残差 Jacobian 的推导 IMU 预积分残差的雅克比

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. 实现其他更优秀的阻尼因子策略，并给出实验对比（选做，评优秀），策略可参考论文[a] 4.1.1 节。

习题 1：直接使用编译器编译示例代码可以得到以下结果：迭代了 12 次，最终 a=0.9419，b=2.0945，c=0.9656，与真实值 121 接近

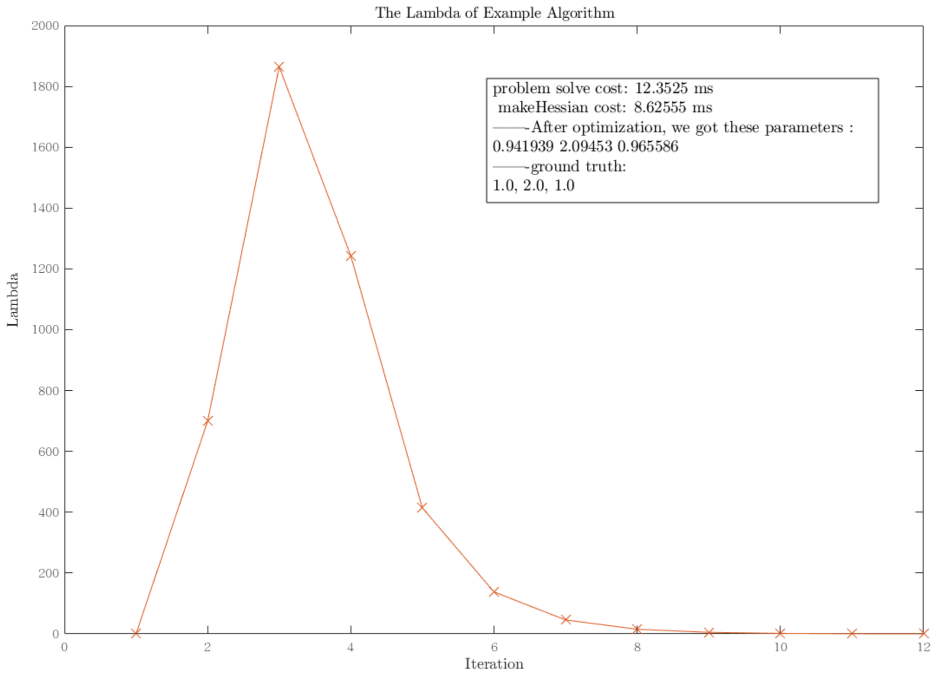
```
Terminal:   Local ×  +
dukeguo@dukeguo-OptiPlex-7060:~/文档/CurveFitting_LM/build/app$ ./testCurveFitting

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: 1.18513 ms
   makeHessian cost: 0.664714 ms
-------After optimization, we got these parameters :
0.941939  2.09453 0.965586
-------ground truth:
1.0,  2.0,  1.0
dukeguo@dukeguo-OptiPlex-7060:~/文档/CurveFitting_LM/build/app$
```

7: Structure 2: Favorites

6: TODO 9: Version Control Terminal 0: Messages CMake

Error running 'Build': Cannot start process, the working directory '/home/dukeguo/文档/CurveFitting_LM/cma

得到的 lambda 的分布如下图所示：



两次实验是不同时间做的，用时上有些区别。

2.把函数改成 y=a*x^2+b*x+c，修改 computeResidual 和 computeJacobian 函数，并将数据量提高到 1000，修正函数后的结果是更少的迭代数。

