从零手写 VIO 作业　第三周

purebob

July 12, 2019

1 编程实践题

1.1 μ 曲线图

示例代码运行截图:

```
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: 27.5091 ms
   makeHessian cost: 19.6693 ms
-------After optimization, we got these parameters :
0.941939  2.09453 0.965586
-------ground truth:
1.0,  2.0,  1.0
```

图 1: 示例代码运行结果

绘制出阻尼因子 μ 随迭代次数的曲线图:

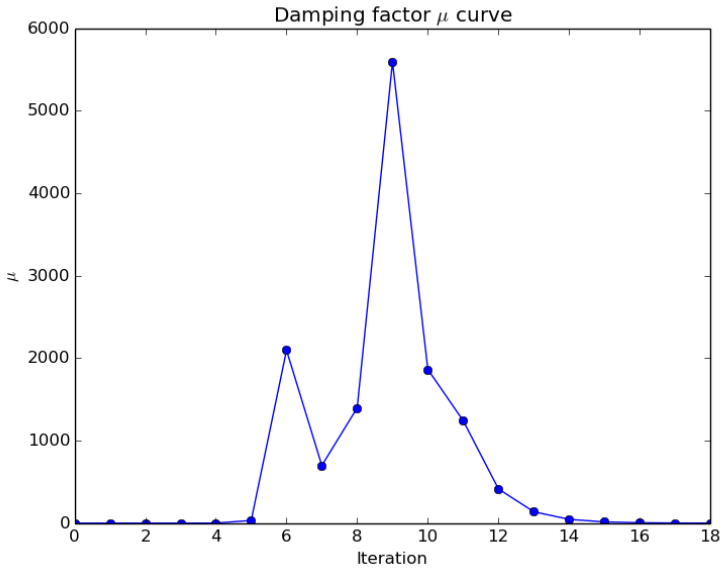

图 2: μ 曲线图

1.2 $y = ax^2 + bx + c$

当曲线函数为 $y = ax^2 + bx + c$, 其对于参数 a, b, c 的雅克比为 $\begin{bmatrix} x^2 & x & 1 \end{bmatrix}$, 修改样例代码中的 ComputeJacobians 和 ComputeResidual 函数:

```
1    virtual void ComputeResidual() override
2    {
3        Vec3 abc = verticies_[0]->Parameters();  // 估计的参数
4        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();

        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;
    }
```

运行代码得到结果: 收敛比较快，三次迭代就得到了结果。

```
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: 81.38 ms
   makeHessian cost: 66.025 ms
-------After optimization, we got these parameters :
0.958923  2.06283 0.968821
-------ground truth:
1.0,  2.0,  1.0
```

图 3: 示例代码运行结果

1.3 其他阻尼因子更新策略

在论文 A NEW DAMPING STRATEGY OF LEVENBERG-MARQUARDT ALGORITHM FOR MULTILAYER PERCEPTRONS http://www.nnw.cz/doi/2011/NNW.2011.21.020.pdf中作者提出了一种 LM 阻尼因子的跟新方法。主要思想是在迭代初期使用新旧预估状态量的内积来更新阻尼因子使其更接近最速下降或高斯牛顿；在损失函数上升时先判断海森矩阵是否正定，正定的话就使用预定的 $\beta(0.1)$ 来更新阻尼因子；若不正定，构建一个正定的对角占优（diagonally dominant）矩阵，使用其主对角线元素的最小值作为新的阻尼因子来保证迭代的下降方向尽量贴合目标状态量的方向。由于时间紧急，我简短的说了一下这篇论文的实现策略。代码的实现在附件的 strategy.cpp 里，运行结果如图四所示。

相对于示例代码中所用更新策略，新的策略需要的迭代次数只有 7 次，达到的精度却几乎一致，说明这篇论文的更新策略在本例上的收敛速度是快于原先策略的。论文见附件的 NNW.2011.21.020.pdf。

