基于 Matlab 的捷联惯导算法设计及仿真[1]

严恭敏
西北工业大学航海学院，西安 （710072）
E-mail：yangongmin@163.com

摘 要：根据圆锥误差补偿算法和划船误差补偿算法的研究成果，考虑到实际捷联惯导算法仿真程序编写的方便性，总结了一些与捷联惯导更新算法有关的函数的计算公式。对圆锥误差补偿算法和捷联惯导算法进行了仿真，仿真结果和理论分析结论吻合。在附录中给出了 Matlab 的 m 文件源程序代码，具有一定的参考价值。

关键词：捷联惯导；四元数；等效旋转矢量；Matlab；算法；仿真
中图分类号：V249.3

1. 引言

在捷联惯导系统中采用数学平台，姿态更新解算是捷联惯导系统算法的核心部分，由于四元数法算的优良特性，它在工程实际中经常被采用。为了减小姿态计算的不可交换性误差，前人研究并建立了等效旋转矢量方程，高精度姿态更新解算的研究主要集中在等效旋转矢量方程的求解上，在圆锥运动环境下，许多研究者提出并完善了圆锥误差补偿算法。基于圆锥误差补偿算法和划船误差补偿算法的等效原理，可将圆锥误差补偿算法移植到划船误差补偿算法中去，从而减少了划船误差推导的繁琐过程。上述研究都已经比较成熟[1-6]，本文根据这些研究结果，并考虑到实际仿真程序编写的方便性，总结了一些与捷联惯导算法有关的函数的计算公式或步骤，其中更详细的推导过程可见参考文献[7,8]。最后，对圆锥误差补偿算法和捷联惯导算法进行了仿真。附录中给出了Matlab的m文件源程序代码具有一定的参考价值。

2. 捷联惯导算法

文中选取东-北-天（E-N-U）地理坐标系为导航坐标系，记为 n 系；捷联惯组坐标系记为 b 系。

2.1 相关函数

（1）四元数的共轭与乘积。四元数 $\boldsymbol{q}$ 可表示为 $\boldsymbol{q}=q_0+\boldsymbol{q}_v=q_0+q_1i+q_2j+q_3k$。用 $\boldsymbol{q}^*$ 表示 $\boldsymbol{q}$ 的共轭四元数；$\boldsymbol{q}=\boldsymbol{q}_1\boldsymbol{q}_2$ 表示四元数 $\boldsymbol{q}$ 是四元数 $\boldsymbol{q}_1$ 与 $\boldsymbol{q}_2$ 的乘积，四元数相乘是不可交换的。

（2）四元数与向量相乘。一般情况下利用矩阵进行坐标变换，如关系式 $\boldsymbol{v}^n=\boldsymbol{C}_b^n\boldsymbol{v}^b$，同样利用四元数也可以表示坐标变换。设变换四元数 $\boldsymbol{q}_b^n$ 与变换矩阵 $\boldsymbol{C}_b^n$ 相对应，则定义变换四元数 $\boldsymbol{q}_b^n$ 和向量 $\boldsymbol{v}^b$ 的乘法，记为 $\boldsymbol{v}^n=\boldsymbol{q}_b^n\boldsymbol{v}^b$，它由以下规则实现：首先，将向量扩展成四元数，令 $\boldsymbol{q}^b=0+\boldsymbol{v}^b$；其次，做四元数乘法，令 $\boldsymbol{q}^n=\boldsymbol{q}_b^n\boldsymbol{q}^b\boldsymbol{q}_b^{n*}$；最后，从四元数 $\boldsymbol{q}^n$ 中提取向量，有 $\boldsymbol{v}^n=\boldsymbol{q}_v^n$。

（3）等效旋转矢量与四元数之间的转换。等效旋转矢量 $\boldsymbol{v}$ 转化为四元数 $\boldsymbol{q}$ 的公式为

[1] 本课题得到水下信息处理与控制国家级重点实验室基金（9140C230206070C2306）资助。

$$\boldsymbol{q}=\boldsymbol{F}_{v\to q}(\boldsymbol{v})=\cos(|\boldsymbol{v}|/2)+\frac{\boldsymbol{v}}{|\boldsymbol{v}|}\sin(|\boldsymbol{v}|/2) \tag{1}$$

其中 $\boldsymbol{F}_{v\to q}(\bullet)$ 表示从等效旋转矢量转换到四元数的函数，容易看出有 $\boldsymbol{q}^{*}=\boldsymbol{F}_{v\to q}(-\boldsymbol{v})$ 成立。假设 $|\boldsymbol{v}|$ 是小量，则从 $\boldsymbol{q}$ 中可以求出 $\boldsymbol{v}$，首先令 $v_{n2}=|\boldsymbol{v}|/2=\arccos(q_0)$，再求得

$$\boldsymbol{v}=\boldsymbol{F}_{q\to v}(\boldsymbol{q})=\frac{2\cdot v_{n2}}{\sin(v_{n2})}\boldsymbol{q}_v \tag{2}$$

