**无约束刚体动力学**

**1模拟基础**

课程的这一部分是一个完整刚体运动的前向实现。在这部分，我们将展示为了模拟刚体运动所需的基本结构。在第二部分，我们实现刚体模拟器将要定义的关系，概念和方程。然后，我们给出一些代码去实现这些方程。一些概念和方程的背景我们将放在附录A。

在这里你仅需要知道常微分方程（ODE）解法的概念。如果你不熟悉这些内容，返回这个课程的开始，读“Differential Equation Basics”部分。你也最好读读 “Particle Dynamics”部分，虽然我们会重复部分内容。

模拟刚体运动通常伴随模拟质点运动，因此让我们从质点模拟开始。我们模拟质点运动的办法如下。我们用函数描述一个质点在时间*t*的空间位置。函数给出质点在时间*t*的速度。质点在时间*t*的位置和速度就是质点的状态。我们推广这个概念通过定义这个系统的状态向量。对于单个质点，

（1-1）

当我们讨论一个实际的实现，我们不得不“压扁”到一个数组。对于单个质点，能描述为一个6个元素数组，典型，我们让前3个元素描述，让其他的3个元素描述。稍后，当我们讨论状态是一个包含向量的矩阵同样作为一个向量，同样的“压扁”到一个数组。当然， 我们也能反转处理这个数组得到。所有这些使我们有一个简单的单一表格。从这开始，我们假设知道怎么转换到一个数组并且可以转换回去。

（1-2）

这里和是第*i*个质点的位置和速度。*n*个质点并不比一个质点更难，我们现在让成为一个单一质点的状态。

去实际模拟我们的质点的运动，我们需要知道更多的是；力作用在质点上的时间*t*。我们定义F(*t*)作为时间*t*作用在我们的质点上力。函数F(*t*)是所有作用在质点上力的总和，引力、风、弹性力、等。如果质点有质量*m，*随着时间改变的**Y**由下式给出。

（1-3）

给出**Y***(t)*的一些数值，方程（1-3）描述在时刻*t*，**Y***(t)*如何在瞬间变化。模拟从初始化条件**Y***(0)*（如：*x(0)、v(0)*）开始，然后使用微分方程的数值解去追踪**Y**随时间的改变，只要我们想可以一直追踪下去。如果我们想知道质点在一秒的位置，我们可以使用求解器去计算**Y**(1)，假如这里时间的单位是秒。如果我们想动画质点，我们需要计算**Y**(1/30)，**Y**(2/30)等等。

在谈到实际的实现时使用的数值方法相对不重要。让我们看看我们怎样实际的配合数值求解器，在C++类的语言中。假设我们访问数值求解器，我们通常写一个函数名字叫ode。通常，ode有下面的声明：

Typedef void (\*dydt\_func)(double t,double y[],double ydot[]);

Void ode(double y0[],double yend[],int len,double t0,double t1,dydt\_func dydt);

我们传送一个初始的状态向量到ode的数组y0。求解器ode不知道y0的内部结构。只到求解器能处理任意的问题，我也必须传送y0的长度参数len，（对于*n*个质点，显然len=6n）我们也传送给求解器模拟的开始和结束时间。*t0*和*t1*。求解器的目标是计算*t1*时刻的状态向量并且从数组中返回它。

我们也需要传入函数dydt进入ode。给定一个状态向量**Y***(t)*的编码数组y和时间t，必须计算通过数组ydot返回。（我们必须传入dydt的理由是，我们必须有一个时变力作用在我们的系统上。在这种情况下，dydt必须知道“现在什么时间”去确定力的数值）。在从*t0*到*t1*追踪**Y***(t)*的过程中，求解器ode可以在需要时调用dydt。给出的这个例程ode，我需要做的是编码出ode的参数dydt例程。

模拟刚体和模拟质点有相同的模式。不同的是状态向量**Y***(t)*包含更多的信息，和导数需要复杂一点。然而，我们将使用的刚体运动的求解器ode具有相同的追踪模式，同样我们要提供函数dydt。

**2 刚体的概念**

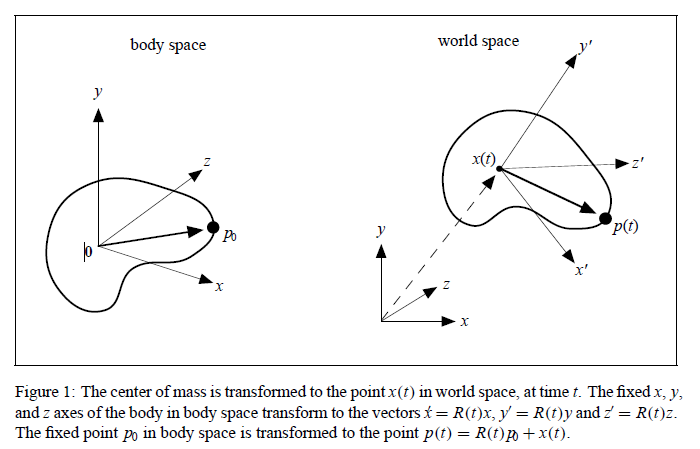
这部分目标是对于刚体给出一个类似（1-3）的方程。最终的微分方程将在2.11中给出。为了做这些，我们需要定义一些概念。更多的导数知识可以看附录A。在下一章，我们展示怎么写数值求解器需要的函数dydt去计算。

