计算摄影学Lab5

姓名：葛帅琦
学号：3150102193

实验内容 - 非线性最小二乘

1. 理论基础

Gauss Newton 法来源于 Newton Rapson 法，后者利用二次型逼近目标函数。为了进行二次逼近，NR 法需要计算目标函数的 Hessian，但这并不是个容易完成的任务。GN 方法利用最小二乘问题本身的特点，通过计算 Jacobian 来近似 Hessian 。

NR 法采用二阶 Taylor 展开近似目标函数：
$F(x+\Delta x) \approx F(x) + J_F \Delta x + \frac{1}{2} \Delta x^T H_F \Delta x$

最小二乘问题具有$F(x) = \|R(x)\|_2^2$ 的形式，对 R 进行一阶 Taylor 展开，可以得到：
$F(x+\Delta x) = |R(x+\Delta x)|_2^2 \ \approx |R(x) + J_R \Delta x|_2^2 \ = |R(x)|_2^2 + 2R^T J_R \Delta x + \Delta x^T J_R^T J_R \Delta x \ = F(x) + J_F \Delta x + \Delta x^T J_R^T J_R \Delta x$
进行一阶 Taylor 展开，可以得到：

将两种展开方式进行对比，可知在最小二乘问题里$H_F \approx 2 J_R^T J_R$

继续依照 NR 法的思路，每一步迭代中，我们求解如下的标准方程：
$J_R^T J_R \Delta x + J_R^T R = 0$

这一标准方程对应于求 $J_R \Delta x = -R$的（线性）最小二乘解，可以采用共轭梯度法求解。本次实验中，可以直接采用 OpenCV 提供的矩阵算法完成。

得到Δx后，以它作为下降方向，我们进行线性搜索求出合适的步长 α ，更新 x ： $x \leftarrow x + \alpha \Delta x$
这样就得到了 GN 法，GN 法的算法框架如下：

- $x \leftarrow x_0$
- $n \leftarrow 0$
- while $n < n_{\max}$:
    - $\Delta x \leftarrow$ Solution of $J_R \Delta x = -R$ Conjugate Gradient or Other
- if $\|R\|_\infty \le \epsilon_r \vee \|\Delta x\|_\infty \le \epsilon_g$ return x
- $\alpha \leftarrow \arg\min_\alpha \{x + \alpha \Delta x\}$
- $x \leftarrow x + \alpha \Delta x$
- $n \leftarrow n + 1$

2. 实验要求

实现 Gauss Newton 求解最小二乘问题。

3. 接口设计

3.1 方程函数 ResidualFunction

```
class ResidualFunction {
public:
```

```
    virtual int nR() const = 0;
    virtual int nX() const = 0;
    virtual void eval(double *R, double *J, double *X) = 0;
};
```

`nR` 函数需要返回余项向量的维度。类似的， `nX` 函数需要返回变量 `X` 的维度。
`eval` 运行时读入 `X` 将计算得到的余项和Jacobian写入 `R` 和 `J` 。
- 计算得到的**余项向量**需要写入到 `R` ， `R` 需要预先分配好，长度为 `nR` 。
- 计算得到的 Jacobian 需要写入到 `J` ， `J` 需要预先分配好，大小为 `nR*nX` 。
- 输入的 `X` 是一个长度为 `nX` 的数组，包含所有的变量。

注意：你的优化器需要负责在优化开始前分配好 `R` 和 `J` 需要的空间，在结束后销毁。 `X` 作为输入，由用户（调用 `solve` 的程序）分配并填充好初始值。

优化的参数通过如下的结构体定义，各个参数的含义见注释。

3.2 GaussNewtonSolver 的定义如下：

```
class GaussNewtonSolver {
public:
    virtual double solve(
                ResidualFunction *f, // 目标函数
                double *X,           // 输入作为初值，输出作为结果
                GaussNewtonParams param = GaussNewtonParams(), // 优化参数
                GaussNewtonReport *report = nullptr // 优化结果报告
                ) = 0;
};
```

这里solve是纯虚函数，需要在子类中重写该函数。并且实现最小二乘求解算法。

- 当你的类的 `solve` 函数被调用时，它将采用 Gauss Newton 法最小化函数 `f` 。
- 输入时数组 `X` 内包含初始点，并且在优化后将被修改为最优点，维度需要与目标函数定义的维度一致。
- `param` 是 GN 算法运行的各种相关参数。
- 如果 `report` 不是空指针，优化完成后应当把相关的报告记录到 `report` 。
- 函数返回值是优化得到的目标函数的最优值。

3.3 优化参数结构体

优化的参数通过如下的结构体定义，各个参数的含义见注释。

```
struct GaussNewtonParams{
    GaussNewtonParams() :
       exact_line_search(false),
       gradient_tolerance(1e-5),
       residual_tolerance(1e-5),
       max_iter(1000),
       verbose(false)
    {}
    bool exact_line_search; // 使用精确线性搜索还是近似线性搜索
    double gradient_tolerance; // 梯度阈值，当前梯度小于这个阈值时停止迭代
    double residual_tolerance; // 余项阈值，当前余项小于这个阈值时停止迭代
    int max_iter; // 最大迭代步数
    bool verbose; // 是否打印每步迭代的信息
};
```