其中 $\boldsymbol{F}_{q\to v}(\bullet)$ 表示从四元数转换到等效旋转矢量的函数。

（4）姿态向量 $\boldsymbol{A}$ 与姿态四元数 $\boldsymbol{q}_b^n$ 之间的转换。记由俯仰角 θ、横滚角 γ 和航向角 ψ 组成的向量 $\boldsymbol{A}=[\theta\quad\gamma\quad\psi]^{\mathrm{T}}$ 为姿态向量，通过姿态矩阵 $\boldsymbol{C}_b^n$ 作为过渡，容易实现 $\boldsymbol{A}$ 与 $\boldsymbol{q}_b^n$ 之间的转换，此处不作详细分析。

2.2 地球模型

通常给出的地球椭球模型参数为长半轴 R_e 和扁率 f，由椭球几何学容易得到其它参数，在惯性导航算法中经常用到，它们是偏心率 $e=\sqrt{2f-f^2}$、短半轴 $R_p=(1-f)R_e$、第二偏心率 $ep=\sqrt{R_e^2+R_p^2}/R_p$、子午圈半径 $R_M=R_e(1-e^2)/(1-e^2\sin^2 L)^{3/2}$、卯酉圈半径 $R_N=R_e/(1-e^2\sin^2 L)^{1/2}$。另外，重力加速度 g 和地球自转角速率 ω_{ie} 也是惯性导航解算的必备参数，在 GRS80 椭球模型中，正常重力公式为

$$g=g_0(1+5.27094\times10^{-3}\sin^2 L+2.32718\times10^{-5}\sin^4 L)-3.086\times10^{-6}h \tag{3}$$

其中 $g_0=9.7803267714$ m/s^2，h 为海拔高度，而 $\omega_{ie}=7.2921151467\times10^{-5}$ rad/s。记地球自转角速度矢量在导航坐标系投影为 $\boldsymbol{\omega}_{ie}^n$，惯导速度 $\boldsymbol{v}^n$ 引起导航坐标系转动速度为 $\boldsymbol{\omega}_{en}^n$，则

$$\boldsymbol{\omega}_{ie}^n=[0\quad\omega_{ie}\cos L\quad\omega_{ie}\sin L]^{\mathrm{T}} \tag{4a}$$

$$\boldsymbol{\omega}_{en}^n=\left[-v_N^n/(R_M+h)\quad v_E^n/(R_N+h)\quad v_E^n\tan L/(R_N+h)\right]^{\mathrm{T}} \tag{4b}$$

并且记

$$\boldsymbol{\omega}_{in}^n=\boldsymbol{\omega}_{ie}^n+\boldsymbol{\omega}_{en}^n \tag{4c}$$

2.3 圆锥误差与划船误差补偿算法

假设陀螺和加速度计均为等周期采样（采样周期 h），圆锥误差与划船误差补偿周期均为 T，且 $T=n\cdot h$，n 为子样数。再假设在第 m 个补偿周期中，陀螺角增量和加速度计比力速度增量采样输出分别为 $\Delta\boldsymbol{\theta}_m(i)$ 和 $\Delta\boldsymbol{v}_m(i)$（$i=1,2,3...n$），并令采样总角增量 $\Delta\boldsymbol{\theta}_m=\sum_{i=1}^{n}\Delta\boldsymbol{\theta}_m(i)$ 和采样总比力速度增量 $\Delta\boldsymbol{v}_m=\sum_{i=1}^{n}\Delta\boldsymbol{v}_m(i)$，则考虑圆锥误差补偿后的等效旋转矢量计算公式为

$$\boldsymbol{\Phi}_m=\Delta\boldsymbol{\theta}_m+[\sum_{i=1}^{n-1}k_i\Delta\boldsymbol{\theta}_m(i)]\times\Delta\boldsymbol{\theta}_m(n) \tag{5}$$

上式中第二项为圆锥误差补偿量，系数 k_i 的选择见表 1。

根据划船误差补偿算法与圆锥误差补偿算法的对偶原理，则考虑划船误差补偿后的比力速度增量计算公式为

$$\begin{aligned}\Delta\boldsymbol{v}_{sfm} &= \Delta\boldsymbol{v}_m + 1/2(\Delta\boldsymbol{\theta}_m \times \Delta\boldsymbol{v}_m) \\ &+ \{[\sum_{i=1}^{n-1} k_i \Delta\boldsymbol{\theta}_m(i)] \times \Delta\boldsymbol{v}_m(n) + [\sum_{i=1}^{n-1} k_i \Delta\boldsymbol{v}_m(i)] \times \Delta\boldsymbol{\theta}_m(n)\}\end{aligned} \tag{6}$$

