$$\min_x E(x) = \min_x \frac{1}{2}\left(\sum_{i=1}^{N} e_i^T(x) Q^{-1} e_i(x)\right)$$ (Q^{-1}是信息增益矩阵，也可以看作权重)

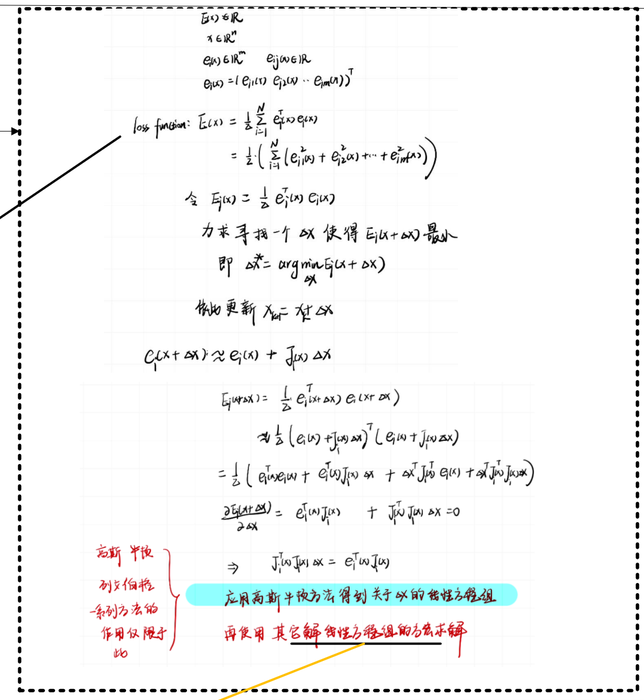
谈及G2O的使用，我们都知道这是一个非线性优化求解框架，但是具体针对的格式是什么呢？本文将一一详解

最基础的使用方法

G2O的使用方法

G2O的根本优化问题如上，是一个非线性无约束优化问题

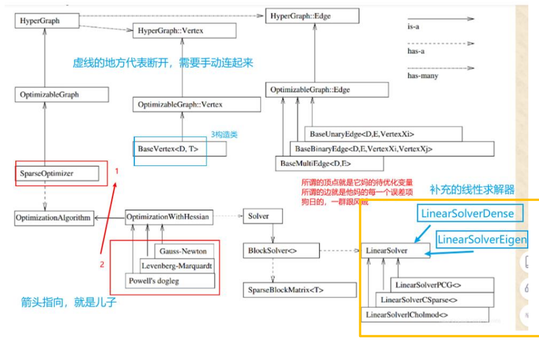
1.G2O解决的问题数学模型详细求解过程



2.G2O的使用流程

$$\frac{\partial E(x+\Delta x)}{\partial \Delta x} = e_1^T(x) J_1(x) + \cdots + e_N^T(x) J_N(x) + \left(J_1^T(x) J_1(x) + J_2^T(x) J_2(x) + \cdots + J_N^T(x) J_N(x)\right)\Delta x = 0$$

G2O的官方类继承图



图中有以下三个重点（对编程理解有帮助）：
1.实线箭头是直接包含关系；
2.虚线箭头是需要利用一个函数连接二者的；
3.箭头指向的关系：父亲---->儿子
4.编程的时候，从下往上写代码，向建房子一样

3.编程开始

依据(1)

首先要明白我们要构建的问题是右图里面的 $e_i(x)$

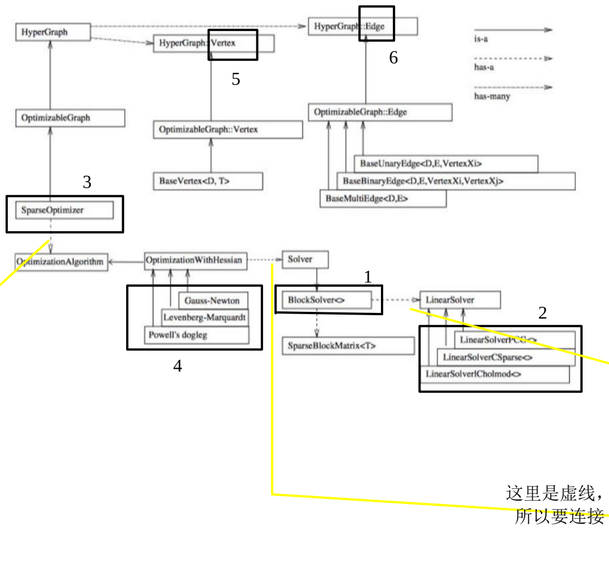
对于高博书里面的拟合曲线来说：$e_i(x) = y_i - e^{(ax^2+bx+c)}$

对于计算重投影误差(126页)来说：$e_{ij}(x) = u_{ij} - \frac{1}{s_i} K \exp(\xi^\wedge) P_j$

依据(2)

以拟合曲线为例子进行说明

(a) 先搭建(2)中的地基模块



solver模块

矩阵块求解器定义

```
typedef g2o::BlockSolver < g2o::BlockSolverTraits<3, 1> > block;
```

3是传给_PoseDim
1是传给_LandmarkDim
_PoseDim（位姿变量的维度）
_LandmarkDim(路标的维度)
但是对于拟合曲线这个问题如此理解不行：
得把3看作ei(x)中是待优化变量的维度(a,b,c) 3维
1看作误差项的维度 ei(x)是一维

创建线性求解器:用于求解Ax=b

```
block::LinearSolverType *linearsolver = new g2o::LinearSolverDense <block::PoseMatrixType>() ;
```

由最后的$\frac{\partial E(x+\Delta x)}{\partial \Delta x}$可以知道，最后得到的A的维度是3*3，3是待优化变量的维度

PoseMatrixType还不知道是什么意思，通过block，上面的维度为3的这个信息是通过block传递的，3和1共同决定了雅可比矩阵的维度

由于官方示例图中线性求解器和块求解器是虚线连接的，因此需要有一句代码把二者连起来，都是父亲找儿子

这里是虚线，所以要连接

```
block *b_slover = new block(linearsolver)
```

这里是虚线，所以要连接

这里是虚线，所以要连接

(b)再搭建(2)中的优化器模块

optimizer模块

创建优化器

```
g2o::SparseOptimizer optimizer;
```

选取优化算法

```
g2o::OptimizationAlgorithmLevenberg *solver = new g2o::OptimizationAlgorithmLevenberg(b_slover);
```

这里选取的是列文伯格优化算法
这里的优化算法主要的作用就是构造出Ax=b这个方程，再利用解齐次方程组的方法去求解

把优化器和优化算法连接起来

```
optimizer.setAlgorithm(solver);
```

(c)构建顶点（待优化变量，内部包括一个迭代更新表达式）

Vertex模块

这里的函数实现，看例程即可理解,这里创建的类没有input

通过类的方式实现，继承自BaseVertex，其中的模板参数D是待优化变量的维度，T是把待优化变量写成指定列向量的形式，T都是用eigen类型，有3个待优化变量，就用3维向量，4个就4维

内部有两个重要从父类派生过来的虚函数：
1.setToOriginImpl() （用于重置待优化变量的值）
2.oplusImpl()（用于更新待优化变量：常见的更新迭代式：x=x+delta x）

Edge模块

创建了一个Edge类
这里的类有输入input（输入值是下面式子中的xi；而yi是等到添加边的时候再传入；待优化变量是a,b,c）

$$e_i(x) = y_i - e^{ax_i^2+bx_i+c}$$

对于重投影误差，input是（这里得看看具体的程序才行，我怎么感觉右边全是未知数了）

$$e_i(x) = u_i - \frac{1}{s_i} K \exp(\xi^\wedge) P_w$$

(d)构建边（每一个误差项）$e_i(x)$

Edge类继承的是BaseUnaryEdge类
其中D是观测值维度，E是观测值的数据类型
VertexXi是连接的顶点类型（取决于你创建的顶点）

这里就是实现上面两个误差项计算的地方

注意内部编程的时候，有两大重要函数
1.computeError()（计算误差项）

2.这里的函数还不太了解

(e)添加顶点，再把顶点链接到边的端点上去边是几元边，就要链接几个顶点，最后再添加边

添加顶点（实际就是待优化变量）：
1.实例化顶点对象
2.给顶点的待优化变量赋初值
3.给顶点添加序列号
4.添加顶点

添加边（实际就是一个误差项）：
1.实例化边对象
2.给边添加序号
3.把指定顶点链接到边的指定端点上
4.给边传入观测值
5.添加边