**2.1 位置和方位**

质点在时间t的空间位置能描述为*x(t)*，描述从原点开始质点的平移。刚体更加复杂，除了这些平移之外，我们也能旋转他们。在空间定位刚体我们使用一个向量*x(t)*描述刚体的平移。我们也必须描述刚体的旋转，通过一个旋转矩阵R*(t)*。我们称*x(t)*和R*(t)*是刚体的空间变量。

一个刚体，不像质点，具有空间的体积和形状。因为一个刚体能旋转和平移，我们定义刚体的形状是固定的和物质的空间位置是不变的。在物体空间给出物体的几何描述，我们使用x*(t)*和R*(t)*去变换物体空间（body space）进入世界空间（world space）（图1）。我将以此使用一些简化方程，我们需要刚体的描述使用物体空间原点在物体的质量中心，（0，0，0）。我稍后确切的定义质量中心，但是现在，质量中心能想成物体的几何中心。在描述物体的形状，我们需要这个几何中心位于物体空间的（0，0，0）。如果我们同意R*(t)*指的是物体关于质量中心的旋转，那么一个在物体空间中固定的向量*r*，在世界空间中将成为时刻*t*旋转的向量R*(t)r。*同样，如果p0在刚体空间中是刚体上任意的一点，那么*p0*世界空间的位置*p(t)*是首先关于原点旋转*p0*然后平移它：

（2-1）



由于物体的质量中心在原点，质量中心的世界空间位置总是直接为*x(t)*。这让我们附上物理的含义，*x(t)*是质量中心在*t*时的世界空间位置。我们也能给R*(t)*附上一个物理含义。考虑物体空间的*x*坐标轴向量（1，0，0）。在时间*t*，这个向量有方向

在世界空间，我们写出R*(t)*的具体元素

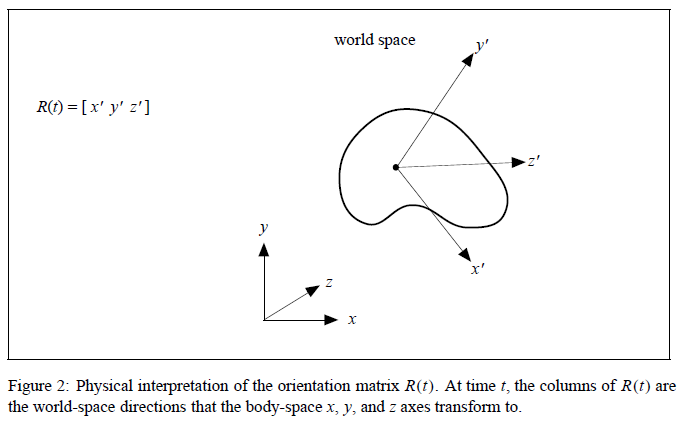
（2-2）

那么

（2-3）

是R*(t)*的第一列。R*(t)*的物理含义是，R*(t)*的第一列给出当变换到世界空间时刚体的x轴的方向矢量。类似的，R*(t)*的第二和第三列是刚体的y和z轴在世界坐标中的方向矢量。（图2）

和



**2.2 线速度**

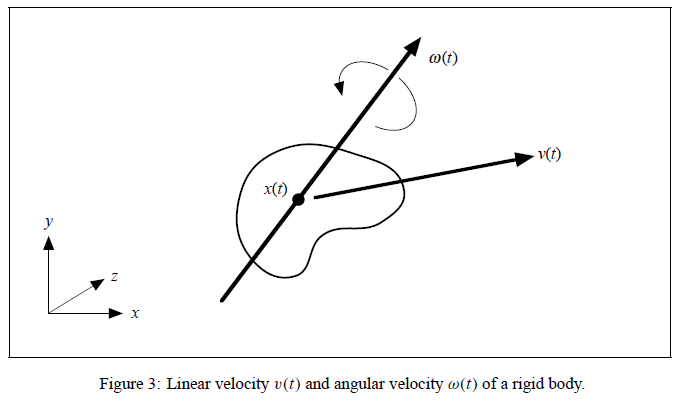
我们叫*x(t)*和R*(t)*是刚体在时间*t*时的位置和方向。我需要做的下一件事是定义这些点和方向怎么随时间改变。这意味我们需要表达和。由于*x(t)*是世界空间质量的中心位置，是世界空间中的质量中心点的速度。我们定义线速度*v(t)*作为这个速度

（2-4）

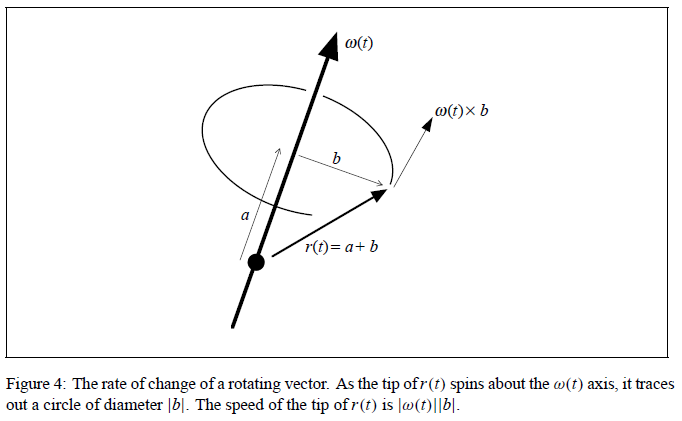
如果我们设想物体的方向是固定的，那么物体的运动就是纯粹的平移。*v(t)*给出这个平移的速度大小。

