

小良・台南・西元 2014 年 10 月

從剛體的轉動尤拉方程到轉動向量於姿態計算的實際應用

# 陀螺運動的物理引擎與動畫

## Part I 導言

以陀螺的三維運動而言，幾乎所有古典力學的書上推導出貼體角速度的尤拉方程之後，就會求諸 Euler angles 來得到 Lagrangian，接著用 elliptical integral 解出解析解。或者，以數值方法解其尤拉角的尤拉方程 ODE，然後再以尤拉角模擬其運動。不過，若對尤拉方程透徹理解，只要有貼體角速度，也可以利用轉動向量〔rotation vector〕，用姿態估測的方法直接簡單地數值模擬剛體轉動，這樣就不需要用到 Lagrangian 或 Euler angles。這個方法屬於姿態估測學中的方向餘弦遞推法〔iteration of direction cosine matrix, DCM〕。這裡將姿態〔orientation or attitude〕動力學與姿態估測應用在剛體轉動的尤拉運動方程上，以姿態估測中的方向餘弦法來積分尤拉方程的貼體角速度轉動向量，以此達到模擬剛體的轉動，並且應用在陀螺的三維運動上。這邊不只是只有介紹公式，還以淺顯易懂的方式給出姿態的運動方程，所以只要有基礎的線性代數矩陣知識，就可以掌握此方法。這裡詳述的這些理論在物理書籍上比較少見，反而在航空太空領域或電機的姿態控制領域才有講解，但大多也缺乏清楚的解釋，只給出複雜的公式，這樣在應用層面的時候會發生比較多不必要的試誤與嘗試。這裡我將這觀念連結錯綜複雜的東西清楚地寫下來，以供自己以及別人參考。

這裡詳述的方法可以廣泛的運用於任何剛體轉動或其尤拉方程。因此，此處所涉及的方向餘弦遞推的完整理解還可應用上其他相關領域。舉例來說，這裡在作陀螺模擬的姿態演算程式可以用在陀螺儀這類型的綁附式慣性感測器 [6, Ch 3.6.4] 上，積分測得的貼體角速度來作姿態估測，這將在第四章作介紹。仿間姿態演算法林立，但真能跑得起來的沒半個，因為那些程式是沒有經過驗證的。這邊從根本的推導開始，一步步進階到實用的程式碼，每一步推導出來的公式都有標明轉動方向是遵守左手還是右手定則，以及標明轉動矩陣是取其主動性轉向量或是被動性轉坐標軸，這一點是所有書上所共同缺乏的，而這在實際應用的時候尤其重要，這樣子推導出來的公式我們可以很方便的檢查其正確性，也可以很放心地應用在其他地方。另外，這邊提供與第三方程式碼的比較，驗證我們公式的正確性。這裡介紹的尤拉方程的數值化也可以應用上剛體物理模擬〔simulation of rigid body〕，這在電腦遊戲的物理引擎 [7, Ch 2.3] 中也是有其廣泛的應用性。這邊我們呈現以這些方法數值模擬高精度的三維陀螺運動，並且模擬結果也與其它文獻 [5] 作比較驗證了其正確性。這邊提供的詳細解說也適合當作大學物理系或資工系或航太系的練習題。

Wolfgang Christian 教授也有做過類似的陀螺模擬 [8]，不過他使用的是四元數遞推法，可惜剛體轉動部分的演算法並不公開，也沒有跟解析解作比較。我這裡是有跟尤拉角法的解析解做比較，不過我下一步也希望能跟四元數法這類的數值解能做比較，因此我還在找辦法把他的程

式的結果輸出出來。值得一提的是，他有把程式寫成一個獨立的可執行檔，公開在 comPADRE 網路上，此程式是互動性的，使用起來非常的方便，這也是我的目標之一。四元數法的理論也可以在 [9] 文獻中找到。

這裡的實際應用例子，也提供了一個，以尤拉方程來驗證姿態估測演算法的正確性的一個實例。即以尤拉方程解析解或數值解給出的貼體角速度來做姿態估測，結果再與解析解本身的尤拉角解析結果直接比較，這樣比較的即是姿態估測演算法的估測能力。這邊似乎是第一個完整融合尤拉方程與姿態估測的實際應用的例子。

這篇文章的架構是：

1. 第一部分為最完整的剛體轉動運動方程尤拉方程的推導證明。因為要做轉動軸的轉動向量數值積分必須要對尤拉方程有最正確的理解。這邊補充了 Goldstein Classical Mechanics[1] 中的證明觀念跳來跳去的缺失，以及大多數的書上推導尤拉方程解釋模糊的地方。轉動理論在航太科系書籍較有教導，但可惜講述通常過於複雜，要不就是過於簡化，缺乏與基本原理的結合，並且大部分也無提供實際實例操作這重要的一環，這裡則提供完整的陀螺模擬實例與完整公開的 code 供練習。
2. 接著藉由第一段尤拉方程的推導來嚴謹的證明 Euler equation 中的貼體角速度 (angular velocity along body frame) 可直接用於建立剛體特徵軸與 lab frame 間的主動與被動轉動矩陣，並以此推導出方向餘弦法的主要遞推原理來積分剛體轉動，追蹤每一時刻的剛體特徵軸在 lab frame 的位置。並以 python 程式編寫演算法，並以 python 繪圖庫 matplotlib 來作 3D 動畫。
3. 接著說明了以貼體轉動向量來近似  $t$  到  $t+dt$  時間的微小轉動時我們將以在  $t+dt$  時間的貼體角速度來近似，即以  $\vec{\omega}_b(t+dt)$  來建立  $dt$  時間內的轉動矩陣<sup>1</sup>。這裡也比較一般常見的  $\vec{\omega}_b(t)$  方法的結果來證明此優化所帶來精度上的提升。最後將以上方法應用上陀螺的三維運動，模擬結果將與文獻 [5] 的尤拉角法做比較。

## Part II

# 尤拉方程與姿態動力方程的推導

若你需要用到剛體轉動尤拉方程的計算，你應該會遇過下面的疑問。尤拉方程中的力矩項，明明在推導的時候是沿著固定的 space 座標取分量，為什麼在計算在應用的時候反而要沿著貼體的 body 座標取分量？(如 Goldstein classical mechanics 第二版書中 4-124 公式)。這邊很少有書上會去詳細解釋清楚，這裡從最根本出發一次解決你的疑惑。

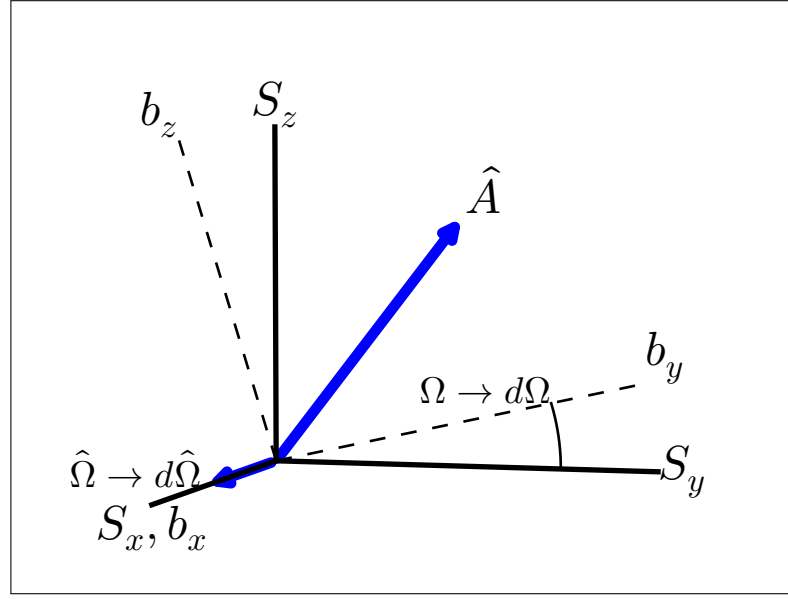
首先我們必須再次證明向量變化量在不同觀測座標中的關係，即公式1，這次你會發現在應用此公式的時候有不少地方是要注意的。由於當我實際在解這問題時我發現 Goldstein classical mechanics 書中還有幾點證明在應用層面還不夠清楚，讓我在應用的時候花了好多時間去釐清，因此這邊寫上我認為可以補充書上的推導證明。首先我們從以下公式開始

$$\left(\frac{d\vec{L}}{dt}\right)_s = \left(\frac{d\vec{L}}{dt}\right)_b + \vec{\omega} \times \vec{L} \quad (1)$$

對此公式的理解將是整篇文章最重要的基礎。此公式如何而來？此公式為一隨時間變動的向量在恆定座標與非恆定座標 (此例為轉動中座標) 之間線性變換的結果。以下是此公式的推導。

<sup>1</sup>The path order exponential of  $\vec{\omega}_b$  from time  $t$  to  $t+dt$ . This will be discussed more in the text.

