使用指南

为什么要在Matlab中使用Eigen。

Matlab中转换为单精度数据：

```
single(x)
```

Eigen本身非常容易使用，实现算法只要简单熟悉很快就能上手Quick reference guide.

如果我们想像Matlab函数一样调用，通过matlab工作区输入变量，并且返回的变量也能储存在工作区，matlab官方文档有提供一些例子，包括c和c++版本，但是这些例子并不好用，也不太好扩展。因此问题的难点就在于如何转换Matlab矩阵为Eigen矩阵或者说：如何将输入的Matlab工作区变量转换为Eigen可用的形式。

输入参数

对于输入矩阵，Matlab官方社区有如下解决方案：

```
 //! Extracts the pointer to underlying data from the non-const iterator (`TypedIterator<T
/*! This function does not throw any exceptions. */
        template <typename T>
        inline T* toPointer(const matlab::data::TypedIterator<T>& it) MW_NOEXCEPT {
                static_assert(std::is_arithmetic<T>::value && !std::is_const<T>::value,
                        "Template argument T must be a std::is_arithmetic and non-const ty
                return it.operator->();
        }
        template <typename T>
        inline T* getPointer(matlab::data::TypedArray<T>& arr) MW_NOEXCEPT {
                static_assert(std::is_arithmetic<T>::value, "Template argument T must be a
                return toPointer(arr.begin());
        }
        template <typename T>
        inline const T* getPointer(const matlab::data::TypedArray<T>& arr) MW_NOEXCEPT {
                return getPointer(const_cast<matlab::data::TypedArray<T>&>(arr));
        }
```

实际使用时在程序开头插入该模板，具体插入位置。使用时，只需要：

```
matlab::data::TypedArray<float> ca_input = std::move(inputs[2]);
auto ca_ptr = getPointer(ca_input);
MatrixXf ca = Map<MatrixXf>(ca_ptr,nzbc,nxbc);
```

`inputs[2]` 中 2 表示第三个输入变量，如下所示，ca作为eigen_staggerfd_single函数的第三个输入变量，在Eigen程序中成功输入。

```
[seismo_u,seismo_w,~]= eigen_staggerfd_single(input_vector,temp,ca,cl,cm,cm1,b,b1,s);
```

除此之外， `int nt = inputs[0][0];` 也能直接读取第一个输入变量的第一个数并转换为整数。这里input_vector是一个向量。

输出参数

输出数据时要把Eigen转换为Matlab，在Matlab文档中：

> createArray
> template
> TypedArray createArray(ArrayDimensions dims)
> template <typename ItType, typename T>
> TypedArray createArray(ArrayDimensions dims,
> ItType begin,
> ItType end,
> InputLayout inputLayout)
> template
> TypedArray createArray(ArrayDimensions dims,
> const T* const begin,
> const T* const end)
> template
> TypedArray createArray(ArrayDimensions dims,
> std::initializer_list data)

因此输出数据首先得到输出数据的维度，假设要输出一个nt*ng的矩阵，首先：

```
std::vector<size_t> size_output(1,nt);
size_output.insert(size_output.end(),ng);
```

得到矩阵维度，然后：

```
float* eig_seismo_u_ptr = seismo_u.data();
outputs[0] = factory.createArray<float>(size_output,eig_seismo_u_ptr,eig_seismo_u_ptr + number_
```

这里 seismo_u是要输出的矩阵，eig_seismo_u_ptr和eig_seismo_u_ptr + number_elements包括要输出的所有数据，number_elements为输出变量的元素个数，这里就是nt*ng.
Eigen 似乎并不支持三维张量，但是通过<size_t>也能将二维矩阵输出成三维张量，具体见代码。

调试过程

实现过程中格外要注意的是各种变量的维度是否匹配，一旦错误Matlab直接崩溃。直接在matlab中编写，在matlab中调试，调试起来并不方便。Eigen使用起来也比较友好，不过有些API官方也标注了还需要改进，用起来的效果也就不是很好。
编译程序只需要：

```
mex '-I\Intel\mkl\latest\include' eigen_e2drtm_single.cpp
```