上式中第二项为旋转速度增量，第三项为划船误差补偿量。

表 1　圆锥误差补偿算法系数

子样数 n	k1	k2	k3	k4
2	2/3	-	-	-
3	9/20	27/20	-	-
4	54/105	92/105	214/105	-
5	250/504	525/504	650/504	1375/504

2.4 捷联惯导更新算法

捷联惯导更新的基本思想是，将前一时刻的导航参数（姿态、速度和位置）作为初值，利用前一时刻至当前时刻的惯性器件采样输出，解算当前时刻的导航参数，作为下一时刻捷联惯导解算的初值，如此反复不断。惯性器件采样经过 2.3 小节的误差补偿后获得等效旋转矢量 $\boldsymbol{\Phi}_m$ 和比力速度增量 $\Delta\boldsymbol{v}_{sfm}$，再经过以下三步骤便可实现捷联惯导更新，这里直接给出计算公式。

（1）速度更新算法

$$\boldsymbol{v}_m^n = \boldsymbol{v}_{m-1}^n + \boldsymbol{F}_{v\to q}(-\boldsymbol{\omega}_{inm-1}^n T_m/2)\boldsymbol{q}_{bm-1}^n \Delta\boldsymbol{v}_{sfm} + [\boldsymbol{g}_{m-1} - (2\boldsymbol{\omega}_{iem-1}^n + \boldsymbol{\omega}_{enm-1}^n) \times \boldsymbol{v}_{m-1}^n]T_m \tag{7}$$

（2）位置更新算法

$$L_m = L_{m-1} + T_m v_{Nm,m-1}^n / R_M \tag{8a}$$

$$\lambda_m = \lambda_{m-1} + T_m \sec L_{m-1} v_{Em,m-1}^n / R_N \tag{8b}$$

$$h_m = h_{m-1} + T_m v_{Um,m-1}^n \tag{8c}$$

（3）姿态更新算法

$$\boldsymbol{q}_{bm}^n = \boldsymbol{q}_{bm-1}^n \boldsymbol{F}_{v\to q}(\boldsymbol{\Phi}_m - \boldsymbol{q}_{bm-1}^{n\,*} \boldsymbol{\omega}_{inm-1}^n T_m) \tag{9}$$

3. 仿真与分析

根据第 2 节的分析，将各函数和算法编制成 Matlab 的 m 文件，文件名及简要功能如表 2 所示，详细的源程序代码参见附录，文中还着重对圆锥误差补偿算法和捷联惯导更新算法进行了仿真。

表 2　Matlab 文件功能说明

文件	功能	文件	功能
glvs.m	全局变量	a2quat.m	由姿态向量求姿态四元数
earth.m	有关地球参数计算	q2att.m	由姿态四元数求姿态向量
qconj.m	四元数共轭	cnscl.m	圆锥误差和划船误差补偿
qmul.m	四元数相乘	sins.m	捷联惯导更新算法
qmulv.m	四元数乘向量	test1.m	圆锥误差补偿算法仿真
rv2q.m	等效旋转转换为四元数	test2.m	捷联惯导算法算法仿真
q2rv.m	四元数转换为等效旋转		

3.1 圆锥误差补偿算法仿真

如图 1，动坐标系 b 系相对参考坐标系 r 系绕 $o_r z_r$ 轴作圆锥运动，半锥角为 α，锥运动频率为 ω，即：b 系与 r 系坐标原点重合，$o_b x_b$ 轴以 $o_r x_r$ 轴为对称中心，在 $x_r o_r z_r$ 平面内作幅值为 α 的振动；$o_b y_b$ 轴以 $o_r y_r$ 轴为中心，在 $y_r o_r z_r$ 平面内幅值为 α 的振动；$o_b z_b$ 轴在以 $o_r z_r$ 轴为中心的圆锥面上运动。动坐标系 b 相对参考坐标系 r 的方位关系还可用姿态四元数描述为

$$\boldsymbol{q}_b^r(t) = \left[\cos(\alpha/2) \quad \sin(\alpha/2)\cos\omega t \quad \sin(\alpha/2)\sin\omega t \quad 0\right]^{\mathrm{T}} \tag{10}$$