Figure 1:



首先考慮一恆定座標  $S(\text{space})$ ，一轉動座標  $b(\text{body})$ ，為了方便討論矩陣轉換的主被動性與座標轉換的左右手法則，我們這邊方便的先假設  $\hat{S}_x, \hat{b}_x$  兩軸重合，因此圖中顯示了  $\text{body frame}$  沿著  $+\hat{S}_x$  遵守右手定則逆時針轉了  $\Omega$  角度，依右手定則此角位移向量  $\hat{\Omega}$  會在  $+\hat{S}_x$  方向。但是接下來的推導以及所有公式都適用任意的座標旋轉，這邊是為了方便討論矩陣的主動被動的方向性，以及在之後的推導方便我們追蹤正負號以及矩陣主動被動意義的改變，因此在圖中做了一個方便我們思考的情形。另外，大部分書上在討論座標轉換時有時候給的公式是遵守左手定則，但這與物理定律所採納的右手定則相反，因此這邊我寫下完整的右手定則的推導，希望之後的人不需要像我一樣花了大半時間在轉換不同公式間左手右手定則帶來的正負號的改變。

依照圖1所示，我們可以寫下  $\vec{A}$  向量在  $S, b$  座標間的關係

$$(\vec{A})_b = \Omega_{\text{passive, r.h.}} (\vec{A})_s$$

其中  $\Omega$  是  $s$  frame 到  $b$  frame 的座標轉換矩陣，因為是轉換座標軸，因此矩陣取被動含意，並且我們採用右手定則，因此逆時針方向為正方向。接下來只要有用到矩陣的運算我都會標明主被動及左右手 (r.h. right-hand or l.h. left-hand)，這樣我們可以很快速對照圖表來理解轉動方向，這很重要。

若我們考慮  $\Omega$  的角度很小  $\Omega \rightarrow d\Omega$  (infinitesimal rotation)，則  $d\Omega$  矩陣與 unity matrix 相去不遠，可以寫成  $1(\text{unity matrix}) + \epsilon(\text{infinitesimal matrix})$ ， $\epsilon$  具有 antisymmetric matrix 的特性 [1, p. 169]，帶入上式

$$(\vec{A})_b = (1 + \epsilon)_{\text{passive, r.h.}} (\vec{A})_s$$

infinitesimal matrix 有個特性，很容易自行驗證，

$$\epsilon_{\text{r.h., passive or active}} = \begin{bmatrix} 0 & \epsilon_3 \geq 0 & -\epsilon_2 \leq 0 \\ -\epsilon_3 & 0 & \epsilon_1 \geq 0 \\ \epsilon_2 & -\epsilon_1 & 0 \end{bmatrix}, \quad \epsilon_{\text{l.h., passive or active}} = \begin{bmatrix} 0 & -\epsilon_3 \leq 0 & \epsilon_2 \geq 0 \\ +\epsilon_3 & 0 & -\epsilon_1 \leq 0 \\ -\epsilon_2 & \epsilon_1 & 0 \end{bmatrix}$$

現在我們考慮  $\vec{A}$  是  $+\hat{b}_y$  軸的狀況，不過考慮相同矩陣  $(1 + \epsilon)$  的主動特性，也就是主動轉向量，這樣的話轉動方向會與原本的方向相反，變左手定則，我們會得到

$$\left(\hat{S}_y\right)_s = (1 + \epsilon) \times \left(\hat{b}_y\right)_s \quad \text{active, l.h.}$$

整理一下

$$\left(\hat{b}_y\right)_s = \underbrace{[(1 + \epsilon)]^T}_{\text{active, r.h.}} \times \left(\hat{S}_y\right)_s = (1 - \epsilon) \times \left(\hat{S}_y\right)_s \quad \text{active, l.h.}$$

代入上面 r.h.  $\epsilon$  的公式 (因  $\epsilon$  還是原本的矩陣)，整理一下

$$\left(\hat{b}_y\right)_s - \left(\hat{S}_y\right)_s = - \begin{bmatrix} 0 & \epsilon_3 \geq 0 & -\epsilon_2 \leq 0 \\ -\epsilon_3 & 0 & \epsilon_1 \geq 0 \\ \epsilon_2 & -\epsilon_1 & 0 \end{bmatrix} \times \left(\hat{S}_y\right)_s$$

利用向量外積，上式也可寫成

$$\left(\hat{b}_y\right)_s - \left(\hat{S}_y\right)_s = (\vec{\epsilon})_s \times \left(\hat{S}_y\right)_s$$

其中  $\vec{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix}_s$  為一向量，在 S frame 中的分量為  $\epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3$ 。

現在我們將上式跟微小轉動公式 Rodrigues rotation formula 比較

$$\vec{r}' - \vec{r} = d\vec{\Omega} \times \vec{r}$$

$d\vec{\Omega}$  是 r 到 r' 的 r.h. 角位移向量，因此我們得到  $\vec{\epsilon} = d\vec{\Omega}$ ， $d\vec{\Omega}$  就是 s frame 到 b frame 的角位移向量 (follow r.h. rule)

$$\left(\hat{b}_y\right)_s - \left(\hat{S}_y\right)_s = (d\vec{\Omega})_s \times \left(\hat{S}_y\right)_s$$

這一點很重要，因為我們將證明此  $(d\vec{\Omega})_s$  跟接下來我們要推導的尤拉公式中的貼體角速度  $\vec{\omega}$  有直接相關性，並且以此來做我們模擬剛體轉動的基礎。

以上的討論是考慮  $\vec{A}$  向量不隨時間變動的情況，接下來我們必須討論  $\vec{A}$  以及 b frame 皆隨時間變動的狀況。

在時間 t 時我們令 S 與 b frame 重合，過了 dt 時間原本的  $\vec{A}$  向量加了一改變量  $d\vec{A}$  變成  $\vec{A}'$ ，並且 b frame 依右手定則轉動了一微小角度 (infinitesimal rotation)，在此前提下，向量  $\vec{A}$  在 t 時間符合

$$\left(\vec{A}\right)_{s(t)} = \left(\vec{A}\right)_{b(t)} \quad (2)$$

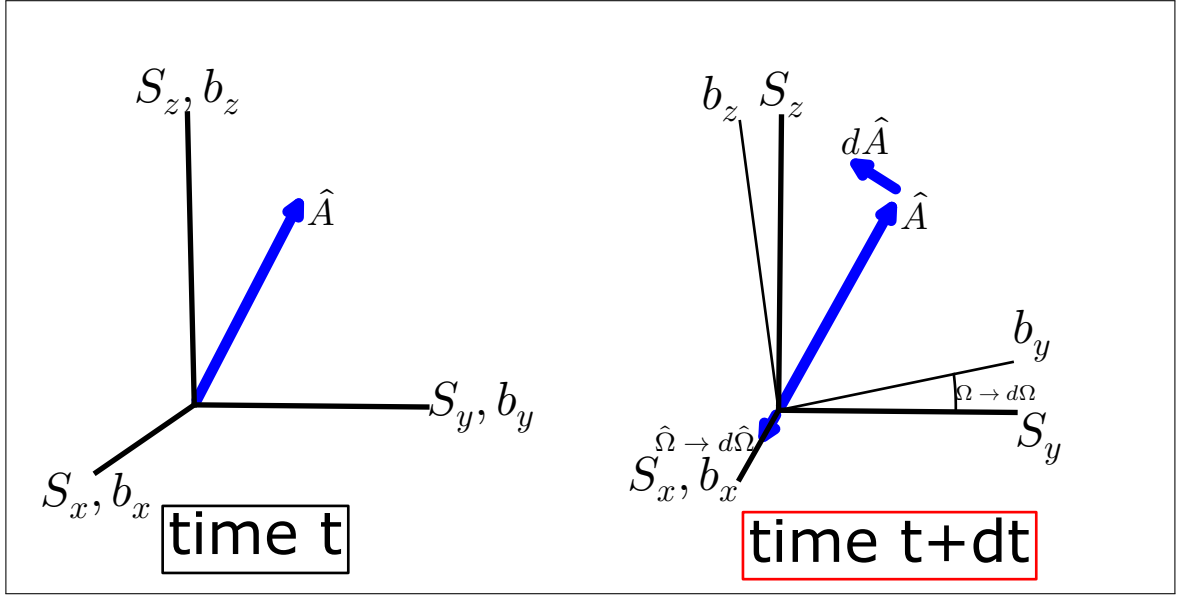
接著，在 t+dt 時間  $\vec{A} + d\vec{A}$  向量在 s 與 b frame 間的關係為

$$\left(\vec{A}'\right)_{b(t+dt)} = \underbrace{\Omega}_{\text{passive, r.h.}} \left(\vec{A}'\right)_{s(t+dt)}$$

$\Omega$  為 s, b frame 轉動矩陣 (passive r.h.)，此  $\Omega$  矩陣與上一段  $\vec{A}$  不變動的情況的  $\Omega$  矩陣完全相同，我們取 s 到 b frame 的轉動為微小量， $\Omega \rightarrow d\Omega$ ，上式依之前所述的原理可寫成

$$\left(\vec{A}'\right)_{b(t+dt)} = \underbrace{(1 + \epsilon)}_{\text{passive, r.h.}} \left(\vec{A}'\right)_{s(t+dt)}$$

Figure 2: Rate change of a vector observed in a inertial and non-inertial frame.



