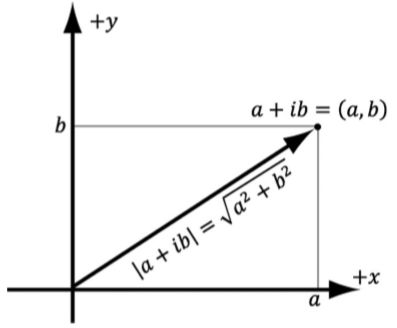
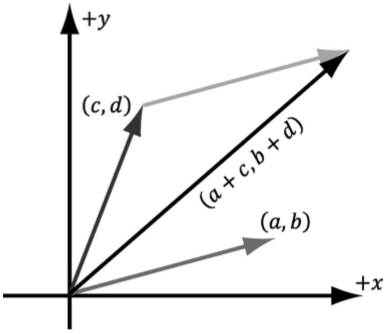
# 四元数(Quaternion)

四元数可视为复数的推广。在3D图形学中，我们使用单位四元数(unit quaternion)表示3D旋转，并具有便利的插值性质。

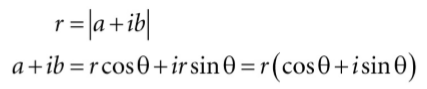
# 复数

我们先从复数着手。我们可以证明，复数p与单位复数相乘的结果，可视为在几何学上对p进行相应的旋转操作。

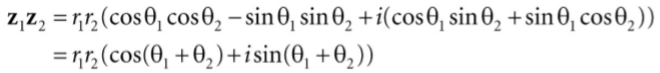
我们把形如z=a+bi的数称为复数，其中a称为实部，b称为虚部，i称为虚数单位，i的平方为-1。我们可以将其记为z=(a,b)，自然而然的，我们可以将其与几何学中的复平面内的2D点或2D向量联系起来。可以看到，其加法运算与向量加法一致，复数的模则对应向量的长度。



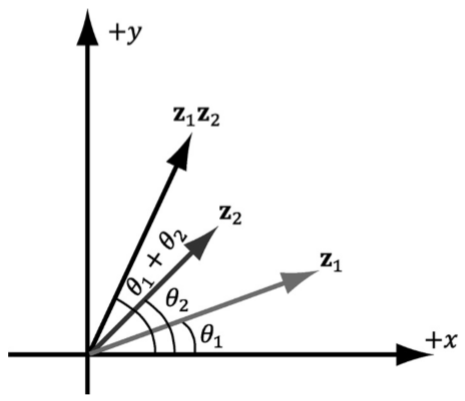
由于复数可以视作2D复平面内的向量，因此我们可以将其分量用极坐标来表示：



我们接下来以极坐标的方式来表示两个复数的乘法运算：



从几何学角度上来看，乘积得到的复数表示模为r1r2且与实轴夹角为两角之和的向量。特别的，当r2的模为1时，则在几何学上即表示将z1按z2的角度进行旋转。



因此，将复数乘以单位复数，其几何意义就是对2D点或向量进行旋转操作。

# 四元数

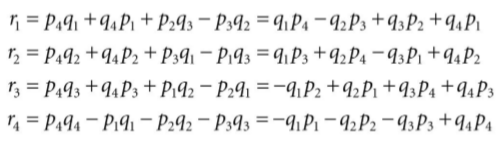
我们可以将四元数记为q=(x,y,z,w)=(q1,q2,q3,q4)。也可以简记为q=(u,w)。其中u=(x,y,z)为虚部向量，是四元数的虚数部分，而w为实部。

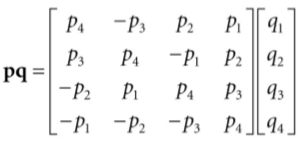
p与q相乘可以表示为矩阵的乘法：

(u, a)(v, b)=(av + bu + u × v, ab − u · v)

其中u × v =(p2q3–p3q2,p3q1–p1q3,p1q2–p2q1)，u · v=p1q1+p2q2+p3q3。

r=pq就能表示为以下形式





# 四元数的极坐标表示

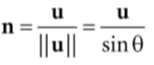
当一个四元数的模或称为范数为1的时候，我们将其称为单位四元数。

若q=(q1,q2,q3,q4)=(u,q4)为单位四元数，显而易见，其每个元素都在-1和1之间。这样必定存在一个角θ，使q4=cosθ。则

_scroll_external/attachments/image2021-4-24_17-57-34-a61929875aa338bc15d696e8ac6d7aa34e501d4061d85665eb3eeb898c6058be.png

_scroll_external/attachments/image2021-4-24_17-57-55-55b3b368bf469bcb8220dc21813176a1a315578c1278bf0be60e8d498d75e85a.png

将n表示为u方向上的单位向量，则



这样我们就可以将四元数表示为极坐标的形式

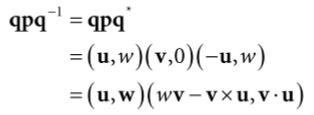
_scroll_external/attachments/image2021-4-24_17-58-35-ffef613c57c765fcd1c92c2046189198ffa7b75daff1f54ac42fc6a381733c5e.png

