Math-c Documentation

Quaternions

Math-c has built-in quaternions operations with the use of operators. Math-c detect a quaternion element any vector of size 1x4 or 4x1 when applying the operator "*" or "^" and execute the quaternion multiplication, as a linear vector multiplication can not be done between vectors of size 4x1 or between vectors of size 1x4.

Operators and functions

U * V operator

Complex base algebra operator or CB product.

Its a experimental algebra that can be an operation complementary to the quaternion multiplication(quat product). This has a different structure to set the x,y and z.

U ^ V operator

Quaternion operator.

It is quaternion multiplication.

U˚ operator

Vector conjugate, it is need to conjugate each element of vector is done with the function cj(U).

cinv(x)

CB inverse, is equal to the vector divided by the square magnitude. It is same quaternion inverse except for the signs.

$$V^{-1} = \frac{[a \quad b \quad c \quad -d]}{\|[a \quad b \quad c \quad d]\|^2}$$

qinv(x)

Quaternion inverse, is equal to the vector conjugate divided by the square magnitude.For the complex version check the Complex Quaternion section.

$$\overline{V} = \frac{[a \quad -b \quad -c \quad -d]}{\|[a \quad b \quad c \quad d]\|^2}$$

det(x)

Complex quaternion "determinant" is defined as.

$$(\overline{A_i} \wedge A_r + \overline{A_r} \wedge A_i)$$

QuaternionMode(a)

Set the complex quaternion multiplication mode. Parameter "a" is 0 to set the complex multiplication in default mode and parameter "a" to 1 to set the complex multiplication to Dual mode.(explained in Complex Quaternion section)

y = unitV(x)

return a scalar or quaternion of magnitude 1

x-> scalar value or quaternion

returns

y -> a scalar or quaternion with magnitude 1

CB Algebra Overview

The CB vector has a different structure than the traditional quaternion and is not associative, that means, U*(V*S) ≠ (U*V)*S, but can be transform to associative U * V = (x0*V)^(U*x0) (Note: x0=[1 0 0 0])

For more information can be found in CBALGEGRA

Example to rotate a vector V=(1.2 3.4 4.2) in a plane with Normal (0.729 0.5607 0.3925) at 60 degrees.

with CBAlgebra:

```
>>> θ=cos(30*π/180) <-> sin(30*π/180)*[-0.3925  0.5607  0.7290]
    θ = [0.866025  -0.19625  0.28035  0.3645]
>>> V=[1.2 3.4 4.2 0];
>>> V*θ*θ
  ans = [3.09838  0.698263  4.53328  0]
```

with classic Quaternions:

```
>>> q=cos(30*π/180) <-> sin(30*π/180)*[0.7290  0.5607  0.3925]
    q = [0.866025  0.3645  0.28035  0.19625]
>>> V=[0 1.2 3.4 4.2];
>>> q^V^q°
  ans = [-1.11022e-16  3.09838  0.698263  4.53328]
```

note the different coordinates position and signs.

One of the advantages of using the CB structure is that the same order in complex number and is easier to convert it to 2D(XY plane) and vice versa; because the third and fourth component are zeros in 2D CB product, like the complex numbers.

Example inverse kinematics 2 DOF in three dimensions without trigonometric functions.

Suppose two bones chain, one of magnitude k1=3 and other bone magnitude k2=5, the target position is r=(2 4 3 0), so d=|r|²=[29 0 0 0]; in plane normal nx=0.733333 ny=0.133333 nz=-0.666667.

Note: the vector (nx,ny,nz) is orthogonal to r; and x0=[1 0 0 0]

First the equations;

This can be solved with 2 simultaneous quaternion equations. The first equation is:

$$B_1+B_2=r$$

and then multiplying by r.

$$B_1r+B_2r=r\cdot r=d$$

The second equation is the following:

$$B_1 \cdot r - B_2 \cdot r = \begin{bmatrix} k_1^2 - k_2^2 \\ 4 \cdot A \cdot (-n_z) \\ 4 \cdot A \cdot (n_y) \\ 4 \cdot A \cdot (n_x) \end{bmatrix} = s$$

where the dot product part(first element of vector) is solved with the cosine law of triangle created (k1,k2,√(d(0)));