**2.3 角速度**

除了平移，刚体也能旋转。我们固定质量中心在空间中的位置，物体上的点的运动必定是绕着通过质量中心的一些轴旋转。（否则质量中心将移动）我们能描述一个旋转向量。的方向给出物体旋转轴的方向。（图3）的重要性，告诉物体旋转有多块。有时间尺度，描述物体在给定的周期中的旋转角度，if the angular velocity remains constant。就叫做角速度。



对于线速度，*x(t)*和*v(t)*有的关联。怎么关联R*(t)*和？（显然，不和关联，由于R*(t)*是一个矩阵而是一个向量）回答这个前，让我们回想R*(t)*的物理含义。我们知道R*(t)*的列告诉我们物体变换轴*x,y*和*z*的方向矢量。意思是必定描述轴*x,y*和*z*的变化速度。为了发现和的关联，让我们考虑在刚体上的一个任意向量的变化和角速度的关联。图4显示一个刚体具有一个角速度。考虑在时刻*t*世界空间中的一个向量*r(t)*。假如这个向量固定在物体上；那么，*r(t)*随着刚体在世界空间中转动。由于*r(t)*是一个方向，它和平移效果无关；特别的，和*v(t)*无关。考虑，我们分解*r(t)*为向量a和b，a是平行与，b垂直与。假设刚体保持恒定的角速度，*r(t)*的箭头方向指向为轴圆的边上。（图4）圆的半径是。由于矢量*r(t)*的箭头是瞬间在圆上移动，又因为*r(t)*的瞬间改变



是垂直与b和的。由于*r(t)*的箭头在半径为的圆上，矢量*r(t)*是在圆上瞬间移动的，*r(t)*瞬间的改变是垂直与b和。同时*r(t)*的箭头在半径为的圆上，*r(t)*的瞬间速度就是。由于b和是垂直的，它们的叉积有

（2-5）

把这些放在一起，我们可以写出。然而由于和a是垂直与，这样有因此

（2-6）

因此，我可以简单表达这个矢量改变率为

（2-7）

让我们把全部放到一起。在时刻*t，*我们知道刚体x轴在世界空间的方向是矩阵R*(t)*的第一列

在时刻*t，*R*(t)*第一列的导数是这个向量的变化率。使用叉积规则我们发现这个改变是

显然R*(t)*的其他两行也一样。这样我们写出下式

（2-8）

这个表达式太笨重。我们使用下面手法进行简化。如果a和b是3d-vectors，就是下式

给出向量a，让我们定义

那么

（2-9）

使用“\*”符号，我们重新写为更加简单的形式

（2-10）

根据矩阵乘法规则，我们提出因子

（2-11）

这是一个矩阵与矩阵的乘法。但是由于右边是矩阵R*(t)*本身，我们进一步简化为

（2-12）

这样我们的到了和的关系。注意 都是向量，旋转的是矩阵。

**2.4 物体的质量**

为了解答一些导数，我们需要通过刚体的体积执行一些整合。使得一些倒数比较简单，我们暂时假设刚体是有很多小质点组成的。质点的索引从1到N。第*i*个质点的质量是，每个质点在物体空间有一个位置。在*t*时刻第i个质点在世界空间的位置。可以得出下式

