深蓝学院
shenlanxueyuan.com

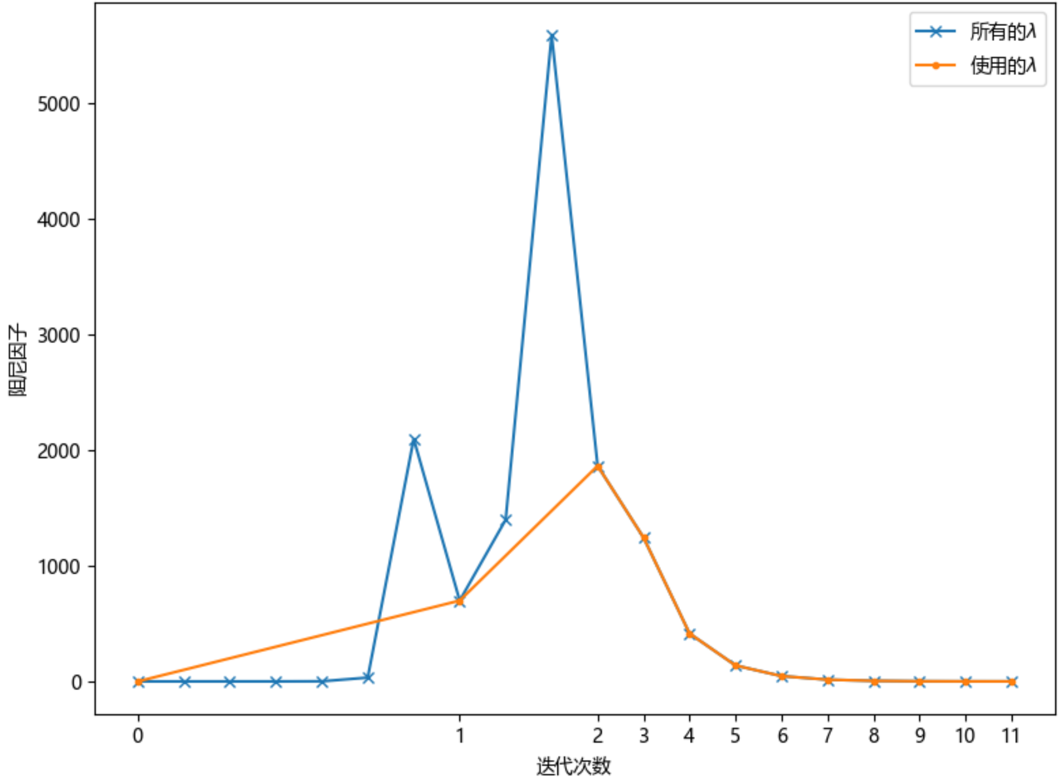
VIO第三章作业分享

主讲人 汤火火火

纲要

- 第一题：LM算法
- 第二题：公式推导
- 第三题：式（9）证明

（1）绘制阻尼因子变化曲线



在第（1）题中，阻尼因子λ的变化存在两种情况，一种是**接受本次迭代时有效的λ**，另一种是**拒绝迭代时无效的λ**。

左图中，蓝色折线表示了整个迭代过程中所有的λ的变化，黄色折线表示了正确迭代时λ的变化。

（1）绘制阻尼因子变化曲线

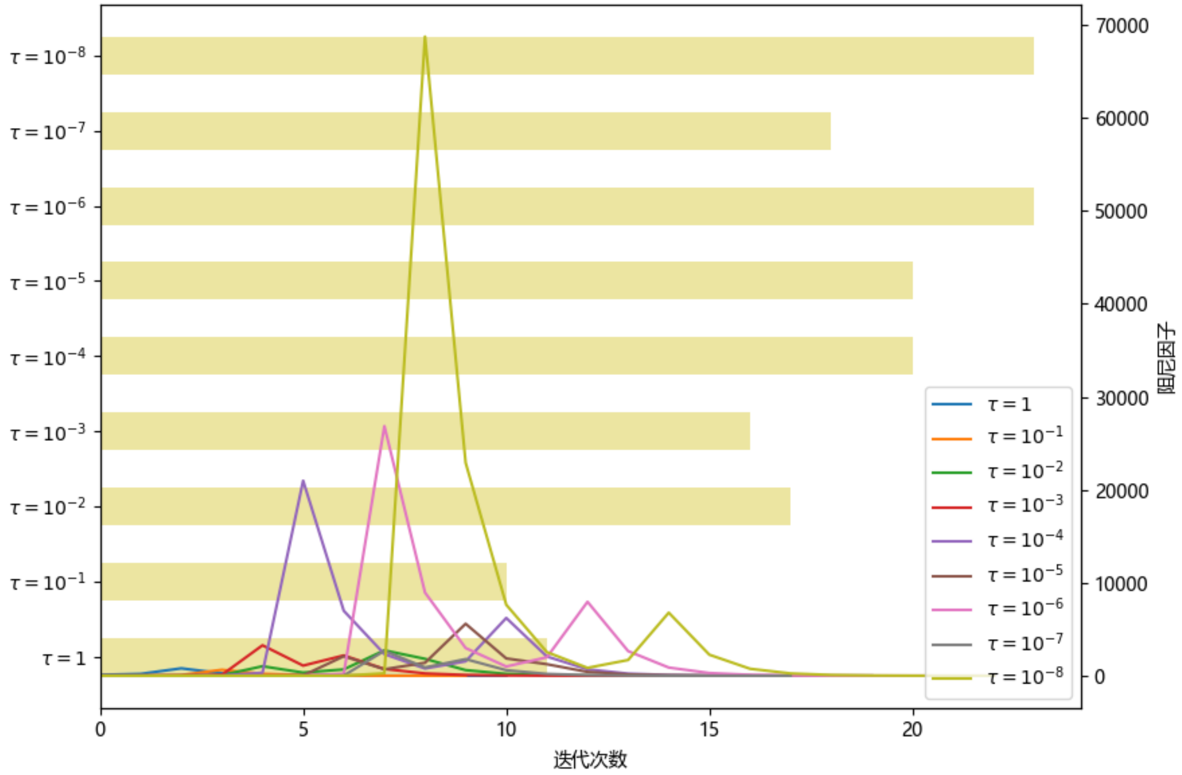
深蓝学院
shenlanxueyuan.com

阻尼因子初值的选取策略：

$$\mu_0 = \tau \cdot \max\{(\mathbf{J}^\top\mathbf{J})_{ii}\}$$

其中：τ~[10^{-8}，1]

在示例代码中，τ的取值为10^{-5}，在论文[1]中τ的取值为10^{-2}。



[1] H. P. Gavin, “The Levenberg-Marquardt algorithm for nonlinear least squares curve-Fitting problems”, 2020.

（2）二次函数曲线参数估计

函数模型改为二次函数：

$$y = ax^2 + bx + c$$

对应的残差和雅可比计算及代码：

$$e_i = f_i(\hat{\mathbf{x}}) - \mathbf{y}_i \qquad \mathbf{J}_i = \frac{\partial f_i(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} x_i^2 & x_i & 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

```
virtual void ComputeResidual() override {
    Vec3 abc = verticies_[0]->Parameters();
    residual_(0) = abc(0) * x_ * x_ + abc(1) * x_ + abc(2) - y_;
}
```

```
virtual void ComputeJacobians() override {
    Eigen::Matrix<double, 1, 3> jaco_abc;
    jaco_abc << x_ * x_, x_, 1;
    jacobians_[0] = jaco_abc;
}
```

