第三讲-习题

姜帆

2019 年 7 月 14 日

目录

1 LM 算法

1.1 阻尼因子变化曲线图

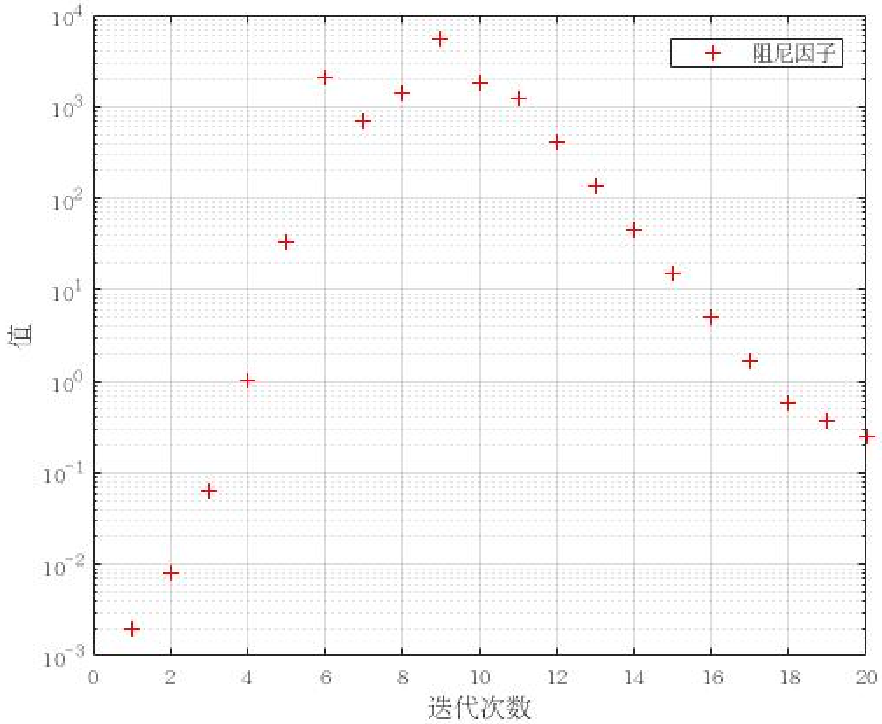


图 1: 阻尼因子变化曲线图

下图为程序运行结果:

```
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
iter: 12 , chi= 91.3959 , Lambda= 0.252554
problem solve cost: 19.8676 ms
   makeHessian cost: 13.7245 ms
-------After optimization, we got these parameters :
0.941842  2.09467 0.965537
-------ground truth:
1.0,  2.0,  1.0
save mu to txt!
```

图 2: 程序运行结果

1.2 更改曲线函数

将曲线函数改为 $y = ax^2 + bx + c$，修改对应雅克比计算函数，残差计算函数。另注意由于生成仿真数据时添加了均值为 0，方差为 1 的噪声项，噪声相对于数据较大，因此对增加仿真数据量。

$y = ax^2 + bx + c$ 函数对应的雅克比计算函数（导数）为 $y' = x^2 + x + 1$

```
class CurveFittingEdge: public Edge
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW
    CurveFittingEdge( double x, double y ): Edge(1,1, std::vector<std::string>{"abc"}) {
        x_ = x;
        y_ = y;
    }
    // 计算曲线模型误差
    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;
            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
    };
```