要強調這邊的  $\epsilon$  矩陣跟之前上一段的  $\epsilon$  矩陣是完全相同的，重新整理上式

$$(\vec{A}')_{s(t+dt)} = (\vec{A}')_{b(t+dt)} - \epsilon (\vec{A}')_{s(t+dt)} \quad (3)$$

接著我們用3式減去2式，

$$\begin{aligned} (\vec{A}')_{s(t+dt)} - (\vec{A})_{s(t)} &= (d\vec{A})_s, \text{ (the change in observable A in space frame)} \\ &= \left( (\vec{A}')_{b(t+dt)} - (\vec{A})_{b(t)} \right) - \epsilon (\vec{A}')_{s(t+dt)} \\ &= (d\vec{A})_{body} - \epsilon (\vec{A}')_{s(t+dt)} \\ &= (d\vec{A})_{body} - \epsilon (\vec{A} + d\vec{A})_{s(t+dt)} \end{aligned} \quad (4)$$

注意，由於 s frame 式恆定座標因此 s frame 不變動， $s(t+dt) = s(t)$ 。忽略高階項  $\epsilon (d\vec{A})_{s(t+dt)}$ ，重新整理成

$$(d\vec{A})_s = (d\vec{A})_b - \epsilon_{r.h.} (\vec{A})_{s(t)}$$

接下來我們只要記得我們的下標 b frame 總是在 t+dt 時的 frame，s frame 總是指 t 時間的 frame，我們將不再寫出 frame 所對應的時間。依之前所述原理代入 r.h.  $\epsilon$  的公式，並且利用向量外積

$$\begin{aligned} (d\vec{A})_s &= (d\vec{A})_b - \begin{bmatrix} 0 & \epsilon_3 \geq 0 & -\epsilon_2 \leq 0 \\ -\epsilon_3 & 0 & \epsilon_1 \geq 0 \\ \epsilon_2 & -\epsilon_1 & 0 \end{bmatrix} (\vec{A})_s \\ &= (d\vec{A})_b - (\vec{A})_s \times (d\vec{\Omega})_s \\ &= (d\vec{A})_b + (d\vec{\Omega})_s \times (\vec{A})_s \end{aligned} \quad (5)$$

因為這裡的  $\epsilon$  矩陣與上一段的  $\epsilon$  矩陣是一樣的，因此我們也可以用上之前轉動公式所推導的微小轉動矩陣  $\epsilon$  所對應的轉動向量  $(d\vec{\Omega})$ ，這樣我們就得到了 rate of change of a vector/observable in a rotating frame 公式

$$(d\vec{A})_s = (d\vec{A})_b + (d\vec{\Omega})_s \times (\vec{A})_s \quad (6)$$

我們連結了不同觀測座標觀測到的物理變化量，並且所用到都是已知的物理量  $(d\vec{\Omega})_s$  與  $(\vec{A})_s$ 。這邊要強調，因為這裡的  $\epsilon$  矩陣與上一段的  $\epsilon$  矩陣是一樣的，所以證明了  $d\vec{\Omega}$  所對應的向量就是 s frame 轉到 b frame 的角位移向量 (r.h.)，這樣強調的目的是，接下來  $d\vec{\Omega}$  所導出的貼體角速度，就是 s frame 轉到 b frame 的角速度，因為這跟一般我們對貼體角速度的定義與認知並不一樣，這裡再次強調，我們必須考慮了 t 時間 s 與 b frame 重合，才能得到這結果。接下來會說明，也是因為如此，我們才能利用貼體角速度來作座標軸的轉動追蹤，因此此觀念至關重要。另外要注意的是  $\vec{A}$  與  $d\vec{\Omega}$  是沿著 t 時間的 s frame 取的投影量。這邊值得一提的是，傳統公式大多寫成

$$(d\vec{A})_s = (d\vec{A})_b + (d\vec{\Omega})_b \times (\vec{A})_b$$

為什麼這邊的  $d\vec{\Omega}$  是沿著 body 取分量呢？注意5式中當我們將微小轉動矩陣  $\epsilon$  寫成向量  $d\vec{\Omega}$  時，我們並沒有侷限此向量是定義在哪一個觀測座標，因此沿著 s 或 b frame 取都是可以的。另外4式中我們忽略了高階項  $\epsilon(d\vec{A})_{s(t+dt)}$ ，因此留下了  $\epsilon(\vec{A})_{s(t+dt)} = \epsilon(\vec{A})_{s(t)}$ ，但其實我們也可以取另一個近似

$$-\epsilon(\vec{A} + d\vec{A})_{s(t+dt)} = -\epsilon(\vec{A})_{b(t+dt)}$$

若我們考慮 s 到 b frame 的轉動非常的微小，我們只是做了一個不一樣的忽略方法。這樣的話，我們就得到傳統公式。這裡也是 Goldstein[1] 裡面的附註說明  $(d\vec{\Omega}) \times (\vec{A})$  沿著 s 或 b frame 取分量都是可以的，只要外積矩陣運算後出來的結果是一樣的就可以，不過他並沒有給出背後的原因。

6式取微分即得到一般常見的形式，這邊我們取沿 s frame 給出的公式，方便我們之後作數值模擬

$$\left(\frac{d\vec{A}}{dt}\right)_s = \left(\frac{d\vec{A}}{dt}\right)_b + (\vec{\omega})_s \times (\vec{A})_s \quad (7)$$

其中  $(\vec{\omega})_s$  為  $\left(\frac{d\vec{\Omega}}{dt}\right)_s$ ，根據我們之前對  $d\vec{\Omega}$  的定義與強調，我們知道  $(\vec{\omega})_s$  即為 s frame 到 b frame 的瞬時角速度。

嚴謹的定義了  $d\vec{\Omega}$  與  $(\vec{\omega})_s$  後，我們接著需要討論如何從  $(\vec{\omega})_s$  求回相對應的轉動矩陣，這邊你會認為，不是將  $(\vec{\omega})_s$  的 xyz 分量帶入之前  $1 + \epsilon$  矩陣中的  $\epsilon_1\epsilon_2\epsilon_3$  就可以了嗎，這樣是不行的，因為從之前微小轉動的推導可以看出， $\epsilon_1\epsilon_2\epsilon_3$  是符合特定的 antisymmetric matrix properties 的，但任意的角速度向量  $(\vec{\omega})_s$  可不然。這邊我們利用 Calvin Klein parameter 來近似原本的轉動矩陣 (CK parameters 矩陣基本上與轉動公式 Rodrigues rotation formula 同源 [1])，這邊我們給他一個新代號  $CK(d\vec{\Omega})$ ，當然，接下來只要是矩陣運算我們都會寫上  $CK$  的主被動及左右手性質，方

便我們與圖對照與思考<sup>2</sup>。

$$CK(d\vec{\Omega})_{\text{r.h.}} = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & a^2 + c^2 - b^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & a^2 + d^2 - b^2 - c^2 \end{bmatrix},$$

$$\text{with } a = \cos\left(\frac{|d\vec{\Omega}|}{2}\right), b, c, d = \text{component of } d\hat{\Omega} \cdot \sin\left(\frac{|d\vec{\Omega}|}{2}\right)$$

現在，我們一再強調  $d\vec{\Omega}$  所對應的是 s frame 轉動到 b frame，因此我們建立的  $CK(d\vec{\Omega})$  矩陣具有以下的特性，根據圖2，

$$\begin{aligned} (\vec{A})_b &= CK(d\vec{\Omega})_{\text{passive, r.h.}} (\vec{A})_s \\ (\hat{S}_y)_s &= CK(d\vec{\Omega})_{\text{active, l.h.}} (\hat{b}_y)_s \end{aligned}$$

或者也可寫成

$$(\vec{A})_s = \underbrace{[CK(d\vec{\Omega})]^T}_{\text{active, l.h.}} (\vec{A})_b \quad (8)$$

$$(\hat{b}_y)_s = \underbrace{[CK(d\vec{\Omega})]^T}_{\text{active, r.h.}} (\hat{S}_y)_s \quad (9)$$

這代表了，若我們知道 s frame 到 b frame 的轉動角度，我們就可以求出 t+dt 時間的 xyz 軸在 t 時間 xyz 軸的投影量。若我們知道的是 s 到 b frame 的瞬時角速度  $(\vec{\omega})_s$ ，則可帶入  $CK((\vec{\omega})_s \cdot dt)^T$  來得到轉矩矩陣。以上兩式就是模擬或追蹤剛體的 body frame 的 x,y,z 軸轉動的基礎。

上述的微小轉動是只考慮 t 到 t+dt 時間內的變化，現在我們將用遞推並且 discrete 的方式，求出 body frame 在實驗者處在的靜止座標的變化。因此現在我們設定一個真正的靜止座標 Lab frame，見圖3，此為真正的觀測者所處在的 inertial frame。考慮任意一段微小轉動 t 到 t+dt，在 t 時刻時我們將剛體的 principle axes 設定為 S frame，再將 t+dt 時刻剛體的 principle axes 設定為 b frame，這樣代表 s frame 到 b frame 就是剛體 t 到 t+dt 的轉動，並且 s 到 b frame 的瞬時角速度也是剛體轉動的瞬時角速度。我們重新將9式寫成

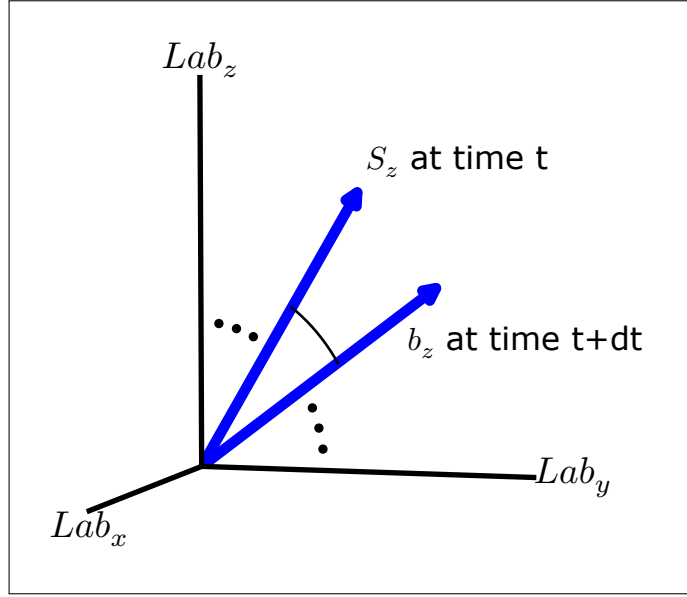
$$\hat{y}_s(t+dt) = \underbrace{[CK(\vec{\omega}_s(t) dt)]^T}_{\text{active, r.h.}} \hat{y}_s(t) \quad (10)$$