（2-13）

物体的总质量M是一个和

**2.5 质点的速度**

第*i*个质点的速度通过微分方程（2-13）获得：使用关系则

（2-15）

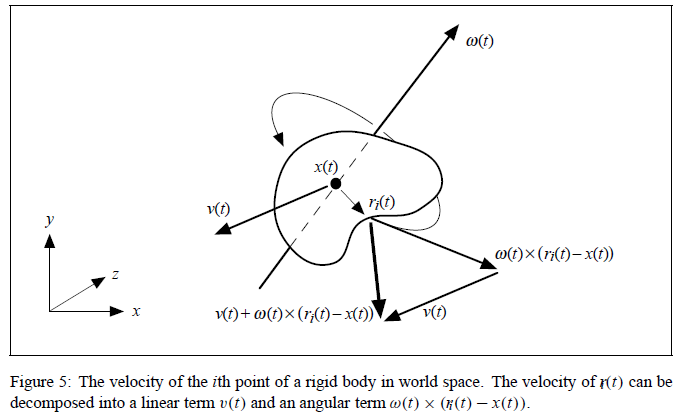
我们能重写成

（2-16）

使用的定义从方程（2-13）。重新使用“\*”操作符的定义。则

（2-17）

注意 刚体上的点的速度是两个部分（图5）：a是线性部分，和角速度部分



**2.6 质量中心**

我们的质量中心的定义是能够在线性方向和角度方向均衡的划分物体的动力学（dynamics）。物体的质量中心在世界空间中的定义是

这里M是物体的质量（全部）。但我们说我们使用一个质量中心坐标系统，我们的意思是物体空间。

注意隐含。

我们说在时刻*t*，存在质量中心*x(t)*。是否确定存在？是的：由于第*i*个质点在时刻*t*有位置，在时刻*t*的质量中心是

加上这个关系

也是很有用的。

**2.7 力和力矩**

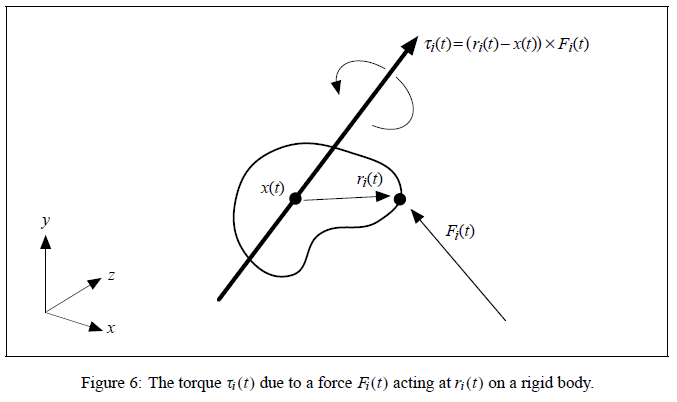
当我们设想有一个外力作用在刚体上（例如引力，风，摩擦力），我们假设这个力作用在某个物体的质点上。（记住我们的质点模型仅是一个概念，我们能使一个力施加在一些几何上或者物体的内部，因为我们总是可以假设它作用在某个准确的位置的质点上）。力作用在定义的质点的位置上。我们用代表在时刻*t*全部作用在第*i*个质点上的外力。同样，我们定义一个力矩作用在第*i*个质点上

（2-21）

力矩和力不同，作用在质点上的力矩依赖质点的位置，相对与质量中心*x(t)。*我们能直观的知道的方向成为物体由于旋转的转轴，如果质量中心位置固定（图6）。全部的外力是作用在物体上的和

总的外部力矩是

注意没有信息表达多少力作用在物体上；但是告诉我们作用在物体上的力的分布。



**2.8 线性动量**

具有质量*m*和速度*v*的质点的线性动量是*p*

（2-24）

刚体的总的线性动量是每个质点的质量和速度乘积的求和

从方程（2-17），速度因此物体总的线动量是

因为我们使用质量中心坐标系统，我们能应用方程（2-20）得

（2-27）

刚体总的线性动量等同于质量为M的质点的*v(t)* 。因此，我们在*P(t)*和*v(t)*之间有一个简单的关系，如果质量是常量

（2-28）

线性动量的概念让我们表达作用在刚体上的全部力简单的。附录A有下面关系的来源

（2-29）

线动量的改变等效与施加在物体上的全部的力。注意*P(t)*没有告诉我们物体的旋转速度，*F(t)*也没有表达物体的旋转速度。由于*P(t)*和*v(t)*的关系是简单的，我们可以使用*P(t)*作为刚体的一个状态变量替代*v(t)。*当然我们也可以使用*v(t)*作为状态变量，使用下面的表达式

（2-30）

然而，使用*P(t)*取代*v(t)*作为状态将和处理角速度和角加速度相同。

**2.9 角动量**

线动量的概念是一个简单直观的（*P(t)=Mv(t)*），刚体的角动量概念就不是了。仅有一个原因干扰我们写出简单的角动量方程那就是你不了解角动量。最好不要费心用一个直观的物理去解释角动量，总的来说角动量不是一个直观的概念。角动量最终是一个简单的方程因为它是守恒的。角速度不守恒，如果你有一个物体悬浮在空中不给它施加力矩，这个物体的角动量是常量。即使物体的角动量不变，物体的角速度可能也不守恒！因此，即使物体不受任何力的作用，物体的角速度还是能改变。因为这个原因，最终选择角动量作为状态变量而不是叫角速度。对于线动量，我们有关系*P(t)=Mv(t)。*类似的，我们定义一个刚体总的角动量*，*这里*I(t*)是一个矩阵叫做惯量张量（*inertia tensor*），我们将描述瞬间的情况。惯量张量*I(t*)描述物体相对于质量中心的质量分布。惯量*I(t*)依赖与物体的方向，但是也依赖于物体的平移。注意线动量是速度的线性函数，角动量是角速度的线性函数，并且比例因子是一个矩阵。注意*L(t)*不依赖与平移效果，*P(t)*也不依赖与旋转效果。角动量和力矩之间的关系是

（2-31）

**2.10 惯量张量**

惯量张量*I(t)*是角动量*L(t)*和角速度之间的比例因子，时刻*t*，第i个质点的位移。张量*I(t)*可以表示位为关于的对称矩阵

（2-32）

对于实际的实现，我们使用积分来替代求和计算物体的世界空间体积。质量用密度函数替换。初看起来，每当R*(t)*改变我们就需要去评估积分来找到。这将是模拟过程计算花费太大，除非物体的图形较为简单（例如球和立方体）积分能作出评估。幸亏，通过物体空间坐标我们能方便的预先计算*R(t)*下在一些方向上的惯量张量。（积分通常可以在模拟前得到，作为一个物体的物理属性输入）使用因子，我们有