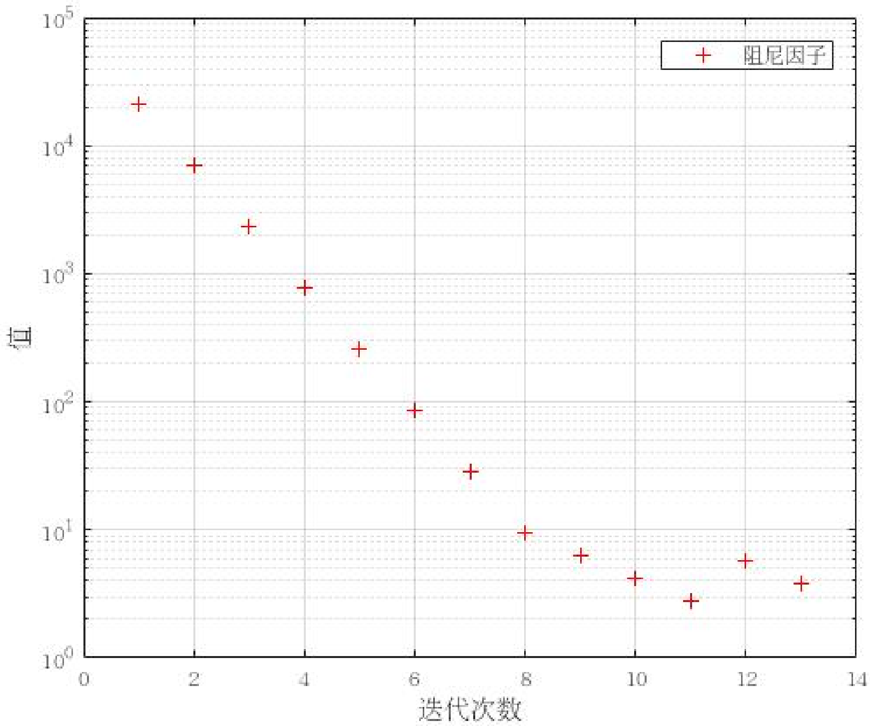


图 3: 阻尼因子变化曲线图

1.3 Marquardt 阻尼因子更新策略

Marquardt 阻尼因子更新策略如下:

$$if \rho < 0.25$$

$$\mu := \mu * 2$$

$$if \rho > 0.75$$

$$\mu := \mu / 3$$

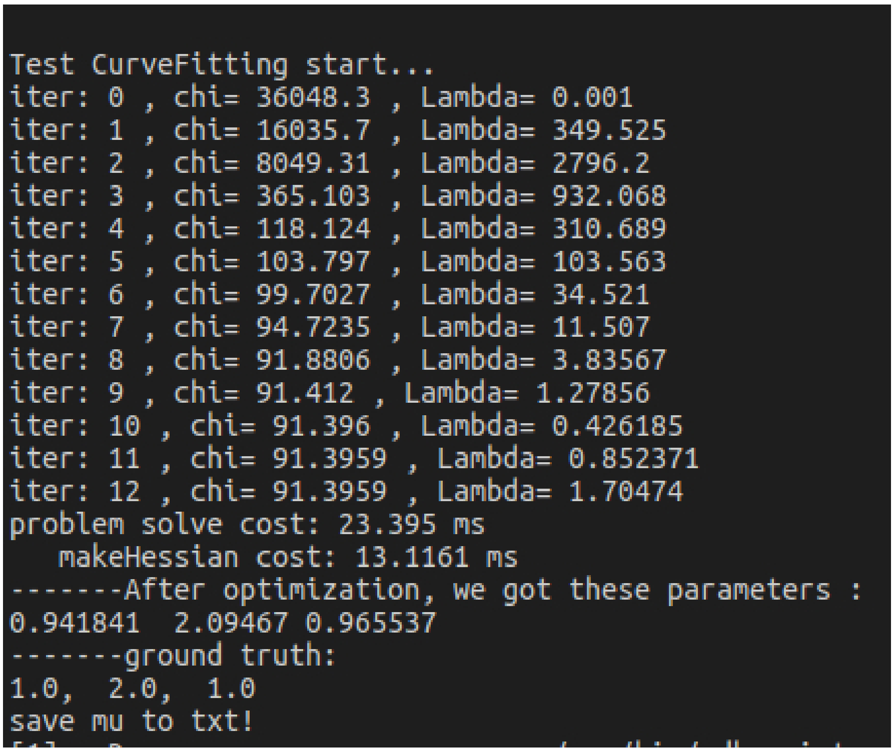
具体阻尼因子更新策略实现代码为:

```
    if(rho >= 0 && isfinite(tempChi))
    {
        if(rho < 0.25 )
            currentLambda_*=2;
        else if(rho >0.75)
            currentLambda_=currentLambda_/3;

        currentChi_ = tempChi;
        return true;
    }
    else
    {
      currentLambda_ =currentLambda_ *2;
```

```
        return false;
    }
```