$\hat{y}_s(t)$  現在為 t 時間剛體特徵軸  $\hat{y}$  在 s frame(也是 t 時刻) 的投影，這代表  $\hat{y}_s(t)$  即為單位向量  $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$ 。現在我們再將上式寫成

$$\hat{y}_s(t_{i+1}) = \underbrace{[CK(\vec{\omega}_s(t_i) \Delta t)]^T}_{\text{active, r.h.}} \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \quad (11)$$

<sup>2</sup>力矩給出的角速度是遵守右手定則 (counterclockwise)，所以 CK 矩陣必須使用其 active counterclockwise sense 才能描述正確向量轉動，要小心，因大部分書上 (如 Goldstein) 給的公式都是 active clockwise (follow 左手定則) (舉例如書上的 Caley Klein parameter rotation matrix)，因此差一個負號，這裡我花了許多時間把文獻上所有公式轉成了正確的右手定則。

Figure 3: How to apply rate-of-change-of-a-vector equation to numerically simulate a true rotation. Here  $S_x S_y b_x b_y$  are not shown.



假設陀螺特徵軸在 lab frame 的起始位置已知  $\hat{x}\hat{y}\hat{z}_{lab}(t_0)$ ，初始貼體角速度已知  $\vec{\omega}_s(t_0)$ ，依照以往下標代表的是觀測的 frame。首先我們將 s frame 放在  $(\hat{x}\hat{y}\hat{z}_{lab}(t_0))$ ，這樣依照圖3及其所述原理，b frame 的軸就是我們要求的  $\hat{x}\hat{y}\hat{z}_{lab}(t_1)$ 。我們先看  $\hat{z}$  軸，11式告訴我們

$$\hat{z}_0(t_1) = \underbrace{[CK(\vec{\omega}_s(t_0)dt)]^T}_{\text{active, r.h.}} \times \hat{z}_0(t_0) = \underbrace{[CK(\vec{\omega}_s(t_0)dt)]^T}_{\text{active, r.h.}} \times [0 \ 0 \ 1]$$

其中  $\hat{z}_0(t_0)$ ， $\hat{z}_0(t_1)$  代表時間為  $t_0$  與  $t_1$  的  $\hat{z}$  軸在  $t_0$  時間的座標軸 (也就是 s frame) 的投影，因此  $\hat{z}_0(t_0)$  為單位向量  $[0 \ 0 \ 1]$ 。這樣子我們求得下一個  $\hat{z}$  軸的位置在  $t_0$  的投影，不過我們得轉回 lab frame，我們假設 lab frame 的  $xyz$  軸到陀螺初始位置  $\hat{x}\hat{y}\hat{z}_{lab}(t_0)$  的轉動向量是  $\vec{\Omega}_0$ ，這樣我們可以用  $\vec{\Omega}_0$  輕易的改變陀螺初始位置，運用上8式

$$\hat{z}_{lab}(t_1) = \underbrace{[CK(\vec{\Omega}_0)]^T}_{\text{passive, l.h.}} \times \hat{z}_0(t_1)$$

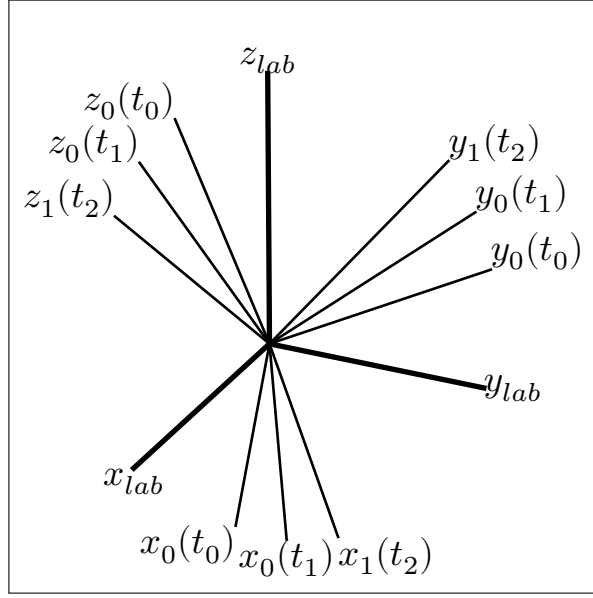
注意這邊矩陣就取被動含意，結合以上兩式得到

$$\begin{aligned} \hat{z}_{lab}(t_1) &= \underbrace{[CK(\vec{\Omega}_0)]^T}_{\text{passive, l.h.}} \underbrace{[CK(\vec{\omega}_s(t_0)dt)]^T}_{\text{active, r.h.}} \times \hat{z}_0(t_0) \\ &= \underbrace{[CK(\vec{\Omega}_0)]^T}_{\text{passive, l.h.}} \underbrace{[CK(\vec{\omega}_s(t_0)dt)]^T}_{\text{active, r.h.}} \times [0 \ 0 \ 1] \end{aligned}$$

這樣我們就從  $t_0$  時間得到  $t_1$  時間陀螺  $\hat{z}$  軸的位置。



Figure 4: Bobby 軸在每一分段  $t$  到  $t+dt$  的追蹤示意圖。



接著若我們知道  $\hat{z}_{lab}(t_i)$ ，再從尤拉方程數值法解出的  $\vec{\omega}_s(t_0, t_1, \dots, t_i)$  (接下來會說明)，我們同樣可以求得  $\hat{z}_{lab}(t_{i+1})$ ，首先用9式

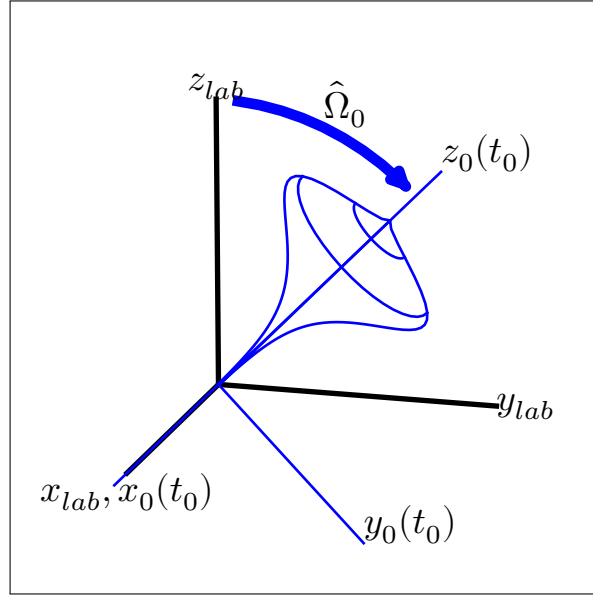
$$\begin{aligned}\hat{z}_i(t_{i+1}) &= \underbrace{[CK(\vec{\omega}_s(t_i) dt)]^T}_{\text{active, r.h.}} \times \hat{z}_i(t_i) \\ &= \underbrace{[CK(\vec{\omega}_s(t_i) dt)]^T}_{\text{active, r.h.}} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

再用8式轉回到 lab frame

$$\begin{aligned}\hat{z}_{lab}(t_{i+1}) &= \underbrace{[CK(lab \rightarrow t_i)]^T}_{\text{passive, l.h.}} \times \hat{z}_i(t_{i+1}) \\ &= \underbrace{[CK(\vec{\Omega}_0) \cdot CK(\vec{\omega}_s(t_0) dt) \cdot CK(\vec{\omega}_s(t_1) dt) \cdot \dots \cdot CK(\vec{\omega}_s(t_{i-1}) dt)]^T}_{\text{passive, l.h.}} \times \\ &\quad \hat{z}_i(t_{i+1}) \\ &= \underbrace{[CK(\vec{\Omega}_0) \cdot CK(\vec{\omega}_s(t_0) dt) \cdot CK(\vec{\omega}_s(t_1) dt) \cdot \dots \cdot CK(\vec{\omega}_s(t_{i-1}) dt)]^T}_{\text{passive, l.h.}} \times \\ &\quad \underbrace{[CK(\vec{\omega}_s(t_i) dt)]^T}_{\text{active, r.h.}} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\end{aligned}\tag{12}$$

這裡用上不同時間微小轉動矩陣的 commutative 性質  $(AB)C = A(BC)$ 。這樣我們就得到了所有時刻  $(t_0, t_1, \dots, t_{i+1})$  z 軸在 lab frame 位置的公式，同樣方法可求得 x, y 軸。可以看出上面所有 passive 的矩陣的作用只是再把坐標軸從 body frame 轉回到 lab frame。若我們再用上轉動矩陣相

Figure 5: 陀螺的初始值設定。此初始值設定對應的尤拉角初始值  $\mathfrak{r}$  以  $z - x' - z'$  為轉軸順序分別以右手定則轉動  $\phi, \theta, \varphi$  為  $\phi_0 = 0, \theta_0 = -|\hat{\Omega}_0|, \varphi_0 = 0$ 。在 Hasbun 教授的尤拉解法中需要此尤拉角初始值，來給出與我們方向餘弦法相同的初始條件。