例如，有一四元数  q=(0,1/2,0,sqrt(3)/2)，将其转换为极坐标表示法，我们求得θ=arccos(sqrt(3)/2) ，就能得到结果_scroll_external/attachments/image2021-4-24_18-3-56-d36321138ae4f6a99904d7fe4444014140c6f9f6090a2b776b0348f075d2ccc1.png

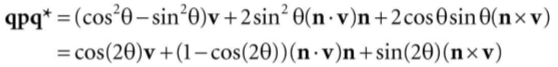
# 四元数的旋转操作

设q=(u,w)为单位四元数，v为3D点或向量。我们可以将v视为纯四元数p=(v,0)。单位四元数有一性质，其逆等于其共轭四元数。

我们考虑以下乘积

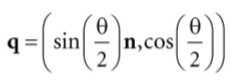


对其进行简化，最终可以得到



观察该式，其就是旋转公式。即该式令向量v绕轴n按角2θ进行旋转。

因此，若给定一个旋转轴n以及角θ，我们可以通过下式构建出相应的旋转四元数

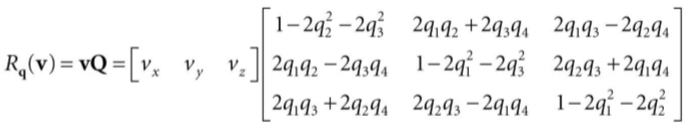


接着运用上面的公式则可求得旋转算子。

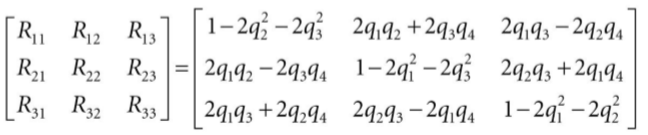
# 四元数旋转与矩阵

设q=(u,w)=(q1,q2,q3,q4)为单位四元数。我们也可以将旋转算子表示为矩阵形式:

_scroll_external/attachments/image2021-4-24_18-6-45-d64fd53f616072b73824b44f0adbc55d567a73d05910d59f643c05ba94f5e713.png

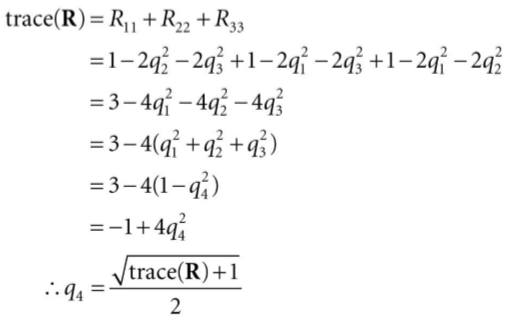


对应到旋转矩阵则为



若是反过来，知道旋转矩阵，要求四元数，则需要解出q1,q2,q3,q4。

我们可以将主对角线(左上至右下)相加，得到



从而求得q4，之后再抵消其他项来求得剩下的值。

# DirextX数学库中与四元数有关的函数

四元数由4个数字表示，故DirectX数学库用XMVECTOR类型来存储四元数。一下是一些常用的四元数函数

XMVECTOR **XMQuaternionDot**(XMVECTOR Q1, XMVECTOR Q2);

返回两个四元数的点积

XMVECTOR **XMQuaternionIdentity**();

返回四元数单位元(0,0,0,1)

XMVECTOR **XMQuaternionConjugate**(XMVECTOR Q);

返回四元数的共轭四元数

XMVECTOR **XMQuaternionLength**(XMVECTOR Q);

返回四元数的模

XMVECTOR **XMQuaternionNormalize**(XMVECTOR Q);

把四元数作为一个4D向量进行规范化处理

XMVECTOR **XMQuaternionMultiply**(XMVECTOR Q1, XMVECTOR Q2);

计算四元数的乘积

XMVECTOR **XMQuaternionRotationAxis**(XMVECTOR Axis, FLOAT Angle);

根据旋转轴及旋转角来计算相应的四元数

XMVECTOR **XMQuaternionRotationNormal**(XMVECTOR NormalAxis,FLOAT Angle);

与上函数类似，但这里的旋转轴需是一个规范化的向量，此函数的执行速度要比XMQuaternionRotationAxis快

XMVECTOR **XMQuaternionRotationMatrix**(XMMATRIX M);

根据给定的旋转矩阵来计算四元数

XMMATRIX **XMMatrixRotationQuaternion**(XMVECTOR Quaternion);

根据给定的四元数来计算旋转矩阵

VOID **XMQuaternionToAxisAngle**(XMVECTOR \*pAxis, FLOAT \*pAngle, XMVECTOR Q);

从四元数中提取旋转轴以及旋转角

XMVECTOR **XMQuaternionSlerp**(XMVECTOR Q0, XMVECTOR Q1, FLOAT t);

返回对q1和q2进行球面插值的结果

由于四元数表示旋转，通常我们在插值时希望旋转的速率保持恒定。若直接采用线性插值，则转速会忽快忽慢。如下图，在ab上进行均匀的线性插值，但投影到球面上则是不均匀的。