The second to fourth elements are the "cross" product(cross with the sign change and order) and A is the Area created in the triangle (k1,k2,d) solved with the heron formula.

$$\begin{matrix} B_1 \cdot r + B_2 \cdot r & = & d \\ B_1 \cdot r - B_2 \cdot r & & s \end{matrix}$$

Then we transform the equations to Quat products(Bn*x0 = Bn because the forth component of Bn is zero)

$$\begin{matrix} (x_0 \cdot r) \wedge B_1 + (x_0 \cdot r) \wedge B_2 & = & d \\ (x_0 \cdot r) \wedge B_1 - (x_0 \cdot r) \wedge B_2 & & s \end{matrix}$$ [equation 3]

Which can be represented by a Hypervector:

$$\left[(x_0 \cdot r) \quad (x_0 \cdot r) \right] \cdot \left[B_1 \quad B_2 \right] = \left[d \quad s \right]$$

And the solution is:

$$\begin{matrix} B_1 = \frac{1}{2}\left((\overline{x_0} \cdot r) \wedge d + (\overline{x_0} \cdot r) \wedge s \right) \\ B_2 = \frac{1}{2}\left((\overline{x_0} \cdot r) \wedge d - (\overline{x_0} \cdot r) \wedge s \right) \end{matrix} \cdots \rightarrow HyperVector \cdots \rightarrow \left[B_1 \quad B_2 \right] = \frac{1}{2}\left[\overline{x_0} \cdot r \quad \overline{x_0} \cdot r \right] \cdot \left[d \quad s \right]$$

Note: The general solution for 2 quaternion equations of two variables is:

$$\begin{matrix} A_1 \wedge B_1 + A_2 \wedge B_2 = C_1 \\ A_2 \wedge B_1 - A_1 \wedge B_2 = C_2 \end{matrix} \rightarrow \cdots \rightarrow \begin{matrix} B_1 = \overline{(\overline{A_1} \wedge A_2 + \overline{A_2} \wedge A_1)} \wedge (\overline{A_2} \wedge C_1 + \overline{A_1} \wedge C_2) \\ B_2 = \overline{(\overline{A_1} \wedge A_2 + \overline{A_2} \wedge A_1)} \wedge (\overline{A_1} \wedge C_1 - \overline{A_2} \wedge C_2) \end{matrix}$$

After we have the solution equation we can solve the problem in Math-c:

```
>>> [k1 k2 r] = [3 5 [2 4 3 0]];
    >>> d=[sumsq(r) 0 0 0];
    >>> A=heronf(k1,k2,√(d(0)));
    >>> s=k1^2-k2^2 <-> 4*A*[0.666667  0.133333  0.733333]
    >>> B1=0.5*(qinv(x0*r)^d+qinv(x0*r)^s)
      B1 = [-1.11575  2.69858  -0.687605  0]

    >>> B2=0.5*(qinv(x0*r)^d-qinv(x0*r)^s)
      B2 = [3.11575  1.30142  3.6876  0]
```

To find the second solution, we only have to change the sign of A, and repeat or just the conjugate "s".

```
>>> B1=0.5*(qinv(x0*r)^d+qinv(x0*r)^s°)
      B1 = [2.0123  -0.905473  2.03243  0]
    >>> B2=0.5*(qinv(x0*r)^d-qinv(x0*r)^s°)
      B2 = [-0.012297  4.90547  0.967568  0]
```

For 2D, the equation solution can be easily transformed to complex numbers and solve the complex matrix, example: if the target position is r=(4 6), so $d=|r|^2=52$; in plane normal nz=1; in the equation 3, we eliminate the third and forth element and the second element become to imaginary part.

$$\begin{bmatrix} r^{\cdot} & r^{\cdot} \\ r^{\cdot} & -r^{\cdot} \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}=\begin{bmatrix} d \\ k_1^2-k_2^2 \mp 4Ai \end{bmatrix}$$

```
>>> M = [4-6i 4-6i;4-6i -4+6i];
    >>> cD1=[52;3^2-5^2+4i*heronf(3,5,√(52))];
    >>> cD2=[52;3^2-5^2-4i*heronf(3,5,√(52))];
    >>> cB = inv(M)*cD1
      cB = [3i ;            //solution 1
          4+3i]
    >>> cB = inv(M)*cD2
      cB = [2.76923+1.15385i ;  //solution 2
          1.23077+4.84615i]
```

Quaternions

There is a lot of documentation available online about, so intros section will discuss the single rotation 3d.