（2）二次函数曲线参数估计

如果不修改其他部分代码，直接运行的话，会得到下面的结果：

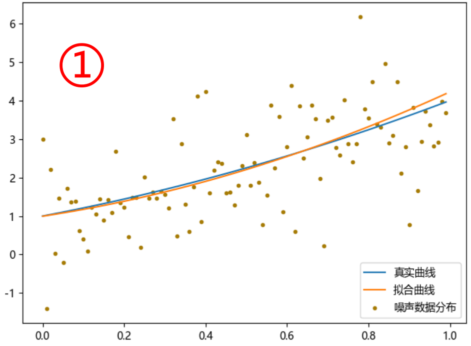
```
➜ app "/home/tang/data/手写vio第七期/第三章：基于优化的IMU与视觉信息融合/CurveFitting_LM/build/app/testCurveFitting"

Test CurveFitting start...
iter: 0 , chi= 719.475 , Lambda= 0.001
iter: 1 , chi= 91.395 , Lambda= 0.000333333
problem solve cost: 4.40725 ms
   makeHessian cost: 3.50658 ms
-------After optimization, we got these parameters :
 1.61039  1.61853 0.995178
-------ground truth:
1.0,  2.0,  1.0
```

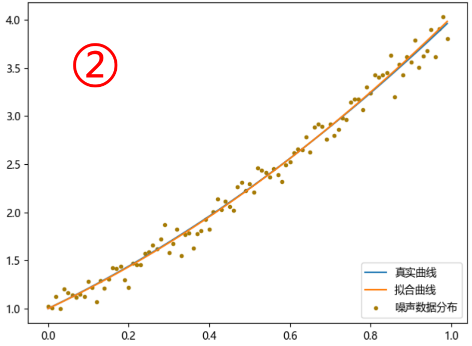
我认为出现这种结果的主要原因是抽样出来的数据不足以反应真实的母体分布。

（2）二次函数曲线参数估计

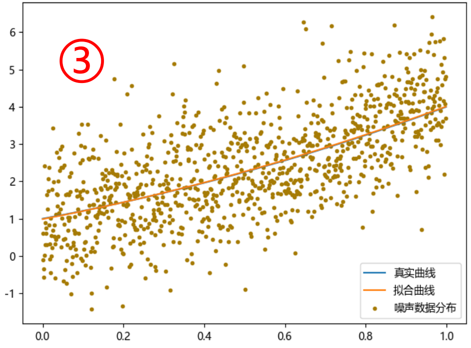
$\sigma^2 = 1, N = 100, [0-1]$



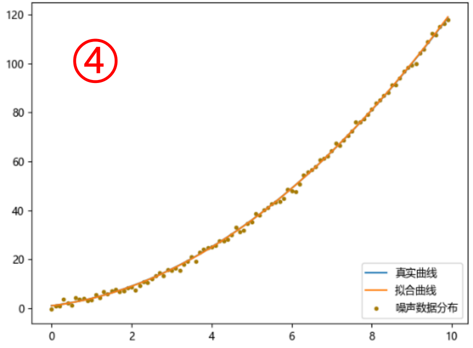
$\sigma^2 = 0.1, N = 100, [0-1]$



$\sigma^2 = 1, N = 1000, [0-1]$



$\sigma^2 = 1, N = 100, [0-10]$



三种比较好的改进方式：

② 减少噪声的方差

③ 增加多余观测

④ 增大数据范围

拟合曲线参数：

① $a = 1.61029, b = 1.61877, c = 0.995099$

② $a = 1.06114, b = 1.96177, c = 0.999529$

③ $a = 0.959478, b = 2.06226, c = 0.988924$

④ $a = 1.0061, b = 1.96188, c = 0.995075$

（3）其他阻尼因子更新策略

在论文[1]的4.1.1中，共介绍了三种阻尼因子的更新策略，示例代码中使用的是其中的第三种（Nielsen策略），下面介绍前两种：

此外，两种策略在阻尼因子初值选择和参数估值的更新方式上都是不一样的。

第一种策略（Marquardt）：

$$
\begin{aligned}
&\text{If } \rho > 0 \\
&\qquad \lambda = \max\{\lambda / L_{\downarrow}, 10^{-7}\} \\
&\text{Else} \\
&\qquad \lambda = \min\{\lambda L_{\uparrow}, 10^{7}\}
\end{aligned}
$$

