# 重塑

密集矩阵和数组操作

从 3.4 版本开始，Eigen 公开了将矩阵重塑为另一个不同大小或向量的矩阵的便捷方法。所有情况都通过 DenseBase::reshape(NRowsType,NColsType) 和 DenseBase::reshape() 函数处理。这些函数不执行就地整型，而是返回输入表达式的*视图*。

# 重塑 2D 视图

更一般的整形转换是通过以下方式处理的：`reshaped(nrows,ncols)`。这是一个将 4x4 矩阵重塑为 2x8 矩阵的示例：

例子：

```
Matrix4i m = Matrix4i::Random();
cout << "Here is the matrix m:" << endl << m << endl;
cout << "Here is m.reshaped(2, 8):" << endl << m.reshaped(2, 8) << endl;
```

输出：

```
Here is the matrix m:
 7  9 -5 -3
-2 -6  1  0
 6 -3  0  9
 6  6  3  9
Here is m.reshaped(2, 8):
 7  6  9 -3 -5  0 -3  9
-2  6 -6  6  1  3  0  9
```

默认情况下，无论输入表达式的存储顺序如何，输入系数始终按列优先顺序进行解释。有关排序、编译时大小和自动大小扣除的更多控制，请参阅 DenseBase::reshape(NRowsType,NColsType) 的文档，其中包含许多示例的所有详细信息。

# 一维线性视图

重塑的一个非常常见的用法是在给定的二维矩阵或表达式上创建一维线性视图。在这种情况下，可以推导出大小并因此省略，如下例所示：

例子：

```
Matrix4i m = Matrix4i::Random();
cout << "Here is the matrix m:" << endl << m << endl;
cout << "Here is m.reshaped().transpose():" << endl <<
m.reshaped().transpose() << endl;
cout << "Here is m.reshaped<RowMajor>().transpose():  " << endl <<
m.reshaped<RowMajor>().transpose() << endl;
```

输出：

```
Here is the matrix m:
 7  9 -5 -3
-2 -6  1  0
 6 -3  0  9
 6  6  3  9
Here is m.reshaped().transpose():
 7 -2  6  6  9 -6 -3  6 -5  1  0  3 -3  0  9  9
Here is m.reshaped<RowMajor>().transpose():
 7  9 -5 -3 -2 -6  1  0  6 -3  0  9  6  6  3  9
```

此快捷方式始终返回列向量，并且默认情况下输入系数始终按列主序进行解释。同样，请参阅 DenseBase::reshape() 的文档以获得对排序的更多控制。

# 原地重塑

上面的例子创建了重新整形的视图，但是如何在给定的矩阵中重新整形呢？当然，此任务仅适用于具有运行时维度的矩阵和数组。在许多情况下，这可以通过PlainObjectBase::resize(Index,Index) 完成：

例子：

```
MatrixXi m = Matrix4i::Random();
cout << "Here is the matrix m:" << endl << m << endl;
cout << "Here is m.reshaped(2, 8):" << endl << m.reshaped(2, 8) << endl;
m.resize(2,8);
cout << "Here is the matrix m after m.resize(2,8):" << endl << m << endl;
```

输出：

```
Here is the matrix m:
 7  9 -5 -3
-2 -6  1  0
 6 -3  0  9
 6  6  3  9
Here is m.reshaped(2, 8):
 7  6  9 -3 -5  0 -3  9
-2  6 -6  6  1  3  0  9
Here is the matrix m after m.resize(2,8):
 7  6  9 -3 -5  0 -3  9
-2  6 -6  6  1  3  0  9
```

但是请注意，与 不同`reshaped`， 的结果`resize`取决于输入的存储顺序。因此它的行为类似于`reshaped<AutoOrder>`：

例子：

```
Matrix<int,Dynamic,Dynamic,RowMajor> m = Matrix4i::Random();
cout << "Here is the matrix m:" << endl << m << endl;
cout << "Here is m.reshaped(2, 8):" << endl << m.reshaped(2, 8) << endl;
cout << "Here is m.reshaped<AutoOrder>(2, 8):" << endl <<
m.reshaped<AutoOrder>(2, 8) << endl;
m.resize(2,8);
cout << "Here is the matrix m after m.resize(2,8):" << endl << m << endl;
```

输出：

```
Here is the matrix m:
 7 -2  6  6
 9 -6 -3  6
-5  1  0  3
-3  0  9  9
Here is m.reshaped(2, 8):
 7 -5 -2  1  6  0  6  3
 9 -3 -6  0 -3  9  6  9
Here is m.reshaped<AutoOrder>(2, 8):
 7 -2  6  6  9 -6 -3  6
-5  1  0  3 -3  0  9  9
Here is the matrix m after m.resize(2,8):
 7 -2  6  6  9 -6 -3  6
-5  1  0  3 -3  0  9  9
```

最后，当前不支持将重构矩阵分配给自身，并且由于aliasing将导致未定义行为 。禁止以下行为：

```
A = A.reshape(2,8);
```

这样可以：

```
A = A.reshape(2,8).eval();
```