 262.917 , Lambda= 39.9321
iter: 6 , chi= 97.028 , Lambda= 4.43689
iter: 7 , chi= 91.6281 , Lambda= 0.492988
iter: 8 , chi= 91.3962 , Lambda= 0.0547765
iter: 9 , chi= 91.3959 , Lambda= 0.00608628
problem solve cost: 12.7822 ms
   makeHessian cost: 8.30382 ms
-------After optimization, we got these parameters :
0.941867  2.09463 0.965551
-------ground truth:
1.0,  2.0,  1.0
```

第二题

$$\alpha_{b_ib_{k+1}} = \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}a\delta t^2$$

$$q_{b_ib_k+1} = q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}$$

$$\omega = \frac{1}{2}[(\omega^{b_k} - b_k^g) + (\omega^{b_{k+1}} - b_k^g)]$$

$$f_{15} = \frac{\partial \alpha_{b_ib_{k+1}}}{\partial \delta b_k^g} = \frac{1}{4}\frac{\partial q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -\frac{1}{2}\delta b_k^g \delta t \end{bmatrix} (a^{b_{k+1}} + n_{k+1}^a - b_k^a)\delta t^2}{\partial \delta b_k^g}$$

$$= \frac{1}{4}\frac{\partial R_{b_ib_{k+1}} \exp([-\delta b_k^g \delta t]_\times)(a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2}{\partial \delta b_k^g}$$

$$= \lim_{\delta b_k^g \to 0} \frac{1}{4}\frac{R_{b_ib_{k+1}}[-\delta b_k^g \delta t]_\times (a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2}{\delta b_k^g}$$

$$= \lim_{\delta b_k^g \to 0} \frac{1}{4}\frac{-R_{b_ib_{k+1}}[(a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2]_\times(-\delta b_k^g \delta t)}{\delta b_k^g}$$

$$=\text{-}\frac{1}{4}(R_{b_ib_{k+1}}[(a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2]_\times)(-\delta t)$$

$$=\text{-}\frac{1}{4}(R_{b_ib_{k+1}}[(a^{b_{k+1}} - b_k^a)]_\times \delta t^2)(-\delta t)$$

$$g_{12} = \frac{\partial \alpha_{b_ib_{k+1}}}{\partial n_k^g} = \frac{1}{4}\frac{\partial q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{4}n_k^g \delta t \end{bmatrix} (a^{b_{k+1}} + n_{k+1}^a - b_k^a)\delta t^2}{\partial n_k^g}$$

$$= \frac{1}{4}\frac{\partial R_{b_ib_{k+1}} \exp([\frac{1}{2}n_k^g \delta t]_\times)(a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2}{\partial n_k^g}$$

$$= \lim_{n_k^g \to 0} \frac{1}{4}\frac{R_{b_ib_{k+1}} \exp([\frac{1}{2}n_k^g \delta t]_\times)(a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2}{n_k^g}$$

$$= \lim_{n_k^g \to 0} \frac{1}{4}\frac{R_{b_ib_{k+1}}[\frac{1}{2}n_k^g \delta t]_\times (a^{b_{k+1}} + n_{k+1}^a - b_k^a)]\delta t^2}{n_k^g}$$

$$= \lim_{n_k^g \to 0} \frac{1}{4}\frac{-R_{b_ib_{k+1}}[(a^{b_{k+1}} + n_{k+1}^a - b_k^a)\delta t^2]_\times(\frac{1}{2}n_k^g \delta t)}{n_k^g}$$

$$=\text{-}\frac{1}{4}(R_{b_ib_{k+1}}[(a^{b_{k+1}} - b_k^a)]_\times \delta t^2)(\frac{1}{2}\delta t)$$

第三题：证明公式（9）： $\triangle x_{lm} = -\sum_{j=1}^{n} \frac{v_j^T F^{'T}}{\lambda_j + \mu} v_j$

证明：

$$(J^T J + \mu I)\Delta x_{lm} = F^{'}(x)$$

$$(V \Lambda V^T)\Delta x_{lm} = [V(\Lambda + \mu I)V^T]\Delta x_{lm} = -F^{'}(x)$$

$$\Delta x_{lm} = -\sum_{j=1}^{n} \frac{v_j^T F^{'T}}{\lambda_j + \mu} v_j$$