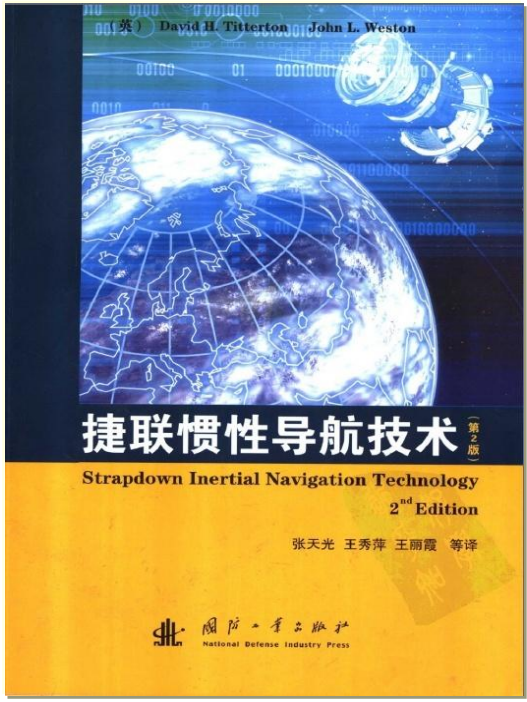
惯性导航基础

- 第一部分 惯性导航概述（相关概念）
- 第二部分 捷联惯性导航系统的基本原理
- **第三部分 捷联惯导系统计算**
- 第四部分 惯导系统的对准
- 第五部分 惯导系统的误差分析
- 第六部分 惯性系统的测试与标定

第三部分

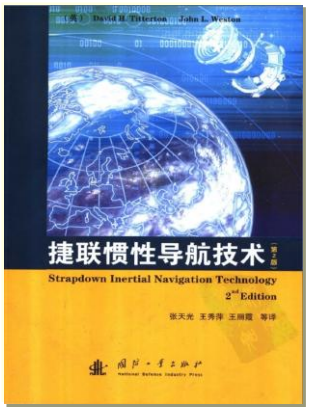


捷联惯导系统计算

参考书-捷联惯性导航技术（第11章）

主要内容

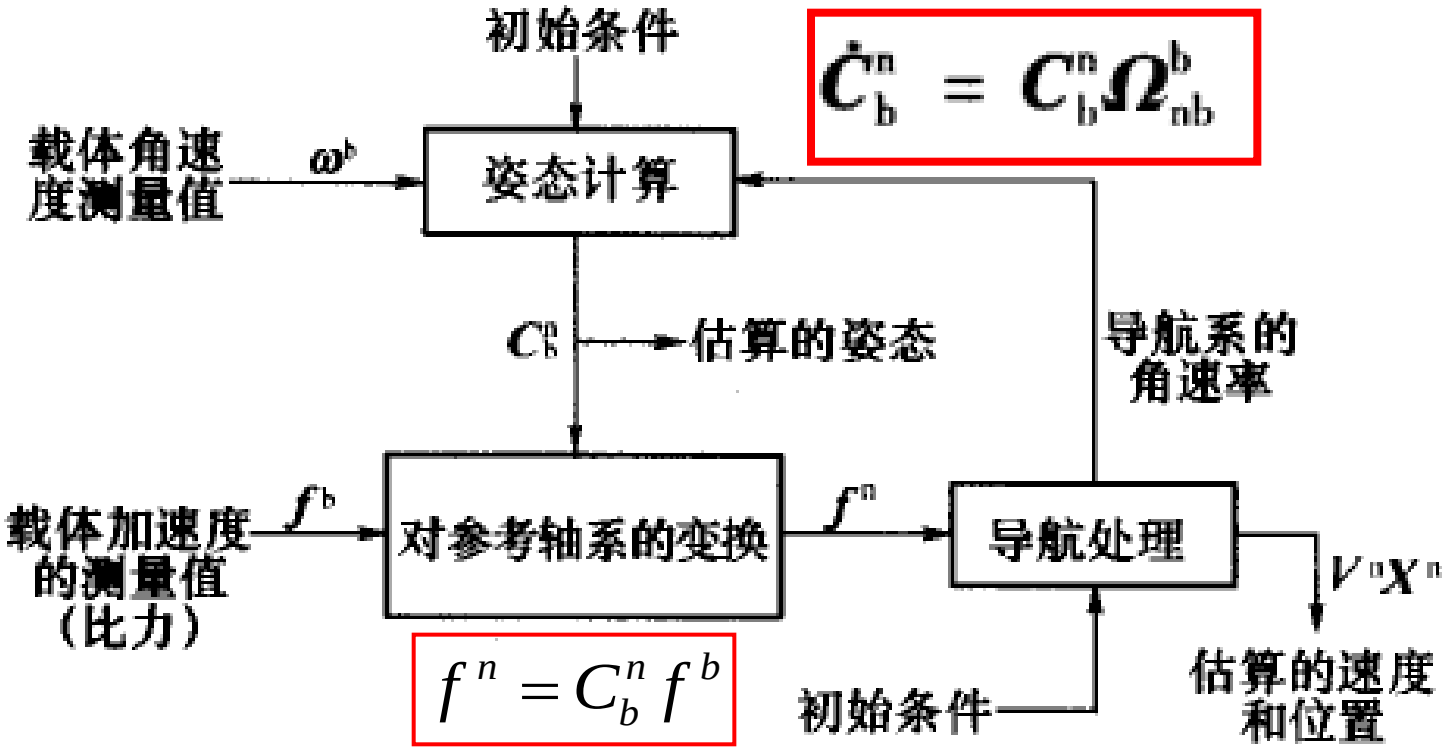
- SINS的计算任务和思想（11.1）
- 姿态计算（方向余弦算法）（11.2）
- 加速度矢量变换算法（比力变换）（11.3）
- 导航算法（速度和位置的计算）（11.4）
- 小结