CurveFitting.cpp   problem.cc

jacobians_   Match Case   Words   Regex   2 matches

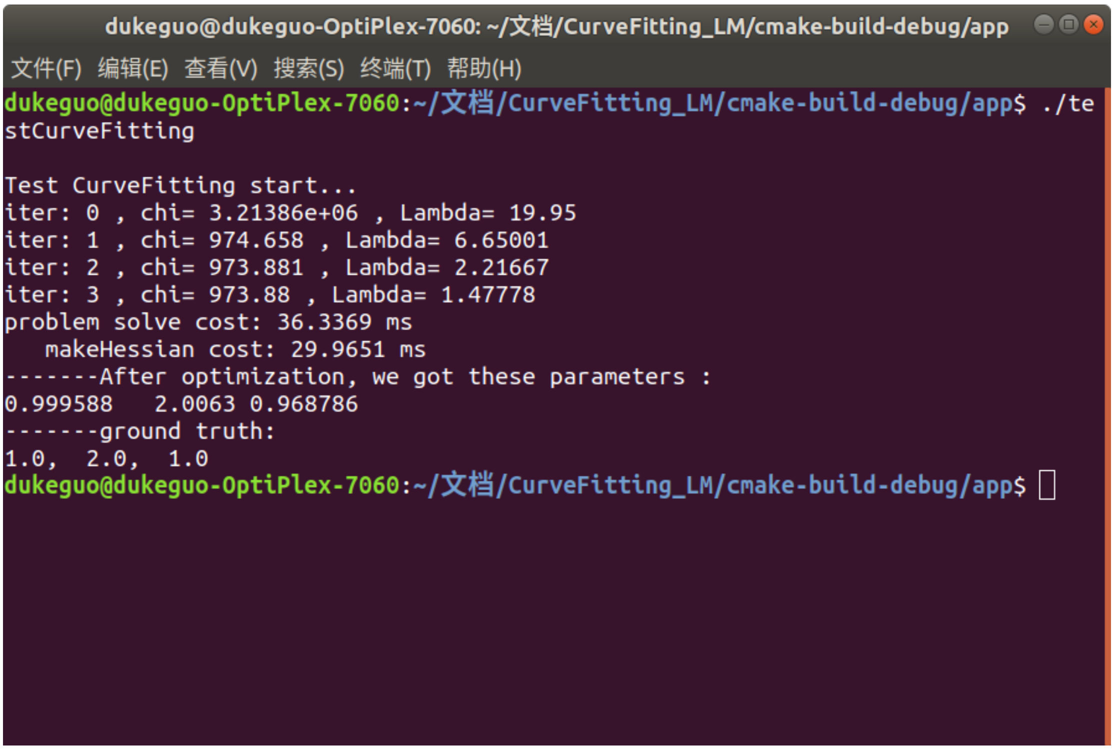
```
    // 计算残差对变量的雅克比
    virtual void ComputeJacobians() override
    {
        Vec3 abc = verticies_[0]->Parameters();
        double exp_y = std::exp( x: abc( index: 0)*x_*x_ + abc( index: 1)*x_ + abc( index: 2) );
        Eigen::Matrix<double, 1, 3> jaco_abc;  // 误差为1维，状态量 3 个，所以是 1x3 的雅克比矩阵
        jaco_abc << x_ * x_ * exp_y, x_ * exp_y , 1 * exp_y;
        jacobians_[0] = jaco_abc;

        // dukeguo 20191104
//        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 ;
//        jacobians_[0] = jaco_abc;

    }
    /// 返回边的类型信息
    virtual std::string TypeInfo() const override { return "CurveFittingEdge"; }
public:
    double x_,y_;  // x 值， y 值为 measurement
```

main

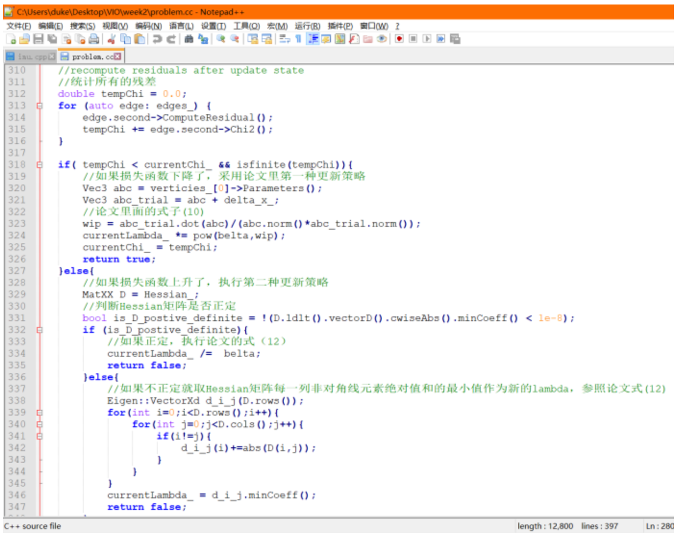
Run: testCurveFitting

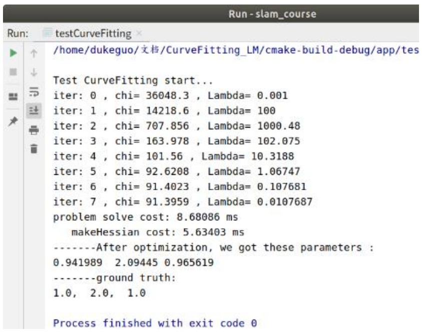
/home/dukeguo/文档/CurveFitting_LM/cmake-build-debug/app/te

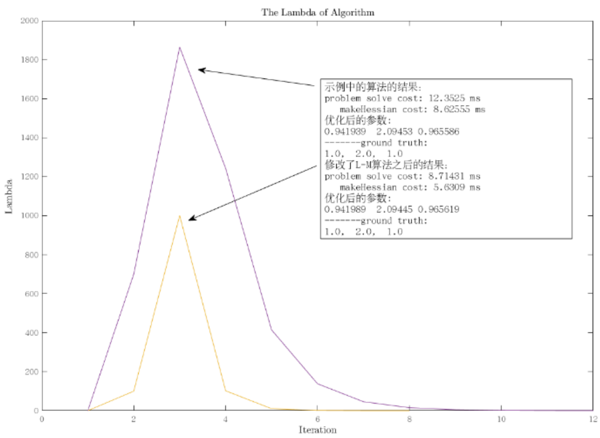
```
    // 计算曲线模型误差
    virtual void ComputeResidual() override
    {
        //示例代码
        Vec3 abc = verticies_[0]->Parameters();  // 估计的参数
        residual_( index: 0) = std::exp( x: abc( index: 0)*x_*x_ + abc( index: 1)*x_ + abc( index: 2) ) - y_;  // 构建残差

        // dukeguo 20191104
//        Vec3 abc = verticies_[0]->Parameters();  // 估计的参数
//        residual_(0) = abc(0) * x_ * x_ + abc(1) * x_ + abc(2) - y_; //y=ax2+bx+c 构建残差
    }
```