其中，$L_{\downarrow}$为λ的降低因子，$L_{\uparrow}$为λ的提升因子。在论文的代码实现中，$L_{\downarrow}$取默认值为9，$L_{\uparrow}$取默认值为11。

[1] H. P. Gavin, “The Levenberg-Marquardt algorithm for nonlinear least squares curve-Fitting problems”, 2020.

（3）其他阻尼因子更新策略

第二种策略（Quadratic）：

$$\alpha=\mathbf{b}^{\mathrm{T}}\Delta\mathbf{x}/\{[F(\mathbf{x}+\Delta\mathbf{x})-F(\mathbf{x})]/2+2\mathbf{b}^{\mathrm{T}}\Delta\mathbf{x}\}$$

If $\rho(\alpha\Delta\mathbf{x})>0$

$$\lambda=\max\{\lambda/(1+\alpha),10^{-7}\}$$

Else

$$\lambda=\lambda+|F(\mathbf{x}+\alpha\Delta\mathbf{x})-F(\mathbf{x})|/2\alpha$$

(3) 其他阻尼因子更新策略

三种阻尼因子效果对比：

首先，第二种阻尼因子更新策略**对初值的选取要求较高**。如果初值相差真值较大时，会出现第一次计算改正数Δx使整体的误差增加较多的情况。这种情况下会导致计算α**接近0**，导致后面的更新规则失效，陷入死循环。

第一种更新策略

```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.01
iter: 1 , chi= 34603.6 , Lambda= 16.2678
iter: 2 , chi= 5202.26 , Lambda= 1.80753
iter: 3 , chi= 737.707 , Lambda= 0.200837
iter: 4 , chi= 355.065 , Lambda= 0.0223152
iter: 5 , chi= 141.356 , Lambda= 0.00247947
iter: 6 , chi= 100.515 , Lambda= 0.000275496
iter: 7 , chi= 92.175 , Lambda= 3.06107e-05
iter: 8 , chi= 91.3988 , Lambda= 3.40119e-06
iter: 9 , chi= 91.3959 , Lambda= 1e-06
problem solve cost: 25.5488 ms
   makeHessian cost: 18.5645 ms
-------After optimization, we got these parameters :
0.941934  2.09453 0.965589
-------ground truth:
1.0,  2.0,  1.0
```

