从零开始写 vio

kohill

November 2019

1 样例代码修改

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

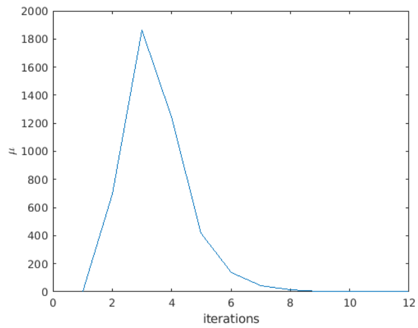


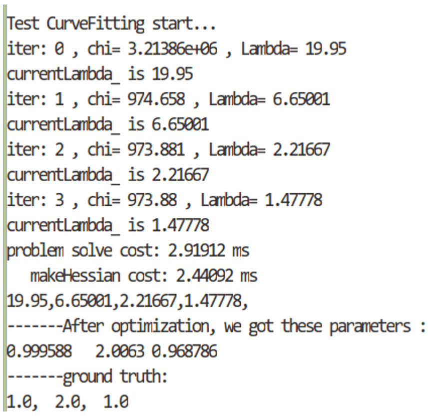
Figure 1: 阻尼因子 μ 随着迭代变化的曲线图

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。

```
#include <iostream>
#include <random>
#include "backend/problem.h"

using namespace myslam::backend;
using namespace std;

// 曲线模型的顶点，模板参数：优化变量维度和数据类型
class CurveFittingVertex: public Vertex
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW

    CurveFittingVertex(): Vertex(3) {}  // abc: 三个参数， Vertex 是 3 维的
    virtual std::string TypeInfo() const { return "abc"; }
};

// 误差模型 模板参数：观测值维度，类型，连接顶点类型
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) = 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
    {
        Eigen::Matrix<double, 1, 3> jaco_abc;  // 误差为维，状态量1 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
    };

    int main()
    {
        double a=1.0, b=2.0, c=1.0;          // 真实参数值
        int N = 1000;                          // 数据点
        double w_sigma= 1.;                  // 噪声值Sigma

        std::default_random_engine generator;
        std::normal_distribution<double> noise(0.,w_sigma);

        // 构建 problem
        Problem problem(Problem::ProblemType::GENERIC_PROBLEM);
        shared_ptr< CurveFittingVertex > vertex(new CurveFittingVertex
());

        // 设定待估计参数 a, b, 初始值c
        vertex->SetParameters(Eigen::Vector3d (0.,0.,0.));
        // 将待估计的参数加入最小二乘问题
        problem.AddVertex(vertex);

        // 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);

```

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

        std::cout<<"\nTest CurveFitting start..."<<std::endl;
        /// 使用 LM 求解
        problem.Solve(300);

        std::cout << "-------After optimization, we got these parameters :" << std::endl;
        std::cout << vertex->Parameters().transpose() << std::endl;
        std::cout << "-------ground truth: " << std::endl;
        std::cout << "1.0,  2.0,  1.0" << std::endl;

        // std
        return 0;
    }
```



```
Test CurveFitting start...
iter: 0 , chi= 3.21386e+06 , Lambda= 19.95
currentLambda_ is 19.95
iter: 1 , chi= 974.658 , Lambda= 6.65001
currentLambda_ is 6.65001
iter: 2 , chi= 973.881 , Lambda= 2.21667
currentLambda_ is 2.21667
iter: 3 , chi= 973.88 , Lambda= 1.47778
currentLambda_ is 1.47778
problem solve cost: 2.91912 ms
   makeHessian cost: 2.44092 ms
19.95,6.65001,2.21667,1.47778,
-------After optimization, we got these parameters :
0.999588   2.0063 0.968786
-------ground truth:
1.0,  2.0,  1.0
```

Figure 2: 运行结果截图

2 公式推导

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：

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

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

3 证明

证明下面的结论

> **阻尼因子 μ 大小是相对于 $\mathbf{J}^\top\mathbf{J}$ 的元素而言的。半正定的信息矩阵 $\mathbf{J}^\top\mathbf{J}$ 特征值 $\{\lambda_j\}$ 和对应的特征向量为 $\{\mathbf{v}_j\}$。对 $\mathbf{J}^\top\mathbf{J}$ 做特征值分解分解后有：$\mathbf{J}^\top\mathbf{J} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ 可得：**
>
> $$\Delta\mathbf{x}_{\mathrm{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j + \mu}\mathbf{v}_j \tag{9}$$

即证：$\Delta\mathbf{x}_{\mathrm{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j+\mu}\mathbf{v}_j$ 是式 1的解。

$$(\mathbf{J}^\top\mathbf{J} + \mu\mathbf{I})\Delta\mathbf{x}_{\mathrm{lm}} = -\mathbf{J}^T\mathbf{f} \tag{1}$$

将：$\Delta\mathbf{x}_{\mathrm{lm}} = -\sum_{j=1}^{n} \frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j+\mu}\mathbf{v}_j$ 代入：

$$\sum_{j=1}^{n}(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j + \mu}\mathbf{v}_j = \mathbf{J}^\top\mathbf{f} \tag{2}$$

考虑到 $\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}$ 的特征值为 $\lambda_j + \mu$：

$$\sum_{j=1}^{n}\mathbf{v}_j^\top \mathbf{F}'^\top \mathbf{v}_j = \mathbf{J}^\top\mathbf{f} \tag{3}$$

也即：

$$\sum_{j=1}^{n}\mathbf{v}_j^\top \mathbf{J}^\top\mathbf{f}\mathbf{v}_j = \mathbf{J}^\top\mathbf{f} \tag{4}$$

$$\sum_{j=1}^{n}\frac{1}{\lambda_j^2}\mathbf{v}_j^\top \mathbf{J}^\top\mathbf{J}\mathbf{J}^\top\mathbf{f}\mathbf{J}^\top\mathbf{J}\mathbf{v}_j = \mathbf{J}^\top\mathbf{f} \tag{5}$$

注意到其中有个常数项，将它交换位置：

$$\sum_{j=1}^{n}\frac{1}{\lambda_j^2}\mathbf{J}^\top\mathbf{J}\mathbf{v}_j\mathbf{v}_j^\top \mathbf{J}^\top\mathbf{J}\mathbf{J}^\top\mathbf{f} = \mathbf{J}^\top\mathbf{f} \tag{6}$$

将 $\mathbf{J}^\top\mathbf{J} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ 带入：

$$\sum_{j=1}^{n}\frac{1}{\lambda_j^2}\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{v}_j\mathbf{v}_j^\top \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{J}^\top\mathbf{f} = \mathbf{J}^\top\mathbf{f} \tag{7}$$

考虑特征向量之间线性无关，且 $\mathbf{V}$ 为正交矩阵，$\mathbf{\Lambda}$ 为对角矩阵，可得：

$$\sum_{j=1}^{n} \frac{1}{\lambda_j^2} \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top \mathbf{v}_j \mathbf{v}_j^\top \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top = \mathbf{I} \tag{8}$$

因此式 7显然成立，原式得证。