拿去外面的乘积

用**1**表示单位矩阵，我们可以写出*I(t)*

由于这里是一个常量，。然后，由于

如果我们定义

这样我们根据前一个方程有

（2-39）

由于定义在物体空间，模拟时他是不变的（常量）。这样，我们可以在模拟前预先计算，我们可以从和*R(t)*容易的计算出*I(t)*。5.1章节我们得到一个长方体对象在物体空间的惯量张量通过积分在物体空间对象的体积。

*I(t)*的逆由下式给出

（2-40）

由于对于旋转矩阵和。显然在模拟期间也是一个常量。

**2.11 刚体的运动方程**

最终，我们揭开我们需要定义的刚体状态向量*Y(t)*的全部概念。我们定义*Y(t)*

(2-41)

因此，刚体的状态是由位置、方向和它的线动量、角动量组成。物体的质量M和物体空间的惯量张量是一个常量，我们假设在模拟开始时我们知道。在给定的时刻，辅助量I(t)、和速度*v(t)*可以由下式计算

(2-42)

导数是

(2-43)

下一部分我们将给出一个计算的具体函数dydt实现。最后注意：宁可在*Y(t)*中把物体的方位表述为矩阵*R(t)*，倒不如使用四元数（*quaternions*）。第四章简单讨论使用四元数替换旋转矩阵，一个四元数是一个有四个元素的向量，能用来描述旋转。如果你用*q(t)*替换*Y(t)*中的*R(t)*，我们能把*R(t)*看成是辅助变量，可以从*q(t)*中计算出来。从L(t)中计算，第四章给出一个类似的表达式，用*q(t)*和表达的公式。

**3 计算**

让我们考虑对于刚体函数dydt的一个实现。代码使用C++，我们假设下面这些数据类已经实现，矩阵，三维的点。使用这些数据类型我们可以描述刚体类

struct RigidBody{

//不变的量

double mass; //质量M

matrix Ibody; //

//状态变量

triple x; //*x(t)*

matrixR*;//R(t)*

triple P,L;//*P(t),L(t)*

//导数量

matrix Iinv; //

triple v,omega;//*v(t)*,

//已经计算的量

triple force,torque;//,

};

并且我们假设一个全局的物体数组

RigidBody Bodies[NBODIES];

常量有mass、Ibody、Ibodyinv，假设数组Bodies已经对这些量有合适的计算。也初始化好了每个刚体的其他状态变量x、R、P和L。这一章的目标是实现旋转矩阵；第4章描述对四元数的必要改变。我们使用微分方程的数值求解器ode，传入真是的成员。

//复制状态信息到一个数组

void State\_to\_Array(RigidBody \*rb,double \*y){

\*y++ = rb->x[0];

\*y++ = rb->x[1];

\*y++ = rb->x[2];

for(int i = 0;i<3;i++)

for(int j=0;j<3;j++)

\*y++ = rb->R[i,j];

\*y++ = rb->P[0];

\*y++ = rb->P[1];

\*y++ = rb->P[2];

\*y++ = rb->L[0];

\*y++ = rb->L[1];

\*y++ = rb->L[2];

}

//从数组把数据复制到状态变量

void Array\_to\_State(RigidBody \*rb, double \*y){

rb->x[0] = \*y++;

rb->x[1] = \*y++;

rb->x[2] = \*y++;

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

rb->R[i,j] = \*y++;

rb->P[0] = \*y++;

rb->P[1] = \*y++;

rb->P[2] = \*y++;

rb->L[0] = \*y++;

rb->L[1] = \*y++;

rb->L[2] = \*y++;

/\* Compute auxiliary variables... \*/

/\* \*/

rb->v = rb->P / mass;

/\* \*/

rb->Iinv = R \* Ibodyinv \* Transpose(R);

/\* \*/

rb->omega = rb->Iinv \* rb->L;

}

注意State\_to\_Array的产生是为了计算辅助变量Iinv,v和oega。我们假设triple和matrix已经有恰当的算术运算被定义，并且Transpose返回矩阵的转置。

检查例程，我们看到刚体的状态表示为3+9+3+3=18个成员。

#define STATE\_SIZE 18

void Array\_to\_Bodies(double y[])

{

for(int i = 0; i < NBODIES; i++)

Array\_to\_State(&Bodies[i], &y[i \* STATE\_SIZE]);

}

和

void Bodies\_to\_Array(double y[])

{

for(int i = 0; i < NBODIES; i++)

State\_to\_Array(&Bodies[i], &y[i \* STATE\_SIZE]);

}

现在让我来实现dydt。同时假设存在例程

void Compute\_Force\_and\_Torque(double t, RigidBody \*rb);

计算在时间*t*力*F(t)*和扭矩作用在刚体上，同时存储*F(t)*和到rb->force和rb->torque中。Compute\_Force\_and\_Torque给出所有的力和扭矩：引力，风和其他作用在物体上的。使用这个例程，我们将定义dydt

void dydt(double t, double y[], double ydot[])

{

/\* put data in y[] into Bodies[] \*/

Array\_to\_Bodies(y);

for(int i = 0; i < NBODIES; i++)

{

Compute\_Force\_and\_Torque(t, &Bodies[i]);

ddt\_State\_to\_Array(&Bodies[i],&ydot[i \* STATE\_SIZE]);