11.1 概述

- 在捷联惯导系统中，为了从陀螺仪以及加速度计提供的惯性测量信息中提取姿态、速度和位置信息，必须要求解在第 3 章中已经描述的解析方程。本章主要考虑这些方程在计算机上的实时求解。
- 下图中示出了姿态的确定、比力的分解以及导航方程的求解等主要计算任务。

11.1 概述



$$\dot{\boldsymbol{v}}_e^n = \boldsymbol{C}_b^n \boldsymbol{f}^b - [2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n] \times \boldsymbol{v}_e^n + \boldsymbol{g}_l^n$$

捷联惯导系统的计算任务

11.1 概述

$$\dot{\boldsymbol{C}}_b^n = \boldsymbol{C}_b^n \boldsymbol{\Omega}_{nb}^b$$

$$\boldsymbol{\Omega}_{nb}^b = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$$f^n = C_b^n f^b$$

- 姿态矩阵的方向余弦算法（11.2）
- 比力变换（11.3）
- 速度和位置的计算（11.4）

$$\dot{\boldsymbol{v}}_e^n = \boldsymbol{C}_b^n \boldsymbol{f}^b - [2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n] \times \boldsymbol{v}_e^n + \boldsymbol{g}_l^n$$

$$\boldsymbol{v}^n = \int_0^t \boldsymbol{f}^n \mathrm{d}t - \int_0^t [2\boldsymbol{\omega}_{ie} + \boldsymbol{\omega}_{en}] \times \boldsymbol{v}^n \mathrm{d}t + \int_0^t \boldsymbol{g} \mathrm{d}t$$

$$\boldsymbol{x}^n = \int_0^t \boldsymbol{v}^n \mathrm{d}t \quad \text{或} \quad \begin{aligned} \dot{h} &= -v_D \\ \dot{\boldsymbol{C}}_n^e &= \boldsymbol{C}_n^e \boldsymbol{\Omega}_{en}^n \end{aligned} \quad \boldsymbol{\Omega}_{en}^n = [\boldsymbol{\omega}_{en}^n \times], \boldsymbol{\omega}_{en}^n = [\dot{l}\cos L - \dot{L} - \dot{l}\sin L]^{\mathrm{T}}$$

捷联惯导系统的计算任务

11.2 姿态计算

捷联系统中姿态计算是非常关键的。

确定姿态的常规方法是计算运载体坐标系与参考坐标系相关联的方向余弦矩阵，或者使用一种数值积分方法计算等效的四元数。

只要计算频率足够高，那么在理论上讲即使出现高频角运动的情况，足够精确地计算载体姿态也是可能的。然而，实际上这可能给处理器造成极大的负担。

由博茨(Bortz) 提出的另一种方法是把姿态变化表示为一种旋转矢量。如下面所述，姿态计算可方便地分成低速和高速两部分，这里表示为 k 循环和j 循环。低速计算部分用于计算由运载体机动引起的相对低频、大幅值的运动；高速计算部分用于计算运载体高频、小幅值的运动。

$$\dot{\boldsymbol{C}}_{\mathrm{b}}^{\mathrm{n}} = \boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{\Omega}_{\mathrm{nb}}^{\mathrm{b}} = \boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{\Omega}_{\mathrm{ib}}^{\mathrm{b}} - \boldsymbol{\Omega}_{\mathrm{in}}^{\mathrm{n}}\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}$$

$$\boldsymbol{\Omega}_{nb}^{b} = \boldsymbol{\Omega}_