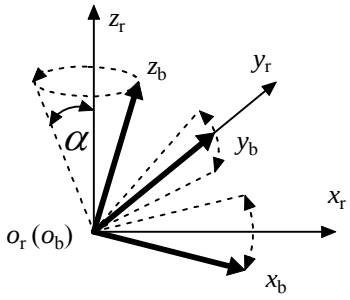

图 1　b 系相对 r 系作圆锥运动

两个相邻采样时刻之间的角增量输出为

$$\Delta\boldsymbol{\theta}_{k+1} = \begin{bmatrix} -2\sin\alpha\sin(\omega h/2)\sin[\omega(t_k + h/2)] \\ 2\sin\alpha\sin(\omega h/2)\cos[\omega(t_k + h/2)] \\ -2(\omega h)\sin^2(\alpha/2) \end{bmatrix} \tag{11}$$

其中，$h = t_{k+1} - t_k$，$\Delta\boldsymbol{\theta}_{k+1}$ 为从 t_k 时刻到 t_{k+1} 时刻的角增量，即 t_{k+1} 时刻采样获得的角增量输出。理论上可以证明，圆锥误差补偿算法在一个补偿周期 T 内的算法漂移误差为

$$\varepsilon = \frac{n \times n!}{2^{n+1}\prod_{k=1}^{n+1}(2k-1)}\alpha^2(\omega h)^{2n+1} \tag{12}$$

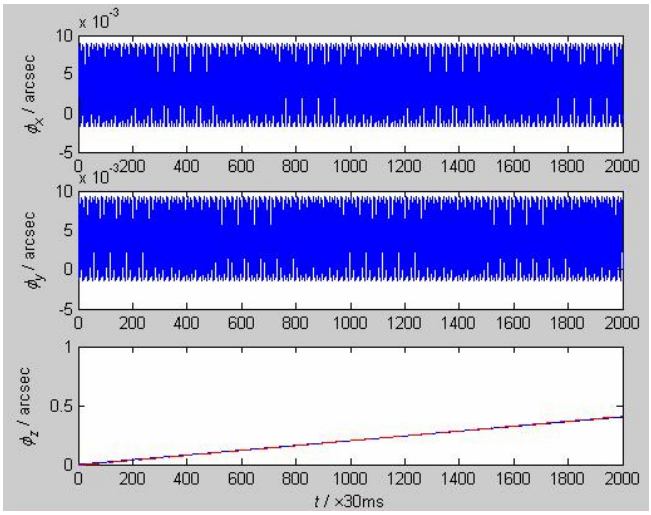

图 2　圆锥误差补偿仿真

当α=1°、ω=5Hz、h=10ms、n=3时，仿真一分钟的姿态误差如图2所示。其中蓝色曲线为圆锥误差补偿姿态与真实姿态之间的误差，可见在$o_r x_r$轴和$o_r y_r$轴有微小振荡误差，而$o_r z_r$轴为圆锥漂移误差。最下面小图中红色曲线为利用公式（12）计算的理论圆锥漂移误差，它与蓝色曲线非常接近。

此外，当ω=1Hz、h=10ms，而α和n取不同值时，进行了一系列的圆锥漂移误差仿真和理论漂移误差计算，结果如表 3 所示。从表中可以看出，只有当半锥角α很小时，高子样数的仿真误差和理论误差才比较接近，而当α较大时，高子样数的仿真误差和理论误差之间相差倍数很大，甚至在仿真误差中出现高子样补偿效果不如低子样的现象。产生这种现象的原因是，由于在圆锥漂移补偿算法推导过程中作出了半锥角α是小角度的假设，还须指出的是，在推导过程中还假设圆锥频率和采样周期乘积$\omega \cdot h$必须为小量。因此，在实际中必须结合应用环境选择合适的圆锥误差补偿算法，而并非子样数越高计算精度就越高，但是，如果考虑到实际陀螺的精度（如陀螺漂移 1×10^{-4} °/h），即使α为 10° 时，选用 3 子样的计算精度也足够了，或者提高采样频率有利于降低计算误差。

表 3　一分钟圆锥漂移误差仿真和理论计算（单位："）

α		1 子样	2 子样	3 子样	4 子样	5 子样
1"	仿真误差	6.013e-7	4.745e-10	4.016e-13	3.522e-16	9.558e-19
	理论误差	6.012e-7	4.747e-10	4.013e-13	3.523e-16	3.161e-19
1′	仿真误差	2.164e-3	1.708e-6	1.444e-9	1.612e-12	**1.623e-12**
	理论误差	2.164e-3	1.709e-6	1.445e-9	1.268e-12	**1.138e-15**
1°	仿真误差	7.790	6.148e-3	4.596e-6	**4.480e-6**	**2.103e-5**
	理论误差	7.792	6.152e-3	5.205e-6	**4.566e-9**	**4.097e-12**
10°	仿真误差	771.242	0.596	**-5.455e-3**	**4.416e-2**	**2.075e-2**
	理论误差	779.272	0.615	**5.205e-4**	**4.566e-7**	**4.097e-10**