}

}

数值求解器ode调用dydt并且负责给数组y分配足够的空间，和ydot[STATE\_SIZE\*NBODIES worth for each]。函数做真正的计算的工作，并存储它们进入数组ydot使用ddt\_State\_to\_Array。

void ddt\_State\_to\_Array(RigidBody \*rb, double \*ydot)

{

/\* copy into ydot \*/

\*ydot++ = rb->v[0];

\*ydot++ = rb->v[1];

\*ydot++ = rb->v[2];

/\* Compute \*/

matrix Rdot = Star(rb->omega) \* rb->R;

/\* copy into array \*/

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

\*ydot++ = Rdot[i,j];

\*ydot++ = rb->force[0]; /\* \*/

\*ydot++ = rb->force[1];

\*ydot++ = rb->force[2];

\*ydot++ = rb->torque[0]; /\* \*/

\*ydot++ = rb->torque[1];

\*ydot++ = rb->torque[2];

}

例程Star，用来计算

matrix Star(triple a);

和返回矩阵

上面给出全部，时间执行模拟是简单的。假设全部刚体的状态有例程InitStates初始化我。我们模拟运行10秒，每1/30秒调用例程一次DisplayBodies去显示物体。

void RunSimulation()

{

double y0[STATE\_SIZE \* NBODIES],

yfinal[STATE\_SIZE \* NBODIES];

InitStates();

Bodies\_to\_Array(yfinal);

for(double t = 0; t < 10.0; t += 1./30.)

{

/\* copy yfinal back to y0 \*/

for(int i = 0; i < STATE\_SIZE \* NBODIES; i++)

y0[i] = yfinal[i];

ode(y0, yfinal, STATE\_SIZE \* NBODIES,t, t+1./30., dydt);

/\* copy 30 / into state variables \*/

Array\_to\_Bodies(yfinal);

DisplayBodies();

}

}

**4 四元数vs旋转矩阵**

有比使用矩阵更好的办法来表示刚体的方向。有几个理由，单位四元数，是长度归一化为单位长度的四个元素向量，它是一个比旋转矩阵更好的选择。对于刚体模拟通常有个重要的原因去避免使用旋转矩阵就是数值漂移，假如我们持续追踪刚体的方向按照下式

我们使用这个公式更新*R(t)*（积分这个方程），我们不可避免的遇到漂移。数值错误逐步加强，在旋转矩阵*R(t)*的反复计算。直观的，这种效果导致物体相位差。

这中问题可以通过使用单位四元数得到改善。由于四元数仅有4个参数，对于3个自由的旋转仅有一个多余的变量被附加。相比之下一个旋转矩阵使用9个参数去表述3个自由的旋转。显然，单位四元数多余的度数明显比旋转矩阵少。结果，四元数导致一个更小的漂移相比旋转矩阵。如果必须解释为什么四元数减少了漂移，那是因为它舍去了单位长度（*unit magnitude*）。重正化四元数到一个单位长度很容易。因为这两个特点，使用单位四元数*q(t)*描述物体的方向是合理的。我们仍然使用表示角速度。方位矩阵*R(t)*需要用来计算，它将作为一个*q(t)*的辅助变量。我们写一个四元数为[*s,v*]。使用这个符号，四元数乘法是

（4-1）

通过单位四元数表述绕单元轴做一个弧度的旋转

在使用四元数描述旋转中，如果和表示旋转，那么在瞬间表示和的组合旋转。我们将暂时怎么用四元数来处理第三章中处理方向的例程。在这之前，我们需要的表达式。附录B有这个公式

（4-2）

这里的乘法是一个标准的四元数 [0,和*q(t)* 的乘法。和方程（4-2）类似

实际四元数表述，我们需要重新定义RigidBody的类型

struct RigidBody{

//不变的量

double mass; //质量M

matrix Ibody; //

matrix Ibodyinv; //

//状态变量

triple x; //*x(t)*

quaternion q; *//q(t)*

matrixR*;//R(t)*

triple P,L;//*P(t),L(t)*

//导数量

matrix Iinv; //

triple v,omega;//*v(t)*,

//已经计算的量

triple force,torque;//,

};

下一步，在例程State\_to\_Array中我们替换双循环

for(int i = 0; i < 3; i++) /\* copy rotation matrix \*/

for(int j = 0; j < 3; j++)

\*y++ = rb->R[i,j];

为

/\*Assume that a quaternion is represented in terms of elements `r' for the real part, and `i', `j', and `k' for the vector part.\*/

\*y++ = rb->q.r;

\*y++ = rb->q.i;

\*y++ = rb->q.j;

\*y++ = rb->q.k;

类似的改变Array\_to\_State。由于Array\_to\_State是负责计算辅助变量，依赖与*R(t)，*Array\_to\_State也必须计算*R(t)*作为一个辅助变量

/\* Compute auxiliary variables... \*/

/\* \*/

rb->v = rb->P / mass;

/\* \*/

rb->Iinv = R \* Ibodyinv \* Transpose(R);

/\* \*/

rb->omega = rb->Iinv \* rb->L;

加一行

rb->R = quaternion\_to\_matrix(normalize(rb->q));

在计算rb->Iinv前，例程正则化返回q除以它的长度；单位四元数长度由normailize返回，然后传入quaternion\_to\_matrix返回一个矩阵。给出一个四元数*q*=[*s,v*]，quaternion\_to\_matrix返回矩阵

在这种情况你需要从一个旋转矩阵转换为四元数

quaternion matrix\_to\_quaternion(const matrix &m)

{

quaternion q;

double tr, s;

tr = m[0,0] + m[1,1] + m[2,2];

if(tr >= 0)

{

s = sqrt(tr + 1);

q.r = 0.5 \* s;

s = 0.5 / s;

q.i = (m[2,1] - m[1,2]) \* s;

q.j = (m[0,2] - m[2,0]) \* s;

q.k = (m[1,0] - m[0,1]) \* s;

}

else

{

int i = 0;

if(m[1,1] > m[0,0])

i = 1;

if(m[2,2] > m[i,i))

i = 2;

switch (i)

{

case 0:

s = sqrt((m[0,0] - (m[1,1] + m[2,2])) + 1);

q.i = 0.5 \* s;

s = 0.5 / s;

q.j = (m[0,1] + m[1,0]) \* s;

q.k = (m[2,0] + m[0,2]) \* s;

q.r = (m[2,1] - m[1,2]) \* s;

break;

case 1:

s = sqrt((m[1,1] - (m[2,2] + m[0,0])) + 1);

q.j = 0.5 \* s;

s = 0.5 / s;

q.k = (m[1,2] + m[2,1]) \* s;

q.i = (m[0,1] + m[1,0]) \* s;

q.r = (m[0,2] - m[2,0]) \* s;

break;

case 2:

s = sqrt((m[2,2] - (m[0,0] + m[1,1])) + 1);

q.k = 0.5 \* s;

s = 0.5 / s;

q.i = (m[2,0] + m[0,2]) \* s;

q.j = (m[1,2] + m[2,1]) \* s;

q.r = (m[1,0] - m[0,1]) \* s;

}

}

return q;

}

矩阵m的结构是m[0,0]，m[0,1]，m[0,2]是m的第一行。例程Array\_to\_Bodies和Bodies\_to\_Array不需要全部改变，但是注意常量要从18改成13，由于四元数比旋转矩阵需要少5个元素。仅仅的改变是我们要替换ddt\_State\_to\_Array中的

matrix Rdot = Star(rb->omega) \* rb->R;

/\* copy into array \*/

for(int i = 0; i < 3; i++)

for(int j = 0; j < 3; j++)

\*ydot++ = Rdot[i,j];

成

quaternion qdot = .5 \* (rb->omega \* rb->q);

\*ydot++ = qdot.r;

\*ydot++ = qdot.i;

\*ydot++ = qdot.j;

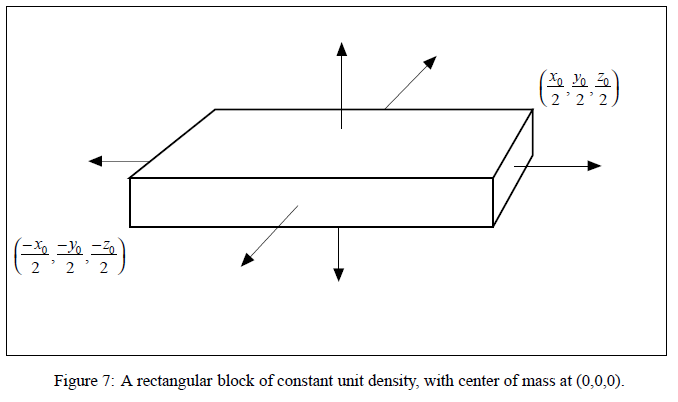
\*ydot++ = qdot.k;

我们假设rb->omega和四元数rb->q的乘法是四元数点积[0,rb->omega]q

**5 例子**

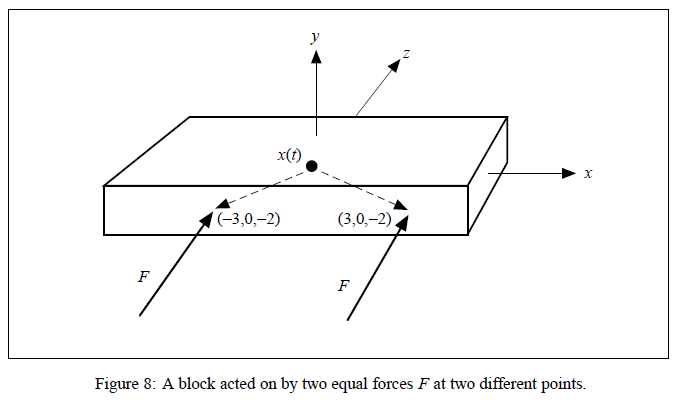
**5.1 长方体的惯量张量**

让我们计算长方体的惯量张量图7。长方体有一个尺寸。需要将方块的质量中心放到原地。因此，顺着x轴方向从到的面积，y和z轴类似。与计算惯量张量，我们必须看待（2-32）中的求和为在块体积上的积分。让我们假设块有一个单位密度。这意思是密度函数总是1。由于块有体积x0y0z0，块的质量M是x0y0z0。那么在物体空间。



类似的和。现在解决斜向的。

（5-2）



（其他的是类似的）因为积分是全部对称，因此块的惯量张量是

（5-3）

**5.3 一致的力场**

假设有相同的力施加在物体上的每一个质点上。例如，典型重力场施加在刚体上的每一个力是。g是一个向下的向量。作用在物体上的引力是

（5-4）

质量中心产生一个加速，重力场的力矩是什么？力矩是一个求和

（5-5）

由方程（2-20）。我们能发现均匀重力场对物体的角动量没有影响。而且，重力场能被当成单一的力Mg作用在物体的质量中心。

**5.3 刚体的自由转动**

现在，让我们考虑相同的力作用在块上图8。假如一个外力*F=(0,0,f)*作用在物体的点*x(t)+(-3,0,-2)*和*x(t)+(3,0,-2)*上，我们预料这将产生线性加速度，没有角加速度。作用在物体上的力是*(0，0，2f)*，质量中心的加速度是

沿着z轴。力作用的在*x(t)*+(-3.0,0,-2)上的力矩的力矩

作用在*x(t)*+(3.0,0,-2)上的力矩

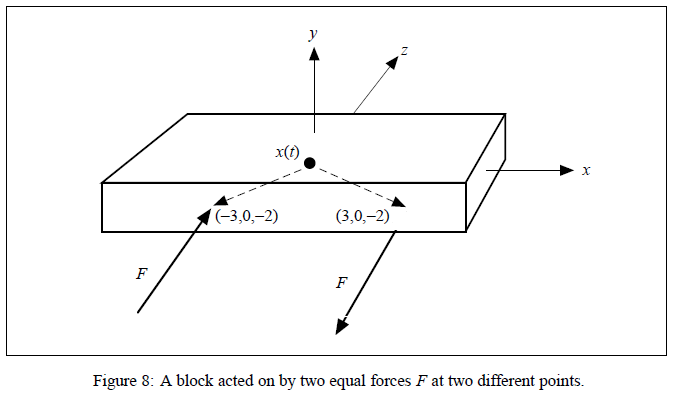
总的力矩

加入F

作用在块上的力没有传递给角加速度。

**5.4 物体的自由平移**

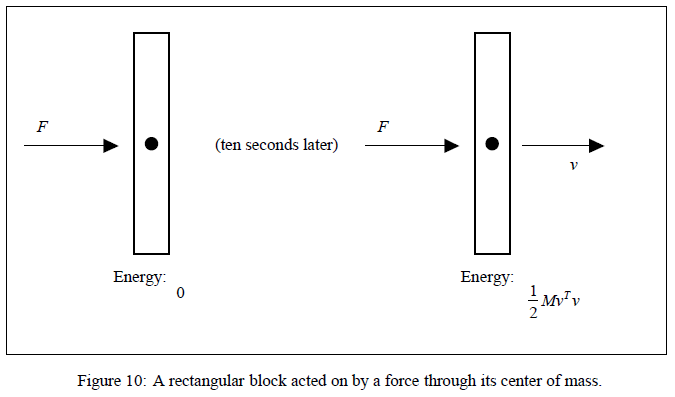
假如现在有一个外力作用在物体*x(t)+(-3,0,-2)*上，外力作用在物体*x(t)+(3,0,2)*上图9。

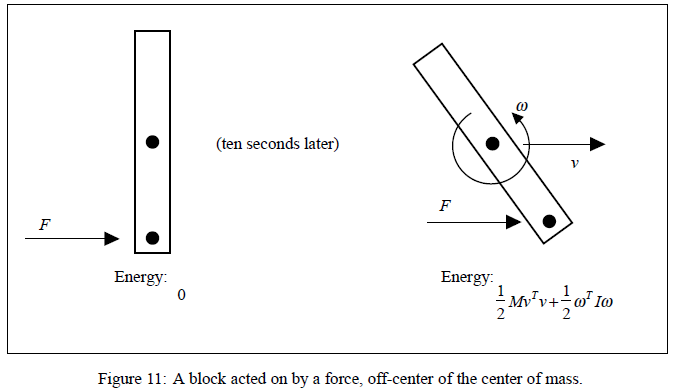


由于，作用在块上的净力是，在质量中心没有线加速度。另外，净力矩是

因此，净力矩是(*0,6f,0*)，平行与y轴。最终的结果是作用引起角加速。

**5.5 力vs力矩**

考虑力作用在物体上一点的效果，有时好像力是作用了两次。如果力*F*作用在物体上一点，第一我们考虑力*F*使质点中心的加速，然后考虑力也使物体旋转。这看似矛盾的，考虑一个长方体（图10）开始禁止。假如给它在质点中心施加一个力*F*一段时间，假如10秒，由于力作用在质点中心，没有力矩施加在物体上。10秒后，物体将得到一个线速度*v。*物体将不会获得角速度；长方体的能能量是。



现在假设相同的力*F*作用物体在偏离中心的位置上如图11。由于力作用在物体上是相同的，质心有相同的加速。结果，十秒后，物体将有线速度*v。*然而，十秒后，物体将得到一些角速度，因为力作用在偏离中心点上，现在应用作用在物体上的力矩。将有动量

长方体的能量高于作用在质量中心的时候。但是如果两种情况同一个力推这个物体，怎么能量会有不同呢？示意：能量，是力作用距离的积分。