乘即等於轉動向量相加的事實 [1]，上式可在寫成

$$\hat{z}_{lab}(t_{i+1}) == \left[ CK \left( \vec{\Omega}_0 + \sum_{m=0}^i \vec{\omega}_s(t_m) \right) \right]^T \times [0 \ 0 \ 1] \quad (13)$$

不過要注意，只有 CK 內包含的所有轉動向量都是要微小轉動向量此條公式才會成立。 $\mathfrak{r}$  在我們的 Python 程式中也包含有使用此公式的選項，讀者可以自由切換比較。用此公式的好處是，矩陣的相乘轉換成了矩陣的相加，在速度及記憶體佔用上此近似會有優勢。不過，程式預設是使用 12 公式，因  $\vec{\Omega}_0$  量值不小。 $\mathfrak{r}$  另外這邊我們就省略了矩陣的主被動左右手的標示，但注意，我們的 CK 矩陣是定義右手定則，遵守物理定律。這裡若不強調會產生問題，因為很多書上給的是 CK 是左手公式。並且，若要了解公式，主被動及左右手的標示是必須保留並且對照圖表才有辦法了解公式如何而來以及轉動方向是如何，而這也是大部分書上所缺乏的。這樣，我們就得到了剛體特徵軸的追蹤公式，並且只要知道"貼體"角速度  $\mathfrak{r}$  注意並不是角速度在 lab frame 的觀察值，當然如果知道這個，我就不用寫這麼多了。 $\mathfrak{r}$  就可以求得。同理可得  $\hat{x}_{lab}(t_{i+1}), \hat{y}_{lab}(t_{i+1})$ 。

若將  $\hat{x}_{lab}(t_{i+1}), \hat{y}_{lab}(t_{i+1}), \hat{z}_{lab}(t_{i+1})$  的三個行組成一矩陣 C，我們會發現 13 式即為導航書籍上常看到的方向餘弦遞推理論的姿態矩陣動力方程 [6]

$$C(t_{i+1}) = C(t_i) \exp\left(\int \omega dt\right) \text{ or} \quad (14)$$

$$\dot{C}(t_{i+1}) = C(t_i) \times \omega(t) \quad (15)$$

不過很少有書上會把此公式解釋的清楚，比如說，20 式中的 transpose 若不經過我們每一步轉動矩陣都紀錄並寫下主被動與左右手性質，實際應用的時候我們將不知道要套用 transpose 矩陣，

轉動向量的主動還是坐標系的被動特性以及轉動方向遵守是左手還是右手定則，這樣將造成極大的困擾。因此我們知道，只給出14公式只是一小步，離實作層面還有很大一段距離。此式中  $\exp(\int \omega dt)$  為 path order exponential，關於此，以及要如何證明  $\exp(\int \omega dt)$  是我們的 CK 矩陣，可以在 [2, Page 49] 找到，並注意連結與  $\sin, \cos$  的泰勒展開的關係。

以上為轉動坐標系中的向量變化量與恆定坐標系中的向量變化量與轉動坐標系的角速度的關係推導，不過我們發現我們必須要知道所有時刻的貼體角速度  $\vec{\omega}_s(t_0 \dots t_i)$ 。若是在慣性感測器的應用，由於 strap-down (捷聯式 or 綁附式) 慣性感測器測量得的就是貼體角速度，因此我們可以直接帶入我們的公式，這就是捷聯式慣性感測器姿態估測演算法的一種演算法，我們會在第四部分說明。接下來我們說明如何從尤拉方程數值求出  $\vec{\omega}_s(t_0 \dots t_i)$ ，並以此帶入姿態遞推公式來模擬陀螺運動。

將7式應用上一段 s 到 b frame，t 到 t+dt 的微小轉動，並且考慮  $\vec{A}$  為剛體角動量  $\vec{L}$ ，則我們得到

$$(\Gamma)_s = \left( \frac{d\vec{L}}{dt} \right)_s = \left( \frac{d\vec{L}}{dt} \right)_b + (\vec{\omega})_s \times (\vec{L})_s \quad (16)$$

這裡第一等號也用上牛頓定律。現在我們從6式知道  $(d\vec{\Omega})_s$  是 s 到 b frame 的角位移，而經由我們之前剛體特徵軸的設定，s 到 b frame 正是我們剛體特徵軸從 t 到 t+dt 的角位移，因此  $(\frac{d\vec{\Omega}}{dt})_s = (\vec{\omega})_s$  正是剛體的瞬時角速度 (沿著 t 時間 s frame 取分量)，接著，因為 s, b frame 都是沿著 body principle axes 而取，因此沿 s frame 的角動量  $(\vec{L})_s$  可以寫成

$$(\vec{L})_s = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \times (\vec{\omega})_s$$

並且沿著 b frame 的  $(\frac{d\vec{L}}{dt})_b$  項也可寫成

$$\left( \frac{d\vec{L}}{dt} \right)_b = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \times \frac{d(\vec{\omega})_b}{dt}$$

其中  $I_{xx}, I_{yy}, I_{zz}$  都不隨時間變動。再次注意  $(\Gamma)_s$  與  $(\vec{\omega})_s$  與  $(\vec{L})_s$  都是沿著 t 時刻的剛體特徵軸 (也就是 s frame) 取的投影，並不是 Lab frame 的投影，這點要特別注意，基本上這代表， $(\Gamma)_s$  是貼體的角動量！這裡大部分的書上都沒有給出恰當的原因，如果用 lab frame 的  $\Gamma$  那麼就無法成功的數值化喔<sup>3</sup>。這邊我們證明了16式最後那一項中的兩個  $\vec{\omega}$  是相同的<sup>4</sup>。代入  $\vec{L}$  並展開16式，我們就得到所謂的尤拉公式 (Euler's equation)

$$\begin{aligned} \Gamma_x(t) &= I_x \dot{\omega}_x + (I_z - I_y) \omega_y(t) \omega_z(t) \\ \Gamma_y(t) &= I_y \dot{\omega}_y + (I_x - I_z) \omega_x \omega_z \\ \Gamma_z(t) &= I_z \dot{\omega}_z + (I_x - I_y) \omega_x \omega_y \end{aligned} \quad (17)$$

其中  $\vec{\omega}_b = [\omega_x \ \omega_y \ \omega_z]$ 。注意  $\vec{\Gamma}$  及  $\vec{\omega}$  的 x,y,z 分量都是沿著 t 時刻的剛體特徵軸 s frame 取的分量，這點必須要強調。之後數值模擬的時候這點是非常重要的。

<sup>3</sup>注意因為 s frame 會持續的改變所以  $(\vec{\Gamma})_s$  不可取  $(\vec{\Gamma})_{lab}$  的值，同理  $(\vec{\omega})_s$  也不是  $(\vec{\omega})_{lab}$ ，兩者都必須經過轉換從 lab 轉到 t 時刻 s frame。

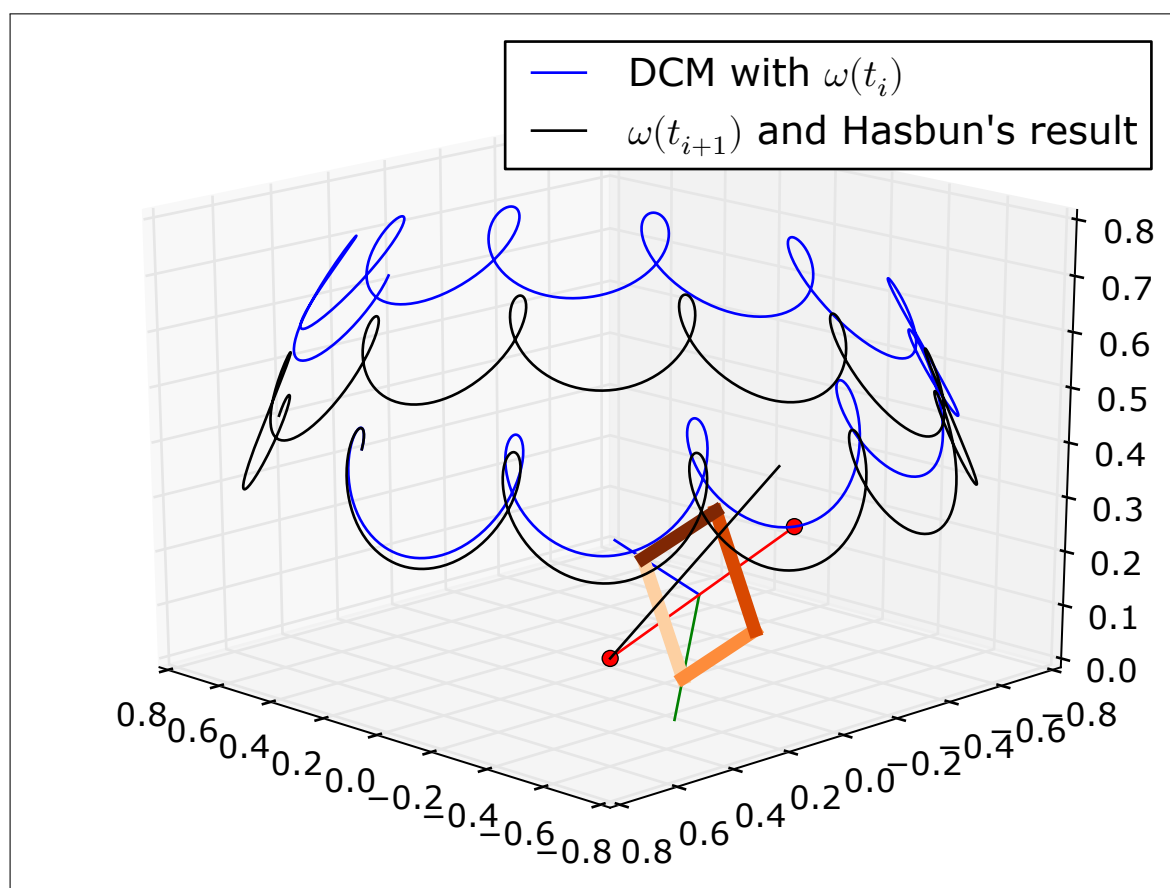
<sup>4</sup>但我們必須強調，任意情況下，角速度  $(\vec{\omega})$  在 body 轉動座標下的投影並不是 body 座標上觀察到的角速度！這是很常見的錯誤，這裡我們是有條件的考慮 t 到 t+dt 時刻的 t 時刻 s, b 座標重和。