3.2 捷联惯导算法仿真

传统上用姿态矩阵表示的计算数学平台$\boldsymbol{C}_b^{n'}$与真实数学平台$\boldsymbol{C}_b^n$之间的关系为

$$\boldsymbol{C}_b^{n'} = \boldsymbol{C}_n^{n'}\boldsymbol{C}_b^n \approx (\boldsymbol{I} - \boldsymbol{\varphi}\times)\boldsymbol{C}_b^n \tag{13}$$

其中$\boldsymbol{\varphi}$为平台误差角，它表示从n'系到n系的小转动角向量（可以看成是等效旋转矢量）。假设变换矩阵$\boldsymbol{C}_b^{n'}$、$\boldsymbol{C}_n^{n'}$、$\boldsymbol{C}_b^n$分别与变换四元数$\boldsymbol{q}_b^{n'}$、$\boldsymbol{q}_n^{n'}$、$\boldsymbol{q}_n^n$相对应，则有$\boldsymbol{q}_b^{n'} = \boldsymbol{q}_n^{n'}\boldsymbol{q}_b^n$，并且利用矩阵与四元数之间的关系可以证明$\boldsymbol{q}_n^{n'} = \boldsymbol{F}_{v\to q}(-\boldsymbol{\varphi})$成立，所以有

$$\boldsymbol{q}_b^{n'} = \boldsymbol{F}_{v\to q}(-\boldsymbol{\varphi})\boldsymbol{q}_b^n \tag{14a}$$

$$\boldsymbol{F}_{v\to q}(-\boldsymbol{\varphi}) = \boldsymbol{q}_b^{n'}\boldsymbol{q}_b^{n^*} \Leftrightarrow \boldsymbol{\varphi} = -\boldsymbol{F}_{q\to v}(\boldsymbol{q}_b^{n'}\boldsymbol{q}_b^{n^*}) \tag{14b}$$

在仿真时，通过（14）式能够方便地进行平台误差角处理，与传统矩阵描述平台误差角相比，四元数描述方法的优点是不必再作正交化处理。

对静态导航进行了仿真，初始平台误差角分别设置为0.5′、0.5′和3′，仿真一小时导航参数结果如图3所示，其中速度误差和位置误差小图中，红色曲线为高度通道的误差，它们是发散的，而水平通道（包括水平平台误差、水平速度和经纬度误差）呈现振荡趋势，方位平台误差变化比较小。仿真结果符合惯导系统误差理论分析的结论。

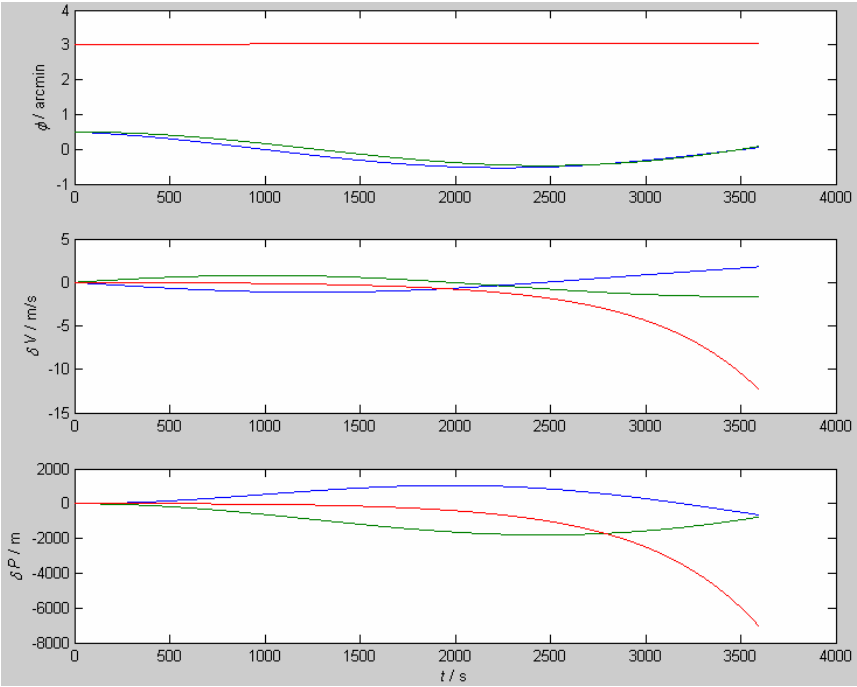

图 3　捷联惯导算法仿真

4. 小结

总结了一些与捷联惯导更新算法有关的函数的计算公式，对圆锥误差补偿算法和捷联惯导算法进行了仿真。在圆锥误差补偿算法中发现，只有当半锥角很小时，高子样数的仿真误差和理论误差才比较接近，否则相差倍数很大，因此在高精度应用场合，必须通过提高采样频率来降低计算误差。捷联惯导静态导航仿真结果与惯导系统误差理论分析结论相吻合，证明了所设计算法的正确性。

