1.5 高级初始化

这一节讨论一些初始化矩阵的高级方法。

1.The comma initializer

eigen提供一个简单设定所有矩阵系数、向量、数组系数的初始化语法，从左上角开始到右下角结束。

[Matrix3f](http://eigen.tuxfamily.org/dox/group__matrixtypedefs.html" \l "ga276bae130c142e906ad8f47d24d11c1c) m;

m << 1, 2, 3,

4, 5, 6,

7, 8, 9;

std::cout << m;

//output

1 2 3

4 5 6

7 8 9

另外，这个初始化也可以将多个向量或者矩阵结合起来，但是必须记住你在使用初始化器之前需要设置大小：

RowVectorXd vec1(3);

vec1 << 1, 2, 3;

std::cout << "vec1 = " << vec1 << std::endl;

RowVectorXd vec2(4);

vec2 << 1, 4, 9, 16;

std::cout << "vec2 = " << vec2 << std::endl;

RowVectorXd joined(7);

joined << vec1, vec2;//vec1后接vec2

std::cout << "joined = " << joined << std::endl;

//output

vec1 = 1 2 3

vec2 = 1 4 9 16

joined = 1 2 3 1 4 9 16

我们也可以使用块来初始化矩阵：

MatrixXf matA(2, 2);

matA << 1, 2, 3, 4;

MatrixXf matB(4, 4);

matB << matA, matA/10, matA/10, matA;

std::cout << matB << std::endl;

//output

1 2 0.1 0.2

3 4 0.3 0.4

0.1 0.2 1 2

0.3 0.4 3 4

这样的初始化形式也能初始化矩阵的块，比如：

Matrix3f m;

m.row(0) << 1, 2, 3;

m.block(1,0,2,2) << 4, 5, 7, 8;

m.col(2).tail(2) << 6, 9;

std::cout << m;

//output

1 2 3

4 5 6

7 8 9

2.特殊的矩阵和数组

矩阵和数组类同样也有静态成员（访问得用作用域访问符）的方法比如Zero()，初始化所有系数为0。有三种用法：第一种是空括号，使用与固定大小的对象；第二种是带一个参数的括号，适用于一维的动态大小对象；第三种是带有两个参数的括号，适用于二维的动态大小的对象。如下：

std::cout << "A fixed-size array:\n";

Array33f a1 = Array33f::Zero();

std::cout << a1 << "\n\n";

std::cout << "A one-dimensional dynamic-size array:\n";

ArrayXf a2 = ArrayXf::Zero(3);

std::cout << a2 << "\n\n";

std::cout << "A two-dimensional dynamic-size array:\n";

ArrayXXf a3 = ArrayXXf::Zero(3, 4);

std::cout << a3 << "\n";

//output

A fixed-size array:

0 0 0

0 0 0

0 0 0

A one-dimensional dynamic-size array:

0

0

0

A two-dimensional dynamic-size array:

0 0 0 0

0 0 0 0

0 0 0 0

类似地，静态成员Constan(value)设置所有系数为某个常数。如果对象的大小需要确定，则在数值参数之前需要额外的两个参数，例如MatrixXd::Constant(rows, cols, value)。Random()的静态成员是给矩阵或数组设置任一的值。单位矩阵是用Identity()函数，这个函数只适用与矩阵，而不适用与数组，因为单位矩阵是线性代数的概念。函数LinSpaced(size,low,high)只适用与向量和一维数组，它让某一大小的向量或数组的系数均匀地从低到高的填满，如下所示：

ArrayXXf table(10, 4);

table.col(0) = ArrayXf::LinSpaced(10, 0, 90);

table.col(1) = M\_PI / 180 \* table.col(0);

table.col(2) = table.col(1).sin();

table.col(3) = table.col(1).cos();

std::cout << " Degrees Radians Sine Cosine\n";

std::cout << table << std::endl;

//output

Degrees Radians Sine Cosine

0 0 0 1

10 0.175 0.174 0.985

20 0.349 0.342 0.94

30 0.524 0.5 0.866

40 0.698 0.643 0.766

50 0.873 0.766 0.643

60 1.05 0.866 0.5

70 1.22 0.94 0.342

80 1.4 0.985 0.174

90 1.57 1 -4.37e-08

以上的函数都是将函数返回值赋值给需要的矩阵而不是直接在需要改变的矩阵上使用，如果想直接在现有矩阵上改变，可以使用setZero(),setIdentity(),setLinSpaced()函数：

const int size = 6;

MatrixXd mat1(size, size);

mat1.topLeftCorner(size/2, size/2) = MatrixXd::Zero(size/2, size/2);

mat1.topRightCorner(size/2, size/2) = MatrixXd::Identity(size/2, size/2);

mat1.bottomLeftCorner(size/2, size/2) = MatrixXd::Identity(size/2, size/2);

mat1.bottomRightCorner(size/2, size/2) = MatrixXd::Zero(size/2, size/2);

std::cout << mat1 << std::endl << std::endl;

MatrixXd mat2(size, size);

mat2.topLeftCorner(size/2, size/2).setZero();

mat2.topRightCorner(size/2, size/2).setIdentity();

mat2.bottomLeftCorner(size/2, size/2).setIdentity();

mat2.bottomRightCorner(size/2, size/2).setZero();

std::cout << mat2 << std::endl << std::endl;

MatrixXd mat3(size, size);

mat3 << MatrixXd::Zero(size/2, size/2), MatrixXd::Identity(size/2, size/2),

MatrixXd::Identity(size/2, size/2), MatrixXd::Zero(size/2, size/2);

std::cout << mat3 << std::endl;

//output

0 0 0 1 0 0

0 0 0 0 1 0

0 0 0 0 0 1

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

0 0 0 1 0 0

0 0 0 0 1 0

0 0 0 0 0 1

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

0 0 0 1 0 0

0 0 0 0 1 0

0 0 0 0 0 1

1 0 0 0 0 0

0 1 0 0 0 0

0 0 1 0 0 0

3.