本次作业内容如下：

作业

1 样例代码给出了使用 LM 算法来估计曲线 $y = \exp(ax^2 + bx + c)$ 参数 a, b, c 的完整过程。
   1. 请绘制样例代码中 LM 阻尼因子 μ 随着迭代变化的曲线图
   2. 将曲线函数改成 $y = ax^2 + bx + c$，请修改样例代码中残差计算，雅克比计算等函数，完成曲线参数估计。
   3. 如果有实现其他阻尼因子更新策略可加分（选做）。

2 公式推导，根据课程知识，完成 $\mathbf{F}, \mathbf{G}$ 中如下两项的推导过程：

$$\mathbf{f}_{15} = \frac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \delta \mathbf{b}_k^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_k^a)]_\times \delta t^2)(-\delta t)$$

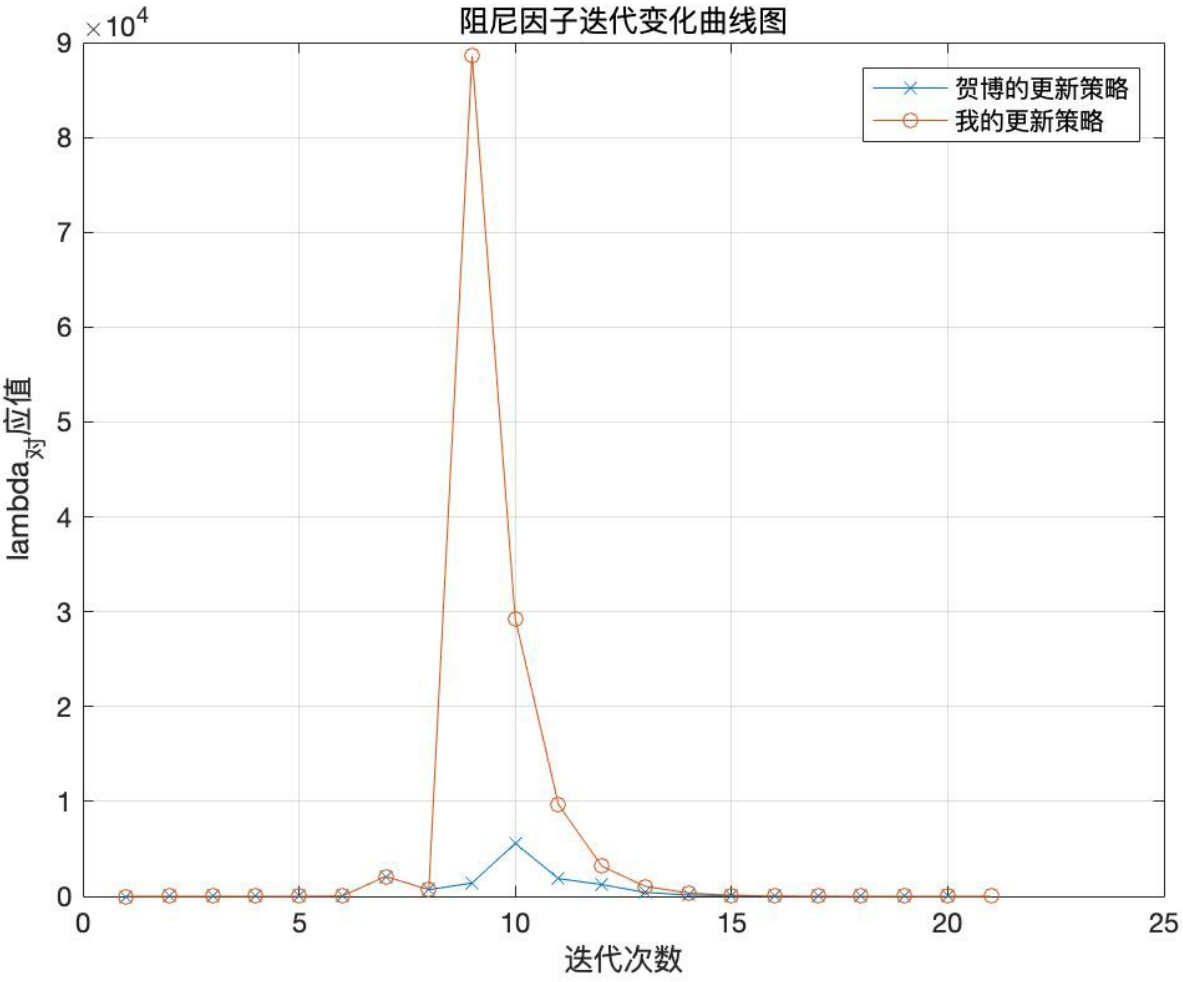
$$\mathbf{g}_{12} = \frac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \mathbf{n}_k^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_k} - \mathbf{b}_k^a)]_\times \delta t^2)(\frac{1}{2}\delta t)$$