参考文献

[1]Bortz J E, A new mathematical formulation for strapdown inertial navigation[J], IEEE Transactions on Aerospace and Electronic Systems,1971,7(1):61-66.
[2]Miller R B, A new strapdown attitude algorithm[J], Journal of Guidance, Control and Dynamics, 1983,6(4):287-291.
[3]Roscoe K M, Equivalency between strapdown inertial navigation coning and sculling integrals / algorithms[J]. Journal of Guidance, Control and Dynamics，2001，24(2)：201-205.
[4]秦永元. 惯性导航[M]. 北京：科学出版社，2006.
[5]张玲霞，陈明，曹晓青. 基于对偶性原理捷联惯导划船误差补偿优化算法[J]. 中国惯性技术学报，2003,11(5):33-38.
[6]潘献飞,吴文启,吴美平. 考虑机抖激光陀螺信号滤波特性的圆锥算法修正[J]. 中国惯性技术学报，2007,15(3):259-264.
[7]严恭敏. 捷联惯导算法及车载组和导航系统研究[D]. 西安：西北工业大学硕士论文，2004,3.
[8]严恭敏. 车载自主定位定向系统研究[D]. 西安：西北工业大学博士论文，2006,5.

Algorithm design and simulation for SINS based on Matlab

Yan Gongmin
College of Marine，Northwestern Polytechnical University，Xi'an （710072）

Abstract

Based on the research achievements of coning error compensation and sculling error compensation algorithm in strapdown inertial navigation system(SINS), and considering the convenience of simulation software program, some calculation formulas about the SINS updating algorithm are summarized in this paper. Coning error compensation algorithm simulation and SINS algorithm simulation are carried out and the simulation results are consistent with the theory analysis. Simulation source codes written in Matlab's m file are present in the appendix, and provide valuable references for SINS software designer.

Keywords：strapdown inertial navigation system；quaternion；equivalent rotate vector；Matlab；algorithm；simulation

作者简介：严恭敏（1977-），男，福建建瓯人，2006 年 9 月毕业于西北工业大学自动化学院，获导航、制导与控制专业博士学位，现在西北工业大学航海学院兵器科学与技术博士后流动站工作，主要从事陆用定位定向系统、水下航行器自主导航技术与组合导航系统理论研究。

附录　Matlab仿真程序

% glvs.m 全局变量

```
%全局变量
    global glv
    glv.Re = 6378160;                   %地球半径
    glv.f = 1/298.3;                    %地球扁率
    glv.e = sqrt(2*glv.f-glv.f^2); glv.e2 = glv.e^2; %地球椭圆度等其它几何参数
    glv.Rp = (1-glv.f)*glv.Re;
    glv.ep = sqrt(glv.Re^2+glv.Rp^2)/glv.Rp;    glv.ep2 = glv.ep^2;
    glv.wie = 7.2921151467e-5;          %地球自转角速率
    glv.g0 = 9.7803267714;              %重力加速度
    glv.mg = 1.0e-3*glv.g0;             %毫重力加速度
    glv.ug = 1.0e-6*glv.g0;             %微重力加速度
    glv.ppm = 1.0e-6;                   %百万分之一
    glv.deg = pi/180;                   %角度
    glv.min = glv.deg/60;               %角分
    glv.sec = glv.min/60;               %角秒
    glv.hur = 3600;                     %小时
    glv.dph = glv.deg/glv.hur;          %度/小时
    glv.mil = 2*pi/6000;                %密位
    glv.nm = 1853;                      %海里
    glv.kn = glv.nm/glv.hur;            %节
    glv.cs = [                          %圆锥和划船误差补偿系数
        2./3,       0,          0,          0
        9./20,      27./20,     0,          0
        54./105,    92./105,    214./105,   0
        250./504,   525./504,   650./504,   1375./504 ];
```

% earth.m 有关地球参数计算

```
function [wnie, wnen, RMh, RNh, gn] = earth(pos, vn)
global glv
    sl=sin(pos(1)); cl=cos(pos(1)); tl=sl/cl; sl2=sl*sl; sl4=sl2*sl2;
    wnie = glv.wie*[0; cl; sl];
    sq = 1-glv.e2*sl2; sq2 = sqrt(sq);
    RMh = glv.Re*(1-glv.e2)/sq/sq2+pos(3);
    RNh = glv.Re/sq2+pos(3);
    wnen = [-vn(2)/RMh; vn(1)/RNh; vn(1)/RNh*tl];
    g = glv.g0*(1+5.27094e-3*sl2+2.32718e-5*sl4)-3.086e-6*pos(3); % grs80
    gn = [0;0;-g];
```