第三种更新策略

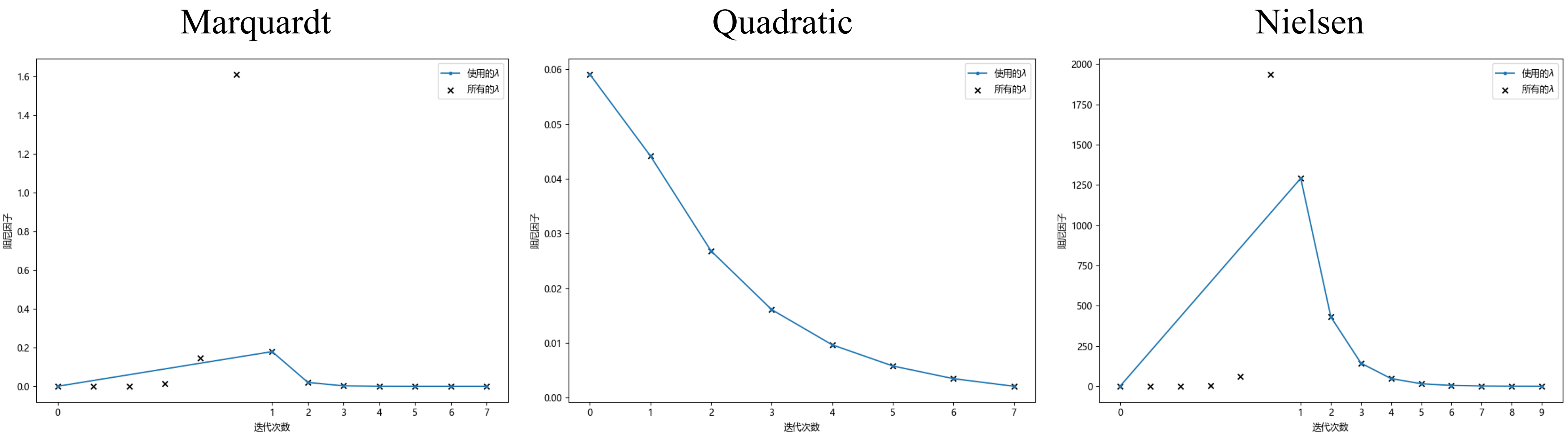
```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 1
iter: 1 , chi= 15144.6 , Lambda= 341.333
iter: 2 , chi= 6912.26 , Lambda= 1820.44
iter: 3 , chi= 294.547 , Lambda= 606.815
iter: 4 , chi= 110.326 , Lambda= 202.272
iter: 5 , chi= 102.297 , Lambda= 67.4239
iter: 6 , chi= 97.7436 , Lambda= 22.4746
iter: 7 , chi= 93.2127 , Lambda= 7.49154
iter: 8 , chi= 91.5522 , Lambda= 2.49718
iter: 9 , chi= 91.3985 , Lambda= 0.832393
iter: 10 , chi= 91.3959 , Lambda= 0.554929
problem solve cost: 27.1664 ms
   makeHessian cost: 19.2018 ms
-------After optimization, we got these parameters :
0.942182  2.09417 0.965707
-------ground truth:
1.0,  2.0,  1.0
```

第二种更新策略

```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 1
problem solve cost: 11.9454 ms
   makeHessian cost: 2.11053 ms
-------After optimization, we got these parameters :
0 0 0
-------ground truth:
1.0,  2.0,  1.0
```

（3）其他阻尼因子更新策略

为了比较三种不同更新策略情况下，阻尼因子的变化情况，选取了与真值较为接近的初值：$a = 0.8$，$b = 1.4$，$c = 0.7$。（也有做法是在代码中为α设定一个下限，如：0.1）



纲要

- 第一题：LM算法
- 第二题：公式推导
- 第三题：式（9）证明

位移的预积分对k时刻角速度bias求导

深蓝学院 shenlanxueyuan.com

（1）位移的预积分对k时刻角速度bias的Jacobian： $\mathbf{f}_{15} = \dfrac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \delta \mathbf{b}_k^g}$

由预积分计算的递推公式:

$$
\begin{aligned}
\boldsymbol{\alpha}_{b_i b_{k+1}} &= \boldsymbol{\alpha}_{b_i b_{k+1}} + \boldsymbol{\beta}_{b_i b_{k+1}} \delta t + \frac{1}{4}\left(\mathbf{q}_{b_i b_k}\left(\mathbf{a}^{b_k} - \mathbf{b}_k^a\right) + \mathbf{q}_{b_i b_{k+1}}\left(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a\right)\right)\delta t^2 \\
&= \boldsymbol{\alpha}_{b_i b_{k+1}} + \boldsymbol{\beta}_{b_i b_{k+1}} \delta t + \frac{1}{4}\left(\mathbf{q}_{b_i b_k}\left(\mathbf{a}^{b_k} - \mathbf{b}_k^a\right) + \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega \delta t \end{bmatrix}\left(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a\right)\right)\delta t^2
\end{aligned}
$$