The quaternion rotation(with quaternion structure) is done by the "sandwich" operation in which rotate twice the angle specified.

```
>>> q=cos(30*π/180) <-> sin(30*π/180)*[0.7290  0.5607  0.3925]
      q = [0.866025  0.3645  0.28035  0.19625]
    >>> V=[0 1.2 3.4 4.2];
    >>> q^V^q°
      ans = [-1.11022e-16  3.09838  0.698263  4.53328]
```

But is possible to do single rotation specifying the first value in input vector as in the example below with 60 degrees(without the half angle):

```
>>> q=cos(60*π/180) <-> sin(60*π/180)*[0.7290  0.5607  0.3925]
      q = [0.5  0.631333  0.48558  0.339915]
    >>> V=[0 1.2 3.4 4.2];
    >>> V(0)=-dot(q(1:3),V(1:3))/(1+q(0));
    >>> V
```

```
   ans = [-2.55748  1.2  3.4  4.2]
>>> V^q°
   ans = [2.55748  3.09835  0.698162  4.53316]
```

We can check the V(0) maintain its value(it is where the equation came from), except for the sign, it like it has an additional imaginary axis, but the explanation will be for the reader. The big issue with this rotation is the singularity the angle is 180 degrees, but this singularity can be eliminated if we multiplied by (1-q(0))/(1-q(0)) in a matrix form of the result, and this operation set the rotation in terms of normals components of the quaternion, but doesn't solve anything, which is the normal of quaternion [-1 0 0 0]?.

Complex Quaternions

The complex quaternions can be done in two ways:

Default mode

This is set with the function QuaternionMode(0) and it is the default mode.

The Complex multiplication in this mode is

$$A_r \wedge B_r - A_i \wedge B_i = C_r$$
$$A_i \wedge B_r + A_r \wedge B_i = C_i$$

also affects the qinv (to solve B only)function the complex quaternion inverse is

$$real(qinv(A)) = \overline{(\overline{A_i} \wedge A_r + \overline{A_r} \wedge A_i)} \wedge \overline{A_i}$$
$$imag(qinv(A)) = \overline{(\overline{A_i} \wedge A_r + \overline{A_r} \wedge A_i)} \wedge (-\overline{A_r})$$

also the det function for a complex quaternion is:

$$(\overline{A_i} \wedge A_r + \overline{A_r} \wedge A_i)$$

Examples:

to solve the inverse kinematic above using complex quaternions in the default mode, it is solved like:

Suppose two bones chain, one of magnitude k1=3 and other bone magnitude k2=5, the target position is r=(2 4 3), so d=|r|^2=29; in plane normal nx=0.733333 ny=0.133333 nz=-0.666667.

```
>>> [k1 k2 r] = [3 5 [2 4 3 0]];
    >>> d=[sumsq(r) 0 0 0];
    >>> A=heronf(k1,k2,√(d(0)));
```

```
>>> s=k1^2-k2^2 <-> 4*A*[0.666667  0.133333  0.733333]
>>> cR=(x0*r)+1i*(x0*r);
>>> cD=s+1i*d;

>>> cB=qinv(cR)^cD
   cB = [-1.11575+3.11575i  2.69858+1.30142i  -0.687605+3.6876i  0]    //solution 1
>>> cD=s°+1i*d;
>>> cB=qinv(cR)^cD
   cB = [2.0123-0.012297i  -0.905473+4.90547i  2.03243+0.967568i  0]    //solution 2
```

Quaternion Dual mode

This is set with the function QuaternionMode(1) and this mode is available only in the current environment scape, for example if you want to continue using this mode in a function, you have to select this mode inside the function too.

The Complex multiplication in this mode is:

$$A_r \wedge B_r = C_r$$
$$A_i \wedge B_r + A_r \wedge B_i = C_i$$

also affects the qinv function the complex quaternion inverse is:

$$real(qinv(A)) = \overline{A_r}$$
$$imag(qinv(A)) = -\overline{A_r} \wedge A_i \wedge \overline{A_r}$$

some of the uses of dual quaternions is for rotation and translation, also can be used to solve 3 simultaneous equations with 3 unknown variables.