一、非线性优化的几个要素与ceres对应关系

1. 优化变量，即状态量
2. 残差，即观测减去预测。观测是传感器得到的数字量，预测是状态量的函数

残差（即观测减预测）

1. 核函数
2. 代价函数

但是，要注意，ceres里面的概念跟通常接触的概念在命名上有些差别

1. 优化变量，在ceres里面叫ParameterBlock，参数块。也可以理解，待优化参数嘛
2. 残差，在ceres里costFunction（代价函数，哈哈，就是让你去混淆的）。我们其实关心两件事：A. 残差如何定义；B. 如何求导
3. 残差的定义。它由一个struct CostFunctor来完成。这个struct其实是一个函数对象，因为它只干了一件事，就是在内部重载了()运算符。这个重载函数有几个要点：

* 它是模板函数，参数为T（实际使用是double和Jet两种类型）
* 它是const函数，其实这个无所谓，因为这个struct没有成员变量
* 它的参数const T\* const x, T\* residual两个，前面是优化变量（参数块），后面是残差
* 它的返回值是bool类型，一般返回true
* 函数体residual[] = ?

1. 如何求导？先看例子，以自动求导为例

CostFunction\* cost\_function =

new AutoDiffCostFunction<CostFunctor, 1, 1>(new CostFunctor);

也就是，用前面用来定义残差的CostFunctor构建一个有求导功能的cost\_function。注意它是一个模板类，有三个模板参数，一个是残差算子，一个是残差的维度（也就是观测的维度），一个是状态量的维度（即ParameterBlock的维度）。

1. 核函数，在ceres里面被成为loss Function（损失函数？怪怪的）
2. 代价函数

在我的定义里面，最后累加起来的一大坨东西称为代价函数。它对应的是ceres里面的Problem类。在ceres定义下，一个残差块是一项加权平方和，残差块叠加起来就变成了problem。所以ceres的problem需要一项一项往里添加残差块：

problem.AddResidualBlock(cost\_function, nullptr, &x);

AddResidualBlock函数有三个参数：残差，核函数，优化变量

1. ceres求解步骤
2. 定义problem，problem要成型需要不停AddResidualBlock，这里面又要定义残差、核函数以及优化变量。残差分两步来构造，先定义算子，再往上构造其求导方式。
3. Solver::Options
4. Solver::Summary
5. solve(options, &problem, &summary);
6. 求导方式
7. 自动求导。最常用，不多讲了
8. 数值求导。用在一些库函数调用下的情形，一般情况推荐自动求导
9. 解析导数。

解析导数用于导数的闭式形式容易得到，而自动求导用的链式法则比较麻烦的