下图为程序运行结果：



```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 16035.7 , Lambda= 349.525
iter: 2 , chi= 8049.31 , Lambda= 2796.2
iter: 3 , chi= 365.103 , Lambda= 932.068
iter: 4 , chi= 118.124 , Lambda= 310.689
iter: 5 , chi= 103.797 , Lambda= 103.563
iter: 6 , chi= 99.7027 , Lambda= 34.521
iter: 7 , chi= 94.7235 , Lambda= 11.507
iter: 8 , chi= 91.8806 , Lambda= 3.83567
iter: 9 , chi= 91.412 , Lambda= 1.27856
iter: 10 , chi= 91.396 , Lambda= 0.426185
iter: 11 , chi= 91.3959 , Lambda= 0.852371
iter: 12 , chi= 91.3959 , Lambda= 1.70474
problem solve cost: 23.395 ms
   makeHessian cost: 13.1161 ms
-------After optimization, we got these parameters :
0.941841  2.09467 0.965537
-------ground truth:
1.0,  2.0,  1.0
save mu to txt!
```

图 4: 程序运行结果

2 公式推导

2.1 f_{15}

f_{15} 求的是位移预积分量对 k 时刻角速度 b_k^g 的 Jacobian。

预积分的离散形式，其中积分方法采用中值积分，即两个相邻时刻 k 到 $k+1$ 的位姿是用两个时刻的测量值的平均值来计算。其中位移的预积分量为：

$$\begin{aligned}
&\alpha_{b_i b_{k+1}} = \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \\
&a = \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a)) \\
&\omega = \frac{1}{2}((\omega^{b_k} - b_k^g) + (\omega^{b_{k+1}} - b_k^g)) = \frac{1}{2}(\omega^{b_k} + \omega^{b_{k+1}}) - b_k^g
\end{aligned} \tag{1}$$

因此位移预积分量也可以写为:

$$
\begin{aligned}
\alpha_{b_i b_{k+1}} &= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \\
&= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{4}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a))\delta t^2 \\
&= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{4}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a))\delta t^2
\end{aligned} \tag{2}
$$

其中只有括号加号后面一项与角速度 b_k^g 有关，因此 f_{15} 可以变为:

$$
\begin{aligned}
f_{15} &= \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta b_k^g} \\
&= \frac{1}{4}\frac{\partial q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ -\frac{1}{2}\delta b_k^g \delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} exp([-\delta b_k^g \delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} (I + [-\delta b_k^g \delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} [-\delta b_k^g \delta t]_\times (a^{b_{k+1}} - b_k^a)\delta t^2}{\partial \delta b_k^g} \\
&= -\frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} [(a^{b_{k+1}} - b_k^a)]_\times \delta t^2 (-\delta b_k^g \delta t)}{\partial \delta b_k^g} \\
&= -\frac{1}{4} R_{b_1 b_{k+1}} [(a^{b_{k+1}} - b_k^a)]_\times \delta t^2 (-\delta t)
\end{aligned} \tag{3}
$$

2.2 g_{12}

f_{15} 求的是位移预积分量对 k 时刻角速度的噪声 $n_{b_k^g}$ 的 Jacobian。

将角速度测量噪声也考虑进模型, 预积分的离散形式，其中积分方法采用中值积分，即两个相邻时刻 k 到 $k+1$ 的位姿是用两个时刻的测量值的平均值来计算。其中位移的预积分量为:

$$
\begin{aligned}
&\alpha_{b_i b_{k+1}} = \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \\
&a = \frac{1}{2}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a)) \\
&\omega = \frac{1}{2}((\omega^{b_k} + n_k^g - b_k^g) + (\omega^{b_{k+1}} + n_{k+1}^g - b_k^g)) = \frac{1}{2}(\omega^{b_k} + n_k^g + \omega^{b_{k+1}} + n_{k+1}^g) - b_k^g
\end{aligned} \tag{4}
$$

因此位移预积分量也可以写为:

$$
\begin{aligned}
\alpha_{b_i b_{k+1}} &= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}a\delta t^2 \\
&= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{4}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_{k+1}}(a^{b_{k+1}} - b_k^a))\delta t^2 \\
&= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{4}(q_{b_i b_k}(a^{b_k} - b_k^a) + q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a))\delta t^2
\end{aligned} \tag{5}
$$

其中只有括号加号后面一项与角速度的噪声 $n_{b_k^g}$ 有关，因此 g_{12} 可以变为：

$$
\begin{aligned}
g_{12} &= \frac{\partial \alpha_{b_i b_k}}{\partial n_{b_k^g}} \\
&= \frac{1}{4}\frac{\partial q_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{4}\delta n_{b_k^g}\delta t \end{bmatrix} (a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_{b_k^g}} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} exp([\frac{1}{2}\delta n_{b_k^g}\delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_{b_k^g}} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} (I + [\frac{1}{2}\delta n_{b_k^g}\delta t]_\times)(a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_{b_k^g}} \\
&= \frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} [\frac{1}{2}\delta n_{b_k^g}\delta t]_\times (a^{b_{k+1}} - b_k^a)\delta t^2}{\partial n_{b_k^g}} \\
&= -\frac{1}{4}\frac{\partial R_{b_1 b_{k+1}} [(a^{b_{k+1}} - b_k^a)]_\times (\frac{1}{2}\delta n_{b_k^g}\delta t)\delta t^2}{\partial n_{b_k^g}} \\
&= -\frac{1}{4} R_{b_1 b_{k+1}} [(a^{b_{k+1}} - b_k^a)]_\times (\frac{1}{2}\delta t)\delta t^2
\end{aligned} \tag{6}
$$

3 证明

L-M 优化算法中，引入阻尼因子，如下式：

$$(J^T J + \mu I)\Delta x_{lm} = -J^T f \tag{7}$$

半正定的信息矩阵 $J^t J$ 特征值 λ_j 和对应的特征向量为 v_j。对 $J^T J$ 做特征值分解分解后有：$J^T J = V\Lambda V^T$。

$J^T J$ 为对称矩阵，对对称矩阵做特征值分解有 $J^T J = V\Lambda V^T$，其中 $V^T V = VV^T = I$。

有 $F' = (J^T f)^T$。

证明：

根据 $J^T J = V\Lambda V^T$ 和 $VV^T = I$，$(J^T J + \mu I)\Delta x_{lm} = -J^T$ 可以写为：

$$
\begin{aligned}
&(V\Lambda V^T + \mu I)\Delta x_{lm} = -F'^T \\
&(V(\Lambda + \mu I)V^T)\Delta x_{lm} = -F'^T
\end{aligned} \tag{8}
$$

等式两边同时左乘 V^T，右乘 V，则有：

$$
\begin{aligned}
&(\Lambda + \mu I)\Delta x_{lm} = -V^T F'^T V \\
&\Delta x_{lm} = -\frac{V^T F'^T V}{\Lambda + \mu I}
\end{aligned} \tag{9}
$$

其中：

$$V = \begin{bmatrix} v_1 & v_2 & ... & v_n \end{bmatrix}^T \tag{10}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & \text{\Large 0} & \\ & & \dots & & \\ & \text{\Large 0} & & \lambda_{n-1} & \\ & & & & \lambda_n \end{bmatrix} \tag{11}$$

$$\mu I = \begin{bmatrix} \mu & & & & \\ & \mu & & \text{\Large 0} & \\ & & \dots & & \\ & \text{\Large 0} & & \mu & \\ & & & & \mu \end{bmatrix} \tag{12}$$

因此变为:

$$\begin{aligned} \Delta x_{lm} &= -\frac{\begin{bmatrix} v_1 & v_2 & ... & v_n \end{bmatrix} F'^T \begin{bmatrix} v_1 \\ v_2 \\ ... \\ v_n \end{bmatrix}}{\begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & \text{\Large 0} & \\ & & \dots & & \\ & \text{\Large 0} & & \lambda_{n-1} & \\ & & & & \lambda_n \end{bmatrix} + \begin{bmatrix} \mu & & & & \\ & \mu & & \text{\Large 0} & \\ & & \dots & & \\ & \text{\Large 0} & & \mu & \\ & & & & \mu \end{bmatrix}} \\ &= -\sum_{i=1}^{n} \frac{v_j^T F'^T v_j}{\lambda_j + \mu} \end{aligned} \tag{13}$$