3 证明式(9)。

一、编程题

1. 绘制阻尼因子的迭代变化曲线图

这里用我的更新策略和贺博的更新策略两种输出阻尼曲线图做对比：



分析：

可以看出，基本走向大家都差不多，我的可能会有点激进。导致多迭代一次。

使用贺博源代码中更新策略的迭代结果：

```
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: 22.9298 ms
   makeHessian cost: 15.725 ms
-------After optimization, we got these parameters :
0.941939  2.09453 0.965586
-------ground truth:
1.0,  2.0,  1.0
```

2. 修改残差计算和雅克比计算函数，完成曲线估计

核心代码：

```
// 计算曲线模型误差
virtual void ComputeResidual() override
{
    Vec3 abc = verticies_[0]->Parameters();  // 估计的参数
    residual_(0) = abc(0)*x_*x_ + abc(1)*x_ + abc(2)  - y_;  // 构建残差 预测值-测量值
}

// 计算残差对变量的雅克比
virtual void ComputeJacobians() override
{
    Vec3 abc = verticies_[0]->Parameters();
    // double exp_y =  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_ , x_ , 1; // 这里是分别对a, b, c求导生成的雅克比装进jaco_abc中
    jacobians_[0] = jaco_abc;
}

// 构造 N 次观测
   for (int i = 0; i < N; ++i) {
       double x = i/100.;
       double n = noise(generator);
       // 观测 y
       double y =  a*x*x + b*x + c  + n;
       //  double y = std::exp( a*x*x + b*x + c );

       // 每个观测对应的残差函数
       shared_ptr< CurveFittingEdge > edge(new CurveFittingEdge(x,y));
       std::vector<std::shared_ptr<Vertex>> edge_vertex;
       edge_vertex.push_back(vertex);
       edge->SetVertex(edge_vertex);

       // 把这个残差添加到最小二乘问题
       problem.AddEdge(edge);
   }
```

输出结果：

```
iter: 0 , chi= 719.475 , Lambda= 0.001
iter: 1 , chi= 91.395 , Lambda= 0.000333333
problem solve cost: 2.45328 ms
   makeHessian cost: 1.91108 ms
-------After optimization, we got these parameters :
 1.61039  1.61853 0.995178
-------ground truth:
1.0,  2.0,  1.0
```

分析：

两次迭代完成，chi已经是很小了，但是a的估计有些出入，LM用在这个函数拟合上有些大才小用

3. 实现其他阻尼因子更新策略

我用的更新策略原理：

1963 年 Marquardt 提出了一个如下的阻尼策略:

$$\begin{aligned} &\text{if } \rho < 0.25 \\ &\quad \mu := \mu * 2 \\ &\text{elseif } \rho > 0.75 \\ &\quad \mu := \mu / 3 \end{aligned} \tag{12}$$

核心代码：

```
if (rho > 0 && isfinite(tempChi))
{
    if (rho < 0.25){
        ni_ = 2;
        currentLambda_ *= ni_;
    }else if (rho > 0.75){
        currentLambda_ *= 0.33;
    }
    currentChi_ = tempChi;
    return true;
} else {
    currentLambda_ *= ni_;
    ni_ *= 2;
    return false;
}
```

使用我的更新策略的迭代结果：

```
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 30015.5 , Lambda= 692.06
iter: 2 , chi= 29217.7 , Lambda= 29232.6
iter: 3 , chi= 26227.2 , Lambda= 9646.76
iter: 4 , chi= 11290.5 , Lambda= 3183.43
iter: 5 , chi= 2229.93 , Lambda= 1050.53
iter: 6 , chi= 158.185 , Lambda= 346.676
iter: 7 , chi= 105.331 , Lambda= 114.403
iter: 8 , chi= 100.254 , Lambda= 37.753
iter: 9 , chi= 95.1615 , Lambda= 12.4585
iter: 10 , chi= 91.9953 , Lambda= 4.1113
iter: 11 , chi= 91.418 , Lambda= 1.35673
iter: 12 , chi= 91.396 , Lambda= 0.447721
iter: 13 , chi= 91.3959 , Lambda= 0.895441
   makeHessian cost: 12.3481 ms
-------After optimization, we got these parameters :
0.941887  2.09461  0.96556
-------ground truth:
```

```
1.0,  2.0,  1.0
```