3.4 优化结果报告结构体

优化结果的详细信息储存的结构体定义如下，含义见注释。

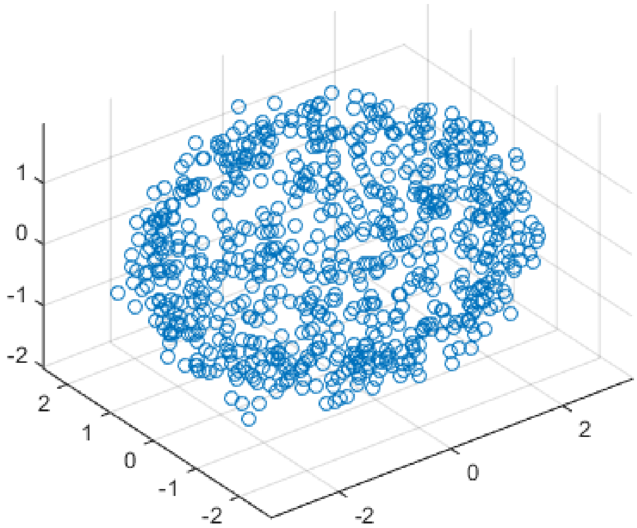
```
struct GaussNewtonReport {
    enum StopType {
        STOP_GRAD_TOL,        // 梯度达到阈值
        STOP_RESIDUAL_TOL,    // 余项达到阈值
        STOP_NO_CONVERGE,     // 不收敛
        STOP_NUMERIC_FAILURE  // 其它数值错误
    };
    StopType stop_type; // 优化终止的原因
    double n_iter;      // 迭代次数
};
```

4. 测试问题

作为测试，我们求解从三维点云拟合椭球的问题，我们的椭球模型是

$\frac{x^2}{A^2} + \frac{y^2}{B^2} + \frac{z^2}{C^2} = 1$

根据所给的731个数据点，求出A,B,C。数据我用python生成了一个data.h文件便于读取。



5. 算法实现过程

按照 1 中最后的伪代码写成，可以进行对照参考：

- $x \leftarrow x_0$
- $n \leftarrow 0$
- while $n < n_{\max}$:
    - $\Delta x \leftarrow$ Solution of $J_R \Delta x = -R$ Conjugate Gradient or Other

- if $\|R\|_\infty \le \epsilon_r \vee \|\Delta x\|_\infty \le \epsilon_g$ return x
- $\alpha \leftarrow \arg\min_\alpha \{x + \alpha \Delta x\}$
- $x \leftarrow x + \alpha \Delta x$
- $n \leftarrow n + 1$

```
class Sover2193 : public GaussNewtonSolver {
public:
    virtual double solve(
        ResidualFunction *f, // 目标函数
        double *X,           // 输入作为初值，输出作为结果
        GaussNewtonParams param = GaussNewtonParams(), // 优化参数
```

```
        GaussNewtonReport *report = nullptr // 优化结果报告
    )
    {
        double *x = X;
        int n = 0;
        double step = 1;
        int nR = f->nR();
        int nX = f->nX();
        double *J = new double[nR*nX];
        double *R = new double[nR];
        double *delta_x = new double[nR];
        while (n < param.max_iter) {
            f->eval(R, J, x);
            Mat mat_R(nR, 1,CV_64FC1, R);
            Mat mat_J(nR, nX, CV_64FC1, J);
            Mat mat_delta_x(nX, 1, CV_64FC1);
            cv::solve(mat_J, mat_R, mat_delta_x, DECOMP_SVD);

            double max_R = -1;
            double max_mat_delta_x = -1;

            for (int i = 0; i < nR; i++) { // get linf of R
                if (abs(mat_R.at<double>(i, 0)) > max_R) {
                    max_R = abs(mat_R.at<double>(i, 0));
                }
            }

            for (int i = 0; i < nX; i++) {
                if (abs(mat_delta_x.at<double>(i, 0)) > max_mat_delta_x) { // get linf of
delta_x
                    max_mat_delta_x = abs(mat_delta_x.at<double>(i, 0));
                }
            }

            if (max_R <= param.residual_tolerance) { // if linf of R is less than residua
l_tolerance, break
                report->stop_type = report->STOP_RESIDUAL_TOL;
                report->n_iter = n;
                return 0;
            }
            if (max_mat_delta_x <= param.gradient_tolerance) { // if linf of delta_x is l
ess than gradient_tolerance, break
                report->stop_type = report->STOP_RESIDUAL_TOL;
                report->n_iter = n;
                return 0;
            }

            // update step

            for (int i = 0; i < nX; i++) {
                x[i] += step * mat_delta_x.at<double>(i, 0);
            }

            n++;
        }
        // case of NO_CONVERGE
        report->stop_type = report->STOP_NO_CONVERGE;
        report->n_iter = n;
        return 1;
```

```
    }
};
```

6. 报告生成

Sover2193 :: Solve()函数再退出之后，会对Report结构体参数进行修改，以便得知具体Solve终止计算的原因。原因有四类：

> - 梯度达到阈值
> - 余项达到阈值
> - 不收敛
> - 其它数值错误

并且打印迭代次数。

7. 实验结果

实验心得：

以前觉得自己数学还行，但是上了这门课以后觉得自己数学和没学过一样。其实到最后我也没能理解必然最小二乘必然会收敛这件事，中间这一堆矩阵，雅克比矩阵，看的我晕头转向。但是老师上课讲的很好，助教的实验指导很详细，所以实现这个还是可以的。这才是计算机科学吧，感觉自己现在越来越局限一个做工程的了。
谢谢助教。