% qconj.m 四元数共轭

```
function qo = qconj(qi)
    qo = [qi(1); -qi(2:4)];
```

% qmul.m 四元数相乘

```
function q = qmul(q1,q2)
    q = [ q1(1) * q2(1) - q1(2) * q2(2) - q1(3) * q2(3) - q1(4) * q2(4);
          q1(1) * q2(2) + q1(2) * q2(1) + q1(3) * q2(4) - q1(4) * q2(3);
          q1(1) * q2(3) + q1(3) * q2(1) + q1(4) * q2(2) - q1(2) * q2(4);
          q1(1) * q2(4) + q1(4) * q2(1) + q1(2) * q2(3) - q1(3) * q2(2) ];
```

% qmulv.m 四元数乘向量

```
function vo = qmulv(q, vi)
    qi = [0;vi];
    qo = qmul(qmul(q,qi),qconj(q));
    vo = qo(2:4,1);
```

% rv2q.m 等效旋转转换为四元数

```
function q = rv2q(rv)
    norm = sqrt(rv'*rv);
    if norm>1.e-20
        f = sin(norm/2)/(norm/2);
    else
        f = 1;
    end
    q = [cos(norm/2); f/2*rv];
```

% q2rv.m 四元数转换为等效旋转

```
function v = q2rv(q)
    n2 = acos(q(1));
    if n2>1e-20
        k = 2*n2/sin(n2);
    else
        k = 2;
    end
    v = k*q(2:4);
```

% a2quat.m 由姿态向量求姿态四元数

```
function q = a2quat(att)
    % 先求姿态矩阵
    si = sin(att(1)); ci = cos(att(1)); sj = sin(att(2)); cj = cos(att(2));
    sk = sin(att(3)); ck = cos(att(3));
    M = [ cj*ck+si*sj*sk, ci*sk,    sj*ck-si*cj*sk;
         -cj*sk+si*sj*ck, ci*ck, -sj*sk-si*cj*ck;
         -ci*sj,          si,        ci*cj ];
    % 再求四元数
```

```
    q = [                           sqrt(abs(1.0 + M(1,1) + M(2,2) + M(3,3)))/2.0;
        sign(M(3,2)-M(2,3)) * sqrt(abs(1.0 + M(1,1) - M(2,2) - M(3,3)))/2.0;
        sign(M(1,3)-M(3,1)) * sqrt(abs(1.0 - M(1,1) + M(2,2) - M(3,3)))/2.0;
        sign(M(2,1)-M(1,2)) * sqrt(abs(1.0 - M(1,1) - M(2,2) + M(3,3)))/2.0 ];
```

% q2att.m 由姿态四元数求姿态向量

```
function att = q2att(qnb)
    % 先求姿态矩阵
    q11 = qnb(1)*qnb(1); q12 = qnb(1)*qnb(2); q13 = qnb(1)*qnb(3); q14 = qnb(1)*qnb(4);
    q22 = qnb(2)*qnb(2); q23 = qnb(2)*qnb(3); q24 = qnb(2)*qnb(4);
    q33 = qnb(3)*qnb(3); q34 = qnb(3)*qnb(4);
    q44 = qnb(4)*qnb(4);
    M = [ q11+q22-q33-q44,   2*(q23-q14),        2*(q24+q13);
          2*(q23+q14),         q11-q22+q33-q44, 2*(q34-q12);
          2*(q24-q13),         2*(q34+q12),       q11-q22-q33+q44 ];
    % 再求姿态
    att = [asin(M(3,2)); atan(-M(3,1)/M(3,3)); atan(M(1,2)/M(2,2))];
    if M(3,3) < 0
        if(att(2) < 0)   att(2) = att(2)+pi;
        else                 att(2) = att(2)-pi;
        end
    end
    if( M(2,2) > 0)
        if(att(3) < 0)       att(3) = att(3)+pi*2;
        end
    else      att(3) = att(3)+pi;
    end
```

% cnscl.m 圆锥误差和划船误差补偿

```
function [phim, dvbm] = cnscl(wm, vm)
global glv
    n = size(wm,2);
    cm = [0;0;0]; sm = [0;0;0];
    for k=1:n-1
        cm = cm + glv.cs(n-1,k)*wm(:,k); %准备圆锥误差补偿
    end
    wmm = sum(wm,2);
    phim = wmm + cross(cm,wm(:,n));
    if nargin==2
        for k=1:n-1
            sm = sm + glv.cs(n-1,k)*vm(:,k); %准备划船误差补偿
        end
        vmm = sum(vm,2);
        dvbm = vmm + 1.0/2*cross(wmm,vmm) + (cross(cm,vm(:,n))+cross(sm,wm(:,n)));
    end
```