接下來應用上陀螺，若考慮陀螺的條件  $I_x = I_y \neq I_z$ ，17式可寫成

$$\frac{d}{dt} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 & -\frac{I_z - I_y}{I_x} \omega_z & 0 \\ -\frac{I_x - I_z}{I_y} \omega_z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} \frac{\Gamma_x}{I_x} \\ \frac{\Gamma_y}{I_y} \\ \frac{\Gamma_z}{I_z} \end{bmatrix} \quad (18)$$

如之前所強調，右邊所有項都是在時間為  $t$  時刻的  $s$  frame 取得值，也因此以上的微分方程組可以用普通數值由拉法或四階 Ruge Kutta 求出左側  $\omega_{x,y,z}(t+dt)$ ，也就是從  $\vec{\omega}_s(t)$  求得  $\vec{\omega}_s(t+dt)$ 。不過對於任意的剛體轉動系統，只要能從17式右側  $\vec{\omega}_s(t)$  求得左側  $\vec{\omega}_s(t+dt)$ ，都還是能夠適用接下來的模擬方法，也因此這裡描述的方法是具有任意一般性的，可以應用在任何的剛體轉動。有不少的數值方法可以解一般的非線性一階 ODE 尤拉方程 [3]。這裡我以 Ruge Kutta 四階法求解上式，來得到  $\vec{\omega}_s(t_0 \cdots t_i)$ ，並且寫成 python 程式，程式將在下一章介紹。

Figure 6: 陀螺的對稱軸的單位向量的模擬軌跡圖。以  $\omega(t_i)$  或  $\omega(t_{i+1})$  轉動向量來近似  $t_i$  到  $t_{i+1}$  時間的轉動的模擬結果，並與 Hasbun 教授的正確解比較。 $\omega(t_{i+1})$  的結果與 Hasbun 教授的結果在此圖形中幾乎重合，這代表以  $\omega(t_{i+1})$  來近似比  $\omega(t_i)$  好很多。



# 1 以 $t_{i+1}$ 時間的轉動向量來近似 $t_i$ 到 $t_{i+1}$ 的轉動

以上10式中以  $CK(\vec{\omega}_s(t_i)dt)$  來近似  $t_i$  到  $t_{i+1}$  的轉動事實上還不夠好，圖6顯示，以  $\omega(t_i)$  轉動向量近似的結果與 Hasbun 的正確結果偏差不少。這邊我提出以  $CK(\vec{\omega}_s(t_{i+1})dt)$  來近似  $t_i$  到  $t_{i+1}$  的轉動，圖6顯示模擬結果幾乎與 Hasbun 的正確解重合，至少此圖表上無法看出任何差別。為什麼有如此大的影響，以下我也嘗試提供物理解釋。這裡我們暫時假設  $\vec{\omega}_s(t_i)dt = \vec{\Omega}_s(t_i)$ ，我們知道轉動向量在  $t_{i+1}$  跟  $t_i$  時刻在 body frame 中的向量值一般不會一樣，也就是  $\vec{\Omega}_{i+1}(t_{i+1}) \neq \vec{\Omega}_i(t_i)$ ，這代表從  $t_i$  到  $t_{i+1}$  時，轉動向量在 body 座標上有變化，也因此我們不能夠單只考慮陀螺轉了  $\vec{\Omega}_s(t_i)$  而已，此額外轉動向量的變化在  $t_i$  時 s frame 的向量值為  $\Omega_{i+1}(t_{i+1}) - \Omega_i(t_i) = \Omega_i(t_i) + d\Omega_i(dt) - \Omega_i(t_i) = d\Omega_s(dt)$ ，也是一個轉動向量，所以 space 空間中總共的轉動可以考慮成兩步，第一步轉  $\Omega_s(t_i)$ ，第二步轉  $d\Omega_s(dt)$ ，寫成轉動矩陣

$$CK(\Omega_s(t_i)) \times CK(d\Omega_s(dt)) = CK(\Omega_s(t_i) + d\Omega_s(dt)) = CK(\Omega_{i+1}(t_{i+1})) \quad (19)$$

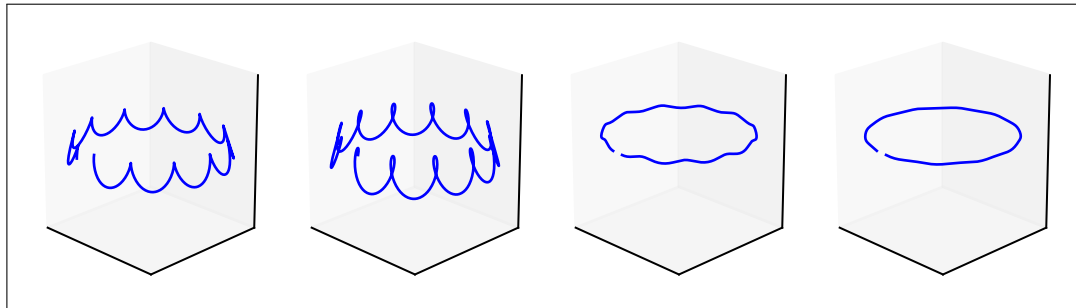
這代表我們只要考慮陀螺從  $t$  到  $t+dt$  的時候是轉了  $\Omega_s(t+dt)$  而不只是  $\Omega_s(t)$ ，因此考慮  $\Omega_s(t+dt)$  我們就更準確的近似了這個轉動，以下的 Python 程式模擬會證明，考慮了  $\Omega_s(t+dt)$  給出的結果比  $\Omega_s(t)$  好非常多。若如此考慮則12式變成

$$\hat{z}_{lab}(t_{i+1}) = \left[ CK(\vec{\Omega}_0) \cdot CK(\vec{\omega}_s(t_1)dt) \cdot CK(\vec{\omega}_s(t_2)dt) \cdot \dots \cdot CK(\vec{\omega}_s(t_{i+1})dt) \right]^T \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad (20)$$

此公式即為我 Python 程式中使用的 DCM 遞推的公式。相同方法可求得另外兩軸 x,y 的運動。

若需要更精確的近似，書 [6, Page 301, Equation 10.24] 中也給出轉動向量的近似公式，不過此公式頗為複雜，因此這邊先嘗試比較以我們  $t_{i+1}$  方法與解析解的不同。下面就將上述方法寫成 python 程式，以尤拉方程數值解出貼體角速度，接著用遞推公式13畫圖模擬其 xyz 軸運動。這邊劃出四種陀螺的經典運動。

Figure 7:



總結而言，我們上述所使用的方法為，數值解轉動座標的貼體角速度尤拉方程，得到下一個新的貼體角速度作 DCM 遞推。我們接下來就跟另外兩種方法比較，並且也說明每種方法適合的應用場合。其中一種是傳統的數值解空間座標的尤拉角尤拉方程。此方法雖然最傳統，但帶入尤拉角的尤拉方程式呈現高度的非線性化，因此數值計算上的誤差會最大。並且，一般若是外力力矩項隨時間而變化，或帶入尤拉角的力矩項過於複雜，則此方式便不適用，因為此步驟還得人為降階成一階 ODE 聯立方程組，才能作數值計算。若力矩項過於複雜甚至降階步驟困難，這時候就適合用我們的 DCM 方法。另一個比較方法跟數值解空間座標尤拉角尤拉方程相

似，當解出尤拉角隨時間的數值解之後，我們將尤拉角的解轉成貼體角速度，然後再以 DCM 方法積分出所有的姿態。此方法適用的情形就是，當貼體角速度已知，而力矩為未知，此方法可以直接以 DCM 方法遞推求出姿態。此方法即為角速度感測器的姿態解算演算原理，因此適用於感測器的姿態估測上。此三種方法的比較在圖10有詳細的圖示解釋。