分析：

1. 迭代次数比贺博版本多两次
2. 但是solve_time 时间要更加快
3. 最终输出的a,b,c参数基本没差别

二、分别证明两个误差传递雅各比f15和g12

$$f_{15} = \frac{\partial \alpha_{bi} b_{k+1}}{\partial \delta b_k^g}$$

首先：$w = \frac{1}{2}\cdot(w^{bk} + w^{bk+1}) - b_k^g$

f_{15}的分子部分：

$$\partial \alpha_{bi b_{k+1}} = \partial\left(\underset{①}{\alpha_{bibk}} + \underset{②}{\beta_{bibk}}\delta t + \underset{③}{\frac{1}{2}a}\delta t^2\right)$$

第①②部分与 b_k^g 无关，直接省略。

故分子部分 $\partial \alpha_{bib_{k+1}} = \partial\left(\frac{1}{2} a \delta t^2\right)$

其中 $a = \frac{1}{2}\left(q_{bibk}(a^{bk} - b_k^a) + q_{bibk+1}(a^{b_{k+1}} - b_k^a)\right)$

将a代入分子并化解：

$$\partial \beta_{bibk+1} = \partial \frac{1}{4} q_{bibk} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} w \delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -\frac{1}{2}\delta b_k^g \delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a)\delta t^2$$

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

$$= \frac{1}{4}\partial R_{bibk+1}(I+[-\delta b_k^g \delta t]_\times)(a^{bk+1}-b_k^a)\delta t^2$$

$$= -\frac{1}{4}\partial R_{bibk+1}\cdot[(a^{bk+1}-b_k^a)\delta t^2]_\times\cdot[-\delta b_k^g \delta t]$$

至此，我们开始求 f_{15}.

$$f_{15} = \frac{\partial R_{bibk+1}}{\delta b_k^g}$$

$$= -\frac{1}{4}R_{bibk+1}\cdot[(a^{bk+1}-b_k^a)\delta t^2]_\times[-\delta t]$$

$$= -\frac{1}{4}R_{bibk+1}[(a^{bk}-b_k^a)]_\times\cdot\delta t^2)\cdot[-\delta t]$$

证毕.

$$g_{12} = \frac{\partial v_{bibk+1}}{\partial n_a^g}$$

其中 n_k^g 是陀螺仪的k时刻测量白噪声.

g_{12} 和 f_{13} 的分子部分大同小异，可以写成如下形式

$$\partial \alpha_{b_i b_{k+1}} = \partial\left(\frac{1}{2} a \delta t^2\right)$$ 注其中a中包含的 α^{b_k} 部分与w无关，约去

$$= \partial\left(\frac{1}{2}\frac{1}{2} q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} w \delta t \end{bmatrix} (a^{k+1} - b_k^a)\right) \delta t^2$$

$$= \partial \frac{1}{4} q_{b_i b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{4} \delta n_k^g \end{bmatrix} (a^{k+1} - b_k^a) \delta t^2$$

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

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

结合分母推导：

$$g_{12} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta n_k^g} = -\frac{1}{4}\left(R_{b_i b_{k+1}} [a^k - b_k^a]_\times \delta t^2\right) \cdot \frac{1}{2} \delta t$$

证毕.

三、公式9推导

公式9的推导：

根据$(J^TJ+uI)\Delta x=-J^Tf$，其中半正定矩阵J^TJ做特征值分解为$\{\lambda_i\}$和对应的特征向量$\{v_i\}$. 它们符合公式：$J^TJ=V\Lambda V^T$

需证明：$\Delta x_{lm}=-\sum_{i=1}^{n}\frac{v_i^TF'^T}{\lambda_i+u}v_i$

首先：

$$(J^TJ+uI)\Delta x=-J^Tf=-F'(x)^T$$

$$\Rightarrow \Delta x=-(J^TJ+uI)^{-1}F'^T$$

$$= -(V\Lambda V^T + uI)^{-1} \cdot F'$$

$$= -V \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} V^T + uI)^{-1} \cdot F'^T$$

$$= -\left(V \begin{bmatrix} \lambda_1+u & & & \\ & \lambda_2+u & & \\ & & \ddots & \\ & & & \lambda_n+u \end{bmatrix} V^T\right)^{-1} \cdot F'^T$$

$$= -\sum_{j=1}^{n} \frac{V_j^T F'^T}{\lambda_j + u} V_j$$

注意：$V^T = V^{-1}$

证毕.