% sins.m 捷联惯导更新

```
function [qnb, vn, pos] = sins(qnb_1, vn_1, pos_1, wm, vm, ts)
    tss = ts*size(wm,2);
    [phim,dvbm] = cnscl(wm,vm);
    [wnie,wnen,rmh,rnh,gn] = earth(pos_1,vn_1);
    wnin = wnie+wnen;
    vn = vn_1 + qmulv(rv2q(-wnin*(1.0/2*tss)),qmulv(qnb_1,dvbm)) + (gn-cross(wnie+wnin,vn_1))*tss;
    pos = pos_1 + tss*[vn_1(2)/rmh;vn_1(1)/(rnh*cos(pos_1(1)));vn_1(3)];
    qnb = qmul(qnb_1, rv2q(phim - qmulv(qconj(qnb_1),wnin*tss)));
```

% test1.m 圆锥误差补偿仿真

```
glvs
clear err dtheta
a = 1*glv.deg;   %半锥角
w = 2*pi*5;       %锥运动频率
```

```
h = 0.01;           %采样时间
n = 3;              %子样数
len = fix(1*60/h);  %仿真时间长度
t = 0;              %初始姿态四元数
q = [cos(a/2); sin(a/2)*cos(w*t); sin(a/2)*sin(w*t); 0];   q1 = q;
for k=n:n:len
    %  姿态真值
    t = k*h;
    q = [cos(a/2); sin(a/2)*cos(w*t); sin(a/2)*sin(w*t); 0];
    %  锥运动解算值
    for m=1:n
        t = (k-n+m-1)*h;
        dtheta(:,m) = [-2*sin(a)*sin(w*h/2)*sin(w*(t+h/2));
            2*sin(a)*sin(w*h/2)*cos(w*(t+h/2));
            -2*w*h*(sin(a/2))^2 ];   %角增量
    end
    phi = cnscl(dtheta);            %圆锥误差补偿
    q1 = qmul(q1,rv2q(phi));        %姿态更新
    err(k/n,:) = -q2rv(qmul(q1,qconj(q)))'; %求姿态误差
end
figure
subplot(3,1,1), plot(err(:,1)/glv.sec), ylabel('\it\phi_x\rm / arcsec');
subplot(3,1,2), plot(err(:,2)/glv.sec), ylabel('\it\phi_y\rm / arcsec');
subplot(3,1,3), plot(err(:,3)/glv.sec), ylabel('\it\phi_z\rm / arcsec');
str = sprintf('\\itt\\rm / \\times%.0fms', n*h*1000); xlabel(str);
%  理论圆锥误差
k2 = 1;
for k=1:n+1
    k2 = k2*(2*k-1);
end
epsilon = a^2*(w*h)^(2*n+1) * n*factorial(n) / (2^(n+1)*k2);
hold on, plot([0,length(err)],[0,epsilon]*length(err)/glv.sec,'r--')
```

% test2.m 捷联惯导更新仿真

```
glvs
clear errphi errvn errpos
ts=0.1;             %采样周期
att0=[0;0;0]*glv.deg; vn0=[0;0;0]; pos0=[0*glv.deg;0;0];    %初始值设置
qnb0 = a2quat(att0);
vn = vn0; pos = pos0;
wnie = glv.wie*[0; cos(pos0(1)); sin(pos0(1))];
g = glv.g0*(1+5.27094e-3*sin(pos0(1))^2+2.32718e-5*sin(pos0(1))^4)-3.086e-6*pos0(3);
wbib = qmulv(qconj(qnb0),wnie);
fb = qmulv(qconj(qnb0),[0;0;g]);
wm=wbib*ts; vm=fb*ts;    %静止时角增量和比力增量
qnb = qmul(rv2q(-[0.5;0.5;3]*glv.min),qnb0); %加入姿态误差
kk = 1;
for k=1:3600*10
    [qnb,vn,pos]=sins(qnb,vn,pos,wm,vm,ts);
    if mod(k,10)==0
        errphi(kk,:) = -q2rv(qmul(qnb,qconj(qnb0)))'; %求姿态误差
        errvn(kk,:) = (vn-vn0)';
        errpos(kk,:) = (pos-pos0)';
        kk = kk+1;
    end
end
figure,
subplot(3,1,1),plot(errphi/glv.min), ylabel('\it\phi\rm / arcmin');
subplot(3,1,2),plot(errvn), ylabel('\it\delta V\rm / m/s');
subplot(3,1,3),plot([errpos(:,1:2)*glv.Re,errpos(:,3)]), ylabel('\it\delta P\rm / m');
xlabel('\itt\rm / s');
```