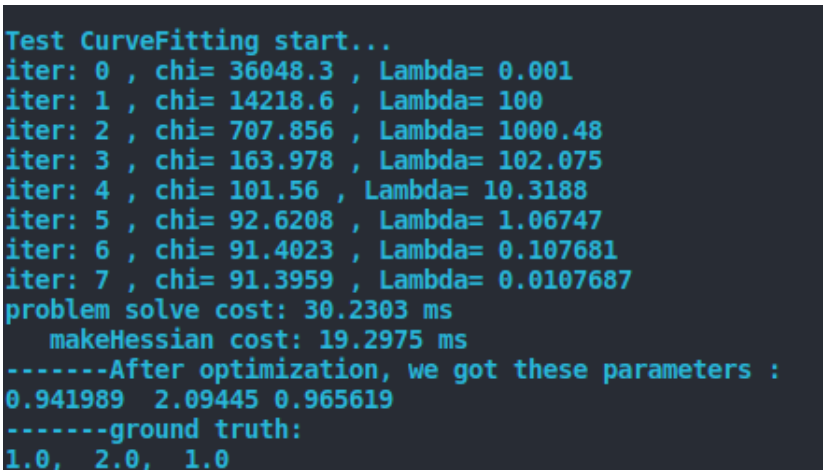

图 4: 新阻尼因子更新策略运行结果

2 推导 F,G 两项

注: 证明过程中的求导同样遵循课件里的简约求导书写方式.

2.1 f_{15}

f_{15} 即

$$\frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta b_k^g}$$

误差传递公式为:

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

$\alpha_{b_i b_{k+1}}$ 的前两项并不受角速度 bias 影响, 因此雅克比推导只考虑最后一部分，应用李代数映射的伴随公式:

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

2.2 g_{12}

g_{12} 即

$$\frac{\partial \alpha_{b_i b_{k+1}}}{\partial n_k^g}$$

类似于 f_{15},g_{12} 有如下递推 (这里 imu 的 g 噪声项前面取加号):

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

n_k^g 对 $\alpha_{b_i b_{k+1}}$ 的误差影响在于 $\begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}$ 上. 对这一部分进行求导 (为了方便起见,δn_k^g 直接写在了分母上) 得:

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

3 证明式 (9)

矩阵 $J^T J$ 半正定, 即对于任意非零向量 x 有 $x^T J^T J x \geq 0$ 恒成立. 对于方程:

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

令 $(J^T J + \mu I)$ 为 A，上式变为

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

. 对于任意非零向量 x，进行下式计算:

$$
\begin{aligned}
x^T A x &= x^T (J^T J + \mu I) x \\
&= x^T J^T J x + x^T \mu I x \\
&= x^T J^T J x + \mu x^T x > 0
\end{aligned}
$$

这说明对于任意非零向量 x，$x^T A x > 0$ 成立，矩阵 A 正定, 可以对 A 进行特征值分解. 由于 $J^T J$ 可进行特征值分解为 $V \Lambda V^T$，即 $\{\lambda_j\}$ 和 $\{v_j\}$ 分别为对应的特征值与特征向量. 有下式成立:

$$J^T J v_j = \lambda_j v_j$$

计算下式:

$$
\begin{aligned}
A v_j &= (J^T J + \mu I) v_j \\
&= J^T J v_j + \mu I v_j \\
&= (\lambda_j + \mu) v_j
\end{aligned}
$$

即可得出 $\{v_j\}$ 仍然是 A 的特征向量，A 的特征值则变为了 $\{\lambda_j + \mu\}$. 对 A 进行特征值分解:

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

其中 $\Sigma = \text{diag}(\lambda_1 + \mu, \lambda_2 + \mu, ..., \lambda_n + \mu)$. 原方程求解:

$$
\begin{aligned}
A\Delta x_{lm} &= -J^T f \\
V\Sigma V^T \Delta x_{lm} &= -J^T f \\
\Delta x_{lm} &= -(V\Sigma V^T)^{-1} J^T f \\
\Delta x_{lm} &= -V\Sigma^{-1} V^T J^T f
\end{aligned}
$$

Σ^{-1} 为 $\mathrm{diag}(\frac{1}{\lambda_1+\mu}, \frac{1}{\lambda_2+\mu}, ..., \frac{1}{\lambda_n+\mu})$，对 $-V\Sigma^{-1}V^TJ^Tf$ 进行展开即可得出：

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

式 (9) 得证.