即時解尤拉方程然後作 DCM 姿態遞推的好處是，不用事先知道力矩項，因此會比尤拉角尤拉方程方法更適用於力矩會隨時間變動或力矩為未知的情況，舉例來說，我們可以用在機器人的姿態控制上面，又或者，可以用在有用到剛體轉動的遊戲物理引擎上面。這些狀況都是尤拉角尤拉方程中的力矩項無法事先預知，因此無法事先解其尤拉角尤拉方程。又或者，若是力矩項太過複雜，如無人機的空氣動力，也適合使用我們這邊所介紹的即時解尤拉方程並作姿態遞推，只要有角加速度計可以測量角加速度就可以。我們這邊所介紹的陀螺，是一個應用的例子，由於陀螺的轉動速度是每秒 20 轉，因此轉動速度很高，對於姿態遞推的精度要求算很高，因此陀螺適合當作驗證姿態遞推演算的非常好的例子，若陀螺能算的準，其他轉動速度不高的情況將會非常準。陀螺的情況我們可以比喻為，把機器人或無人機丟入龍捲風裡面，並且要作姿態控制。

## Part III

# 方向餘弦演算法 Python 程式碼之說明

程式經過幾次升級修改後，目前已經以物件導向方式編寫完成。主要是環繞著一個類別〔class〕名為 RigidBodyObject 來發展。此類別的宣告及其方法〔演算法的主程式〕被包在一個套件 module 裡，此套件為 RGCordTransV11.py 檔，使用前先 import RGCordTransV11。以下為如何使用程式的一個簡單範例，以及其輸出的陀螺動畫，我們將此範例存成一個 test.py 檔，裡面的內容為

```
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as p3
import matplotlib.animation as animation
import RBPlotFunction as RBPlot
import RGCordTransV11 as RG
b1 = RG.RigidBodyObject()
#set para here
b1.tn=1.45
b1.GenerateDependentVariables()
#set case here(after dep var is created)
b1.setcase(2)
b1.HasbunEulerEquationODEsolve()
b1.eulerW2bodyW()
b1.EulerDCMiter()
#%%% Plot figure top animation
fig2 = plt.figure(2,figsize=(4,2.66))
ax4 = p3.Axes3D(fig2)
ax4.view_init(elev=20, azim=125)
ax4.set_xlim3d([-0.8, 0.8])
ax4.set_ylim3d([-0.8, 0.8])
```

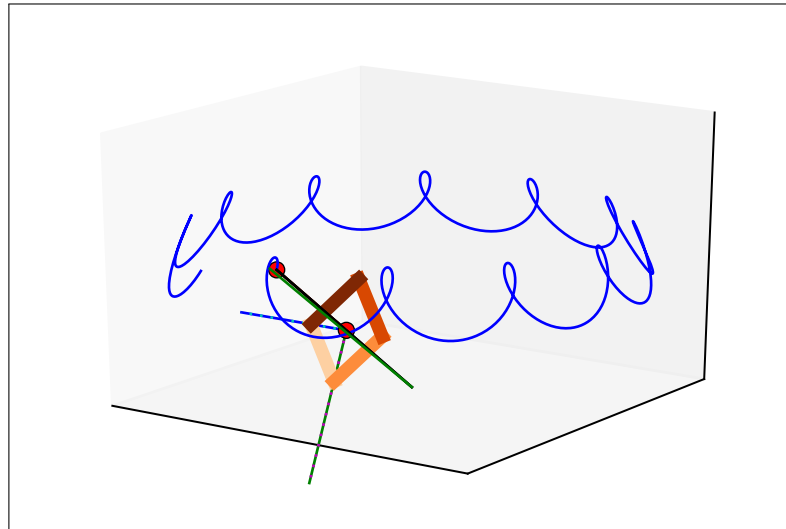
```

ax4.set_zlim3d([0, 0.8])
RBPlot.xthAnimationFrame(b1,ax4,plt)
ax4.set_xticks([])
ax4.set_yticks([])
ax4.set_zticks([])
line_ani = animation.FuncAnimation(fig2, RBPlot.update_line, list(range(1,b.N,15)),
fargs=(b.baxes+b.polylines+b.baxes_hasbun,
b.lineL,b.lineW,b.cordvec,b.L_plot,
b.w_b_norm),interval=10, blit=False,repeat=False)
plt.show()

```

模擬程式畫出剛體特徵軸的  $x, y, z$  軸單位向量，我們以一個正方形來代表剛體，正方形的中心點定在  $z$  軸的二分之一處，由於 Python 的 Matplotlib 三維繪圖程式庫 mplot3d 並沒有太多的 3D 繪圖支援，因此這邊只是以一個簡單的示意的正方形來代表陀螺，之後我也考慮將模擬結果輸出至專門做 3D rendering 的繪圖程式語言，如 OpenGL，來做更精美的陀螺外觀。 $z$  軸單位向量的頂點對時間的軌跡圖，也就是 locus，是以藍色的曲線表示。Hasbun 教授的尤拉角法給出的  $z$  軸單位向量的 locus，是以黑色的曲線表示。另外剛體的總角動量的單位向量以黑色線段表示，剛體的角速度除以角速度初始值量值的 normalized 向量以綠色線段表示。

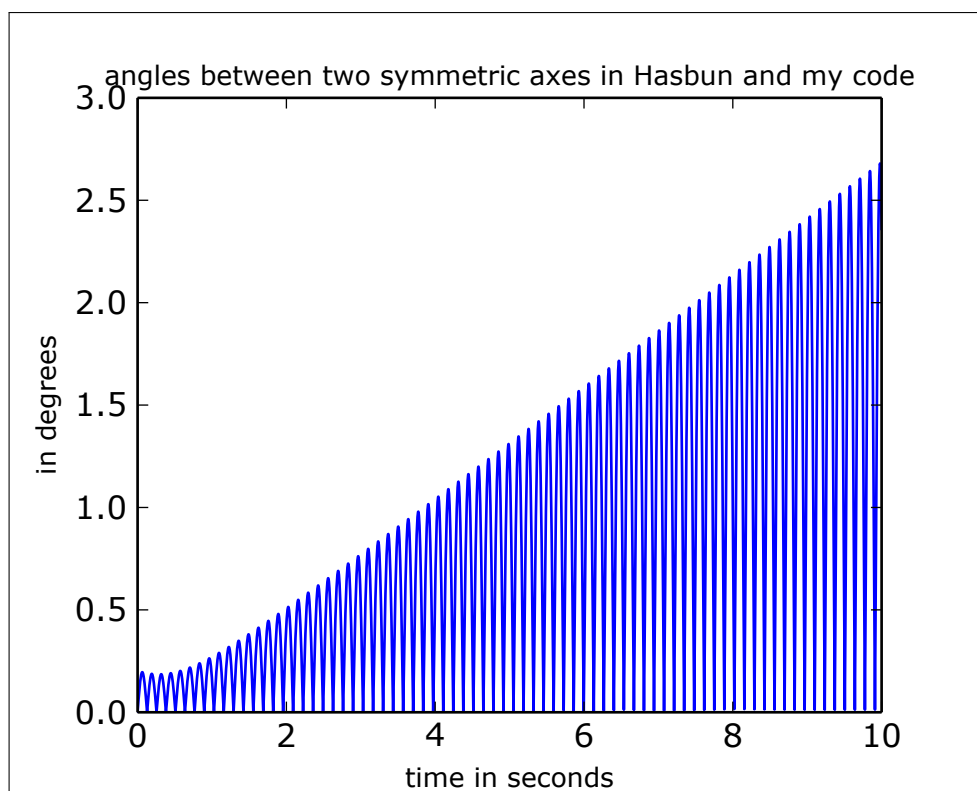
Figure 8: 模擬條件 (SI units):  $I=0.002$ ;  $I_s=0.0008$ ;  $g=9.8$ ;  $M=1$ ;  $\text{arm}=.04$ ;  $\text{spin freq}=20\text{ Hz}$ ; Initial angle from vertical 54.57 degree; Sampling rate 2000 Hz.



這裡提供的 Python 程式碼可以適用於任意的陀螺狀態，因此不會遇到以尤拉角表達的方法所遇到在九十度角所遭遇的奇異點問題。這個程式碼也包含了 Hasbun 教授在他的著作中 [5] 提供的 Matlab 程式碼轉成的 Python 程式碼，由於我希望能夠作系統性的比較，所以花了一些時間將 Hasbun 教授的 Matlab 程式碼重寫成 Python 程式碼。因此在我的 Python 程式碼中也包含了 Hasbun 教授的程式，此程式函數叫做 HasbunMatlabTop()，其定義在 top.py 檔案中。Hasbun 教授的程式碼是尤拉角所寫出來的，所以會有奇異點問題， $\theta$  在九十度時會產生 NaN。不過，尤拉角法解出的是解析解，所以只要不發散，結果會是絕對正確的。由於姿態的方向餘弦遞推是一種數值近似，因此會有數值誤差 (numerical drift)，也因此我們有跟解析解比較的必要。當然，方向餘弦法適用任意的剛體轉動方程，可直接套用，而尤拉角法還得要看轉動尤拉

方程的複雜程度，必須要能寫成一階 ODE 微分方程組，才能做數值積分，因此各有其優缺點。我們這邊比較方向餘弦法與 Hasbun 教授尤拉法所算出來的陀螺對稱軸 (body z 軸) 之間的夾角角度，見圖??。可以看出來模擬結果此夾角角度在任意時刻都小於 0.1 degree。這代表我們的方法是可行的。補充一下，Hasbun 教授的尤拉角的定義也是依照  $z - x' - z'$  軸依序依照右手定則轉動  $\phi \cdot \theta \cdot \varphi$  角。

Figure 9: 模擬條件 (SI units):  $I=0.002$ ;  $I_s=0.0008$ ;  $g=9.8$ ;  $M=1$ ;  $\text{arm}=0.04$ ;  $\text{spin freq}=20$  Hz; Initial angle from vertical 54.57 degree; Sampling rate 2500 Hz.