上式中仅红色部分与**角速度bias**有关。并且:

$$
\boldsymbol{\omega} = \frac{1}{2}\left(\left(\boldsymbol{\omega}^{b_k} - \mathbf{b}_k^g\right) + \left(\boldsymbol{\omega}^{b_{k+1}} - \mathbf{b}_k^g\right)\right) = \frac{1}{2}\left(\boldsymbol{\omega}^{b_k} + \boldsymbol{\omega}^{b_{k+1}}\right) - \mathbf{b}_k^g
$$

位移的预积分对k时刻角速度bias求导

$$\mathbf{f}_{15}=\frac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \delta \mathbf{b}_k^g}$$

$$=\frac{\partial \frac{1}{4}\mathbf{q}_{b_i b_k}\otimes\begin{bmatrix}1\\ \frac{1}{2}(\boldsymbol{\omega}-\delta\mathbf{b}_k^g)\delta t\end{bmatrix}\left(a^{b_{k+1}}-b_k^a\right)\delta t^2}{\partial\delta\mathbf{b}_k^g}$$

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

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

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

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

位移的预积分对k时刻角速度白噪声求导

深蓝学院
shenlanxueyuan.com

（2）位移的预积分对k时刻角速度白噪声的Jacobian： $\mathbf{g}_{15}=\dfrac{\partial \boldsymbol{\alpha}_{b_i b_{k+1}}}{\partial \mathbf{n}_k^g}$

由预积分计算的递推公式：

$$
\begin{aligned}
\boldsymbol{\alpha}_{b_i b_{k+1}} &= \boldsymbol{\alpha}_{b_i b_{k+1}} + \boldsymbol{\beta}_{b_i b_{k+1}}\delta t + \frac{1}{4}\left(\mathbf{q}_{b_i b_k}\left(\mathbf{a}^{b_k}-\mathbf{b}_k^a\right)+\mathbf{q}_{b_i b_{k+1}}\left(\mathbf{a}^{b_{k+1}}-\mathbf{b}_k^a\right)\right)\delta t^2 \\
&= \boldsymbol{\alpha}_{b_i b_{k+1}} + \boldsymbol{\beta}_{b_i b_{k+1}}\delta t + \frac{1}{4}\left(\mathbf{q}_{b_i b_k}\left(\mathbf{a}^{b_k}-\mathbf{b}_k^a\right)+\mathbf{q}_{b_i b_k}\otimes\begin{bmatrix}1\\ \frac{1}{2}\omega\delta t\end{bmatrix}\left(\mathbf{a}^{b_{k+1}}-\mathbf{b}_k^a\right)\right)\delta t^2
\end{aligned}
$$

上式中仅红色部分与**角速度白噪声**有关。并且：

$$
\begin{aligned}
\boldsymbol{\omega} &= \frac{1}{2}\left(\left(\left(\boldsymbol{\omega}^{b_k}+\mathbf{n}_k^g\right)-\mathbf{b}_k^g\right)+\left(\left(\boldsymbol{\omega}^{b_{k+1}}+\mathbf{n}_{k+1}^g\right)-\mathbf{b}_k^g\right)\right) \\
&= \frac{1}{2}\left(\boldsymbol{\omega}^{b_k}+\boldsymbol{\omega}^{b_{k+1}}\right)-\mathbf{b}_k^g+\frac{1}{2}\mathbf{n}_{k+1}^g+\frac{1}{2}\mathbf{n}_k^g
\end{aligned}
$$

