从 Matrix 中继承

在从Matrix继承之前，真的，我的意思是真的，确保使用 EIGEN_MATRIX_PLUGIN 不是您真正想要的（请参阅上一节）。如果您只需要向Matrix添加几个成员，这就是要走的路。

实际需要继承Matrix 的一个示例是，当您有多个继承层时，例如 MyVerySpecificVector1, MyVerySpecificVector2 -> MyVector1 -> Matrix和 MyVerySpecificVector3, MyVerySpecificVector4 -> MyVector2 -> Matrix。

为了让您的对象在 Eigen 框架内工作，您需要在继承的类中定义一些成员。

这是一个简约的例子：

```
#include <Eigen/Core>
#include <iostream>

class MyVectorType : public Eigen::VectorXd
{
public:
    MyVectorType(void):Eigen::VectorXd() {}

    // 这个构造函数允许你从特征表达式构造 MyVectorType
    template<typename OtherDerived>
    MyVectorType(const Eigen::MatrixBase<OtherDerived>& other)
    : Eigen::VectorXd(other)
    { }

    // 此方法允许您将特征表达式分配给 MyVectorType
    template<typename OtherDerived>
    MyVectorType& operator=(const Eigen::MatrixBase <OtherDerived>& other)
    {
        this->Eigen::VectorXd::operator=(other);
        return *this;
    }
};

int main()
{
    MyVectorType v = MyVectorType::Ones(4);
    v(2) += 10;
    v = 2 * v;
    std::cout << v.transpose() << std::endl;
}
```

输出：

```
2 2 22 2
```

如果您不提供这些方法，就会出现这种错误

```
error: no match for ‘operator=’ in ‘v = Eigen::operator*(
const Eigen::MatrixBase<Eigen::Matrix<double, -0x000000001, 1, 0,
-0x000000001, 1> >::Scalar&,
const Eigen::MatrixBase<Eigen::Matrix<double, -0x000000001, 1>
>::StorageBaseType&)
(((const Eigen::MatrixBase<Eigen::Matrix<double, -0x000000001, 1>
>::StorageBaseType&)
((const Eigen::MatrixBase<Eigen::Matrix<double, -0x000000001, 1>
>::StorageBaseType*)(& v))))’
```