使用  $\omega(t + dt)$  來做  $t$  時間的轉動近似讓我們的程式碼變得非常簡單，比書中 [6, Page 301, Equation 10.24] 給出的轉動向量近似公式簡單太多了。簡單的程式碼代表之後在做速度優化，或轉寫成速度快的編譯式語言上，會簡單非常多。主程式叫做 `RG CordTrans.py`。我使用的 Python 版本為 Python 2.7.5 - Anaconda 1.8.0 (32-bit)，NumPy 1.7.1，SciPy 0.13.0，Matplotlib 1.3.1。不同版本的 Python (如 Python 3.x) 會有一些指令的不同需要做小修改，可以聯絡我我可以幫做修正。RG CordTrans 主程式可調整的參數為

```
M = 1 #mass in kg
#R = 0.025 #radius of disk in m
#L = 0.01 #width of disk in m
arm = 0.04 #level arm of Center of mass in m
Iy = 0.002 #moment of inertial, substitute disk formula if needed Iy=0.25*M*R**2 +
1/12*M*L**2 +M*arm**2
Iz = 0.0008 #Iz = 0.5*M*R**2
Ix = 0.002
```



```

g = 9.8 #gravity m/s^2
F = np.array([0,0,-M*g]) #gravity f
freq = 20 #top rotating angular speed in hertz
tn=1.2*1 # end of simulation time, take longer time if tn > 10 seconds
t0=0 # start of time zero
samplerate = 2000 # rate of iteration in Hz
N = (tn-t0)*samplerate #number of steps
h = (tn-t0)/N #time step interval

```

主程式以四階 Rugge Kutta ( `topRK(wi,torquei)` ) 直接數值解陀螺尤拉方程 ( `topEOM(wi,torquei)` ) 得到貼體角速度，接著用遞推公式12來得到陀螺特徵軸 xyz 軸的單位向量 `cordvec` 對時間的變化。`cordvec` 陣列的維度 `shape=(N,3,7)`。其中維度為 7 的方向的首要三個陣列就是 xyz 軸。

DCM 遞推的程式為以下五行

```

for i in range(1,N+1):
    w[i,:]=topRK(w[i-1:],Tau_body_temp)
    TrackMat_temp = np.dot(TrackMat_temp,CK(w[i,:]*h))    # wi here,
    for j in range(3):
        cordvec[i,:,j]=np.dot(TrackMat_temp,np.eye(3)[j,:])

```

注意力矩我們是用沿著 body 軸的 `Tau_body_temp`。

以下列出程式所用到的兩個 module，分別為 `RGCordTransV11.py` 檔與 `RBPlotFunc.py` 檔中所包含的程式

- `RGCordTransV11.py`
  - Class `RigidBodyObject`
    - \* Method
      - user level
      - `GenerateDependentVariable`
      - `setcase`

## Part IV

# 在綁縛式慣性感測器姿態演算的應用

上面的五行遞推 DCM 的程式可以獨立出來應用在感測器的姿態估測上。由於 DCM 是考慮貼體角速度  $\vec{\omega}$ ，因此主要應用在綁縛式慣性感測元件 ( `strap-down IMU` ) 上，此類包含範圍涵蓋陀螺儀角速度感測器，微機電角速度感測器，等諸如此類的元件。只要把元件讀到的貼體角速度資料輸出，然後正確的輸入到我們這邊的 DCM 遞推程式，就可以得到物件轉動的姿態。這邊大部分的書籍都無法從原理說起，都只有提供公式，而且也很少詳細的說明公式如何使用。不過要補充的是，我這邊只有轉動部分，並沒有平移部分的演算法，不過轉動姿態是姿態估測的基礎，平移的演算是相對容易很多的。並且，這邊也沒有噪音過濾的功能，因為任何的量測儀器都會有噪音，因此在輸入 DCM 遞推前，必須先經過過濾。一般常見的過濾方法是卡爾曼濾波，這裡不作噪音濾波的討論。

## Part V

# 補充

陀螺等周速運動 (Figure ??) 的初始值條件如何計算呢? 等周速的條件在 Goldstein 第二版 5-77 式給出

$$Mgl = \dot{\phi} (I_3\omega_3 - I_1\dot{\phi}\cos\theta_0) \quad (21)$$

· 不過此式是由尤拉角 (euler angles) 給出, 但我們需要的是 angular velocity along body 的初始值, 因此我們必須轉換尤拉角到 angular velocity along body, 方法如下。上式中  $\omega_3$  即為我們的  $(\omega_z)_b$ , 比如說是 20 Hz,  $\theta_0$  即為我們之前的 orien 向量所定, 由上式可求出兩組  $\dot{\phi}(t_0)$ 。另外尤拉角跟 angular velocity along body 的關係式在書 [1, Chapter 4, Equation 125] 給出

$$(\omega_x)_b = \dot{\phi}\sin\theta\sin\psi + \dot{\theta}\cos\psi \quad (22)$$

$$(\omega_y)_b = \dot{\phi}\sin\theta\cos\psi - \dot{\theta}\sin\psi \quad (23)$$

$$(\omega_z)_b = \dot{\phi}\cos\theta + \dot{\psi} \quad (24)$$

知道  $\dot{\phi}(t_0)$ 、 $\theta_0$ 、 $\dot{\theta}_0 = 0$  及  $\psi_0$ , 用一二條後就可得到  $\omega_{xb}(t_0)$  與  $\omega_{yb}(t_0)$ , 這樣我們就得到 angular velocity along body 的初始值。因為  $\dot{\phi}(t_0)$  有兩組, 因此解出的貼體角速度也會有兩組, 兩組的物理意義分別如下, 一種情況是 fast top, 這個狀況相當於重力的影響遠小於總角動量  $L$ , 因此這個特別的例子基本上相當於忽略重力, 而陀螺基本上會像一個 free top 一樣進行 precession。另一種狀況是 slow top, 也就是上面模擬結果中第四種的狀況, 這裡提供的 python 程式所有情況都可以模擬。另外一個特殊的情況是在 fast top 的情形下, 如果初始值  $\theta_0 = 0$ , 也就是陀螺  $z$  軸的起始狀態是垂直於水平面的, 這樣的話陀螺幾乎會像靜止不動一樣, 我們也叫這情況做 sleeping top。

**Remark 1** 要陀螺具有 *Precession and Nutation* 的動作,  $L/\Delta L$  必須要大, 如果  $L$  小於  $\Delta L$ , 則只會有陀螺質量受重力影響往下倒下的運動 (不過這對檢查程式有沒有錯誤很有幫助!), 理想上  $L$  至少要大於  $\Delta L$ , 最好  $L$  大大於  $\Delta L$ 。化成數值上的比較: 這代表

$$L \gg \Delta L \Rightarrow I \cdot 2\pi f \gg \vec{r} \Delta t \Rightarrow I \cdot 2\pi f \gg \vec{r} \times \vec{F} \cdot 1/f \Rightarrow f \gg \sqrt{\frac{arm \cdot Mg \cdot \sin(\theta)}{2\pi I \cdot G}} \quad (25)$$

where  $\theta$  is gyro's tilt angle and  $G$  is moment of inertial geometry factor. 考慮  $\Delta t$  的量級大約是陀螺轉幾圈的時間 (characteristic time), 量級上約是  $\sim 1/f$ , 若假設  $arm$  是 10 cm,  $M = 1\text{kg}$ ,  $g = 10\text{ m/s}^2$ ,  $I = 0.5M(0.05)^2$ , 則  $f$  最少要 10 Hertz 以上。因此我們將以這些參數比較  $f = 1, 10, 50$  Hertz 所給出的陀螺運動。

Python code: <https://drive.google.com/file/d/0B96HmLH-SQVmM1dvYlFiQm9ESGM/edit?usp=sharing>

3D animation: <http://tinypic.com/r/10cw9yf/8>

This document is prepared with Scientific Workplace 5.0 and typeset with Tex Live 2013 (Xelatex).

Blogpost: <http://ppt.cc/yS2->

If you feel like supporting this work, you can [donate here: http://ppt.cc/-L9p](http://ppt.cc/-L9p).

## References

[1] Herbert Goldstein, *Classical Mechanics*. Addison Wesley, Massachusetts, 2nd Edition, 1980

- [2] David Tong, *Classical Dynamics University of Cambridge Part II Mathematical Tripos*. Cambridge UK, 2004-2005, (Course note, available on the web)
- [3] [Matlab online documentation - Ordinary differential equations.](#), Matlab R2014a
- [4] 徐小明钟万勰·刚体动力学的四元数表示及保辛积分·《应用数学和力学》2014·35(1): 111
- [5] Javier E. Hasbun, *Classical Mechanics with Matlab Applications*. Jones and Bartlett Publishers, London UK, 2009
- [6] D.H. Titterton and J.L. Weston, *Strapdown inertial navigation technology*, Peter Peregrinus Ltd., London UK, 1997
- [7] David Baraff, [Physically Based Modeling - Rigid Body Simulation](#), Pixar Animation Studios notes
- [8] Harvey Gould and Jan Tobochnik and Wolfgang Christian, *An Introduction to Computer Simulation Methods Third Edition*, Addison-Wesley, 2006, [draft available on comPADRE](#)
- [9] Dennis Rapaport, *The Art of Molecular Dynamics Simulation 2nd Edition*, Cambridge University Press, 2004

Figure 10: 比較 DCM 法與 Hasbun 教授尤拉角法的流程圖。