3.阅读了 LM 论文后，参考了一些资料，修改了 problem.cc 里面的 IsGoodStepInLM():







习题 2:

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:

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

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

$$\begin{aligned}
f_{15} &= \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta b_k^g} \\
&= \frac{\partial \frac{1}{4} q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega \delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -\frac{1}{2}\delta b_k^g \delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a))\delta t^2}{\partial \delta b_k^g} \\
&= \frac{1}{4} \frac{\partial R_{b_i b_{k+1}} \exp([-\delta b_k^g \delta t]_\times)(a_{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= \frac{1}{4} \frac{\partial R_{b_i b_{k+1}} (I + [-\delta b_k^g \delta t]_\times)(a_{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= -\frac{1}{4} \frac{\partial R_{b_i b_{k+1}} ([(a_{b_{k+1}} - b_k^a)\delta t^2]_\times)(-\delta b_k^g \delta t)}{\partial \delta b_k^g} \\
&= -\frac{1}{4} \left(R_{b_i b_{k+1}} [(a_{b_{k+1}} - b_k^a)]_\times \delta t^2\right) (-\delta t)
\end{aligned}$$

$$\begin{aligned}
g_{12} &= \frac{\partial \alpha_{b_i b_{k+1}}}{\partial n_k^g} \\
&= \frac{\partial \frac{1}{4} q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega \delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\frac{1}{2}\delta n_k^g \delta t) \end{bmatrix} (a^{b_{k+1}} - b_k^a))\delta t^2}{\partial \delta n_k^g} \\
&= \frac{1}{4} \frac{\partial R_{b_i b_{k+1}} \exp([\frac{1}{2}\delta n_k^g \delta t]_\times)(a_{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta n_k^g} \\
&= \frac{1}{4} \frac{\partial R_{b_i b_{k+1}} (I + [\frac{1}{2}\delta n_k^g \delta t]_\times)(a_{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta n_k^g} \\
&= -\frac{1}{4} \frac{\partial R_{b_i b_{k+1}} ([(a_{b_{k+1}} - b_k^a)\delta t^2]_\times)(\frac{1}{2}\delta n_k^g \delta t)}{\partial \delta n_k^g} \\
&= -\frac{1}{4} \left(R_{b_i b_{k+1}} [(a_{b_{k+1}} - b_k^a)]_\times \delta t^2\right) (\frac{1}{2}\delta t)
\end{aligned}$$

习题 3:

> 3 **证明式(9)。** $\Delta \mathbf{x}_{\mathrm{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j + \mu} \mathbf{v}_j$

No.

Date

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

∵ J^TJ 半正定. ∀ 非零向量 x 存在 $x^TJ^TJx \geq 0$ 恒成立.

$$(J^TJ + \mu I)\Delta x_{lm} = -J^Tf$$

令 $J^TJ+\mu I$ 为 A. 上式为 $A\Delta x_{lm} = -J^Tf$

$$\begin{aligned}\therefore x^TAx &= x^T(J^TJ+\mu I)x \\ &= x^TJ^TJx + x^T\mu I x \\ &= x^TJ^TJx + \mu x^Tx > 0\end{aligned}$$

∴ $x^TAx > 0$, A 正定, 可特征值分解, 对 J^TJ 进行特征值分解 $V\Lambda V^T$, 即 $\{\lambda_j\}$ 和 $\{v_j\}$ 分别是特征值和特征向量. 就有

$$J^TJv_j = \lambda_j v_j$$

$$\begin{aligned}Av_j &= (J^TJ+\mu I)v_j \\ &= J^TJv_j + \mu v_j \\ &= (\lambda_j + \mu)v_j\end{aligned}$$

∴ $(\lambda_j + \mu)$ 和 v_j 分别是 A 的特征值和特征向量

$$A = V\Sigma V^T$$

其中 $\Sigma = diag(\lambda_1+\mu, \lambda_2+\mu, \dots, \lambda_n+\mu)$

$$A\Delta x_{lm} = -J^Tf$$

$$V\Sigma V^T \Delta x_{lm} = -J^Tf$$

$$\Delta x_{lm} = -(V\Sigma V^T)^{-1}J^Tf$$

$$\Delta x_{lm} = -V\Sigma^{-1}V^TJ^Tf$$

其中 $\Sigma^{-1} = diag(\frac{1}{\lambda_1+\mu}, \frac{1}{\lambda_2+\mu}, \dots, \frac{1}{\lambda_n+\mu})$.

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