1.求解优化问题

Ceres 问题求解主要分成以下三部分：
（1）构建 cost function，即代价函数，也就是寻优的目标式(通过预测结果和测量值求误差的函数)。这个部分需要使用函子（functor）这一技巧来实现；
（2）通过上一步的代价函数构建待求解的优化问题;
（3）配置求解器参数并求解问题，这个步骤就是设置方程怎么求解、求解过程是否输出等，然后调用一下 Solve 方法。
————————————————————

问题：求解 x 值使得 1/2(10−x)2 最小
代码为：

```
#include<iostream>
#include<ceres/ceres.h>
using namespace std;

//第一部分：构建代价函数
struct CostFunctor {
    //模板函数，泛化函数类型
    template <typename T>
    //重载符号（），仿函数；传入待优化变量列表和承接残差的变量列表
    bool operator()(const T* const x, T* residual) const {     //三个 const 意思，从左到右依次是修饰指针不可改变指向的内容，修饰变量 x，修饰成员函数，防止成员函数修改被调用对象的值          //残差计算步骤

        residual[0] = T(0.5)*(T(10.0) - x[0])*(T(10.0) - x[0]);//1/2(10−x)^2

        return true;
    }
};

//主函数
int main(int argc, char** argv)//参考 https://blog.csdn.net/dgreh/article/details/80985928
{
    // 寻优参数 x 的初始值，为 5
    double initial_x = 5.0;
    double x = initial_x;

    // 第二部分：构建寻优问题
    //实例化 Problem
    ceres::Problem problem;
    //代价函数赋值
    //使用自动求导，将之前的代价函数结构体传入，第一个 1 是输出维度，即残差的维度，第二个 1 是输入维度，即待寻优参数 x 的维度。
```

```
//      ceres::CostFunction* cost_function = new AutoDiffCostFunction<CostFunctor, 1, 1>(new CostFunctor);
    //添加误差项，1、上一步实例化后的代价函数 2、核函数 3、待优化变量
    problem.AddResidualBlock(new ceres::AutoDiffCostFunction<CostFunctor, 1, 1>(new CostFunctor), nullptr, &x);

    //第三部分： 配置并运行求解器
    ceres::Solver::Options options;
    //配置增量方程的解法，此处为 QR 求解
    options.linear_solver_type = ceres::DENSE_QR;// Ax=b 矩阵分解方法，ceres 库中有 cholesky，SVD
    //是否输出到 cout
    options.minimizer_progress_to_stdout = true;
    //优化信息
    ceres::Solver::Summary summary;
    //求解：1、求解器 2、实例化 problem 3、优化器
    Solve(options, &problem, &summary);
    //输出优化的简要信息,迭代次数和每次的 cost
    std::cout << summary.BriefReport() << "\n";
    //最终结果
    std::cout << "初始值 x : " << initial_x<< "  迭代到-> " << x << "\n";
    return 0;
}
```

2.分析

第一部分：构建代价函数结构体

CostFunction 结构体中，对括号符号重载的函数中，传入参数有两个，一个是待优化的变量 x，另一个是残差 residual，也就是代价函数的输出。

```
struct CostFunctor {
    //模板函数，泛化函数类型
    template <typename T>
    //传入待优化变量列表和承接残差的变量列表
    bool operator()(const T* const x, T* residual) const {
        //残差计算步骤
        residual[0] = T(0.5)*(T(10.0) - x[0])*(T(10.0) - x[0]);//1/2(10-x)^2
        return true;
    }
};
```

第二部分：通过代价函数构建待求解的优化问题

这一步最主要的部分是残差块添加函数：AddResidualBlock 的使用，涉及到的三个参数分别是（1）自动求导代价函数；（2）是否添加核函数//核函数：Ceres 库中提供的核函数主要有：TrivialLoss 、HuberLoss、 SoftLOneLoss 、 CauchyLoss：减少离群点影响（3）待优化变量。

其中自动求导函数 AutoDiffCostFunction 可以单独进行，它内部的三个参数分别为（1）上述定义的代价函数结构体；（2）参差维度；（3）待优化变量的维度。

第三部分：配置优化器执行优化

这一部分实现求解器实例化；选择求解方式（这里用 QR 分解）;是否输出运行信息；优化器实例化；调用 Slove 函数进行问题求解；简要输出执行信息。
其中 Solve 函数很重要，它负责最后的问题求解，涉及到的三个参数分别是（１）求解器实例化；（２）优化问题实例化；(3)优化器实例化。