位移的预积分对k时刻角速度白噪声求导

深蓝学院 shenlanxueyuan.com

这里的推导过程和公式推导（1）一样，需要注意的就是k时刻和$k+1$时刻的白噪声不同，所以白噪声相较于bias在前面多了个$\frac{1}{2}$。

$$\mathbf{g}_{12}=\frac{\partial \boldsymbol{\alpha}_{b_ib_{k+1}}}{\partial \mathbf{n}_k^g}$$

$$=\frac{\partial \frac{1}{4}\mathbf{q}_{b_ib_k}\otimes\begin{bmatrix}1\\ \frac{1}{2}\left(\boldsymbol{\omega}+\frac{1}{2}\mathbf{n}_k^g\right)\delta t\end{bmatrix}\left(\mathbf{a}^{b_{k+1}}-\mathbf{b}_k^a\right)\delta t^2}{\partial \mathbf{n}_k^g}$$

$$=\frac{1}{4}\frac{\partial \mathbf{R}_{b_ib_k}\exp\left(\left[\left(\boldsymbol{\omega}+\frac{1}{2}\mathbf{n}_k^g\right)\delta t\right]_\times\right)\left(a^{b_{k+1}}-b_k^a\right)\delta t^2}{\partial \mathbf{n}_k^g}$$

$$=\frac{1}{4}\frac{\partial \mathbf{R}_{b_ib_k}\exp([\boldsymbol{\omega}\delta t]_\times)\exp\left(\left[J_r(\boldsymbol{\omega}\delta t)\frac{1}{2}\mathbf{n}_k^g\delta t\right]_\times\right)\left(a^{b_{k+1}}-b_k^a\right)\delta t^2}{\partial \mathbf{n}_k^g}$$

$$=\frac{1}{4}\frac{\partial -\mathbf{R}_{b_ib_{k+1}}\left[\left(a^{b_{k+1}}-b_k^a\right)\delta t^2\right]_\times\left(J_r(\boldsymbol{\omega}\delta t)\frac{1}{2}\mathbf{n}_k^g\delta t\right)}{\partial \mathbf{n}_k^g}$$

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

纲要

- 第一题：LM算法
- 第二题：公式推导
- 第三题：式（9）证明

式（9）证明

深蓝学院
shenlanxueyuan.com

证明： $$\Delta\mathbf{x}_{lm} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^T \mathbf{F}'^T}{\lambda_j + \mu} \mathbf{v}_j$$

由Levenberg-Marquardt方法：

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\Delta\mathbf{x}_{lm} = -(\mathbf{J}^T\mathbf{f})^T = -\mathbf{F}'^T$$

进行特征值分解： $$\Rightarrow (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T + \mu\mathbf{I})\Delta\mathbf{x}_{lm} = -\mathbf{F}'^T$$

其中：$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_j]$，$\mathbf{\Lambda} = \mathrm{diag}([\lambda_1, \lambda_2, \dots, \lambda_j])$。

式（9）证明

由于$\mathbf{J}^T\mathbf{J}$为对称阵且**对称阵不同的特征值对应的特征向量相互正交**，即$\mathbf{V}\mathbf{V}^T=\mathbf{I}$

$$\Rightarrow \mathbf{V}(\mathbf{\Lambda}+\mu\mathbf{I})\mathbf{V}^T\Delta\mathbf{x}_{lm}=-\mathbf{F}'^T$$

$$\Rightarrow \Delta\mathbf{x}_{lm}=-\mathbf{V}(\mathbf{\Lambda}+\mu\mathbf{I})^{-1}\mathbf{V}^T\mathbf{F}'^T$$

上式$\mathbf{V}(\mathbf{\Lambda}+\mu\mathbf{I})^{-\mathbf{1}}\mathbf{V}^T$部分展开，写成相加的形式可以得到：

$$\Rightarrow \Delta\mathbf{x}_{lm}=-\left[\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_j\right](\mathbf{\Lambda}+\mu\mathbf{I})^{-1}\begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \vdots\\ \mathbf{v}_j^T\end{bmatrix}\mathbf{F}'^T$$

$$\Rightarrow \Delta\mathrm{x}_{lm}=-\sum_{j=1}^{n}\frac{\mathrm{v}_j^T\mathrm{F}'^T}{\lambda_j+\mu}\mathrm{v}_j$$

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening