存储顺序

密集矩阵和数组操作

矩阵和二维数组有两种不同的存储顺序：列优先和行优先。本页解释了这些存储顺序以及如何指定应使用的存储顺序。

列优先和行优先存储

矩阵的条目形成二维网格。然而，当矩阵存储在内存中时，条目必须以某种方式线性排列。有两种主要方法可以做到这一点，按行和按列。

如果一个矩阵是**逐行**存储的，我们就说它是按**行主**序存储的。首先存储整个第一行，然后是整个第二行，依此类推。例如考虑矩阵

$$A = \begin{bmatrix} 8 & 2 & 2 & 9 \\ 9 & 1 & 4 & 4 \\ 3 & 5 & 4 & 5 \end{bmatrix} \tag{1}$$

如果此矩阵以行优先顺序存储，则条目在内存中的布局如下：

```
8 2 2 9 9 1 4 4 3 5 4 5
```

在另一方面，一个矩阵被存储在**列优先**顺序，如果它是由列存储的列，从整个第一列，接着整个第二列，等等。如果上述矩阵以列优先顺序存储，则其布局如下：

```
8 9 3 2 1 5 2 4 4 9 4 5
```

此示例由以下特征代码说明。它使用PlainObjectBase::data()函数，该函数返回一个指向矩阵第一个条目的内存位置的指针。

例子

```
Matrix<int, 3, 4, ColMajor> Acolmajor;
Acolmajor << 8, 2, 2, 9,
             9, 1, 4, 4,
             3, 5, 4, 5;
cout << "The matrix A:" << endl;
cout << Acolmajor << endl << endl;

cout << "In memory (column-major):" << endl;
for (int i = 0; i < Acolmajor.size(); i++)
    cout << *(Acolmajor.data() + i) << "  ";
cout << endl << endl;

Matrix<int, 3, 4, RowMajor> Arowmajor = Acolmajor;
cout << "In memory (row-major):" << endl;
for (int i = 0; i < Arowmajor.size(); i++)
    cout << *(Arowmajor.data() + i) << "  ";
cout << endl;
```

输出

```
The matrix A:
8 2 2 9
9 1 4 4
3 5 4 5

In memory (column-major):
8  9  3  2  1  5  2  4  4  9  4  5

In memory (row-major):
8  2  2  9  9  1  4  4  3  5  4  5
```

Eigen 中的存储顺序

可以通过`Options`为Matrix或Array指定模板参数来设置矩阵或二维数组的存储顺序。由于Matrix类解释，Matrix类模板有六个模板参数，其中三个是强制性的（`Scalar`，`RowsAtCompileTime`和`ColsAtCompileTime`）和三个是可选的（`Options`，`MaxRowsAtCompileTime`和`MaxColsAtCompileTime`）。如果`Options`参数设置为`RowMajor`，则矩阵或数组按行主序存储；如果设置为`ColMajor`，则以列优先顺序存储。上面的Eigen程序中使用了这种机制来指定存储顺序。

如果未指定存储顺序，则Eigen默认将条目存储在 column-major 中。在这种情况下，typedefs（`Matrix3f`，`ArrayXXd`等）可以被方便的使用。

使用一种存储顺序的矩阵和数组可以分配给使用另一种存储顺序的矩阵和数组，就像上面程序中`Arowmajor`使用 初始化时发生的那样`Acolmajor`。Eigen将自动重新排序条目。更一般地，可以根据需要在表达式中混合行主矩阵和列主矩阵。

选择哪种存储顺序?

那么，您应该在程序中使用哪种存储顺序？这个问题没有简单的答案。这取决于您的应用程序。以下是一些需要牢记的要点：

- 您的用户可能希望您使用特定的存储顺序。或者，您可以使用Eigen以外的其他库，并且这些其他库可能需要特定的存储顺序。在这些情况下，在整个程序中使用此存储顺序可能是最简单和最快的。
- **由于更好的数据局部性，当矩阵以行优先顺序存储时，逐行遍历矩阵的算法将运行得更快。类似地，对于列主矩阵，逐列遍历更快。**进行一些试验以找出对您的特定应用程序来说更快的方法可能是值得的。
- Eigen 中的默认值是列优先。自然，Eigen库的大部分开发和测试都是使用列主矩阵完成的。这意味着，即使我们的目标是透明地支持列优先和行优先存储顺序，Eigen库也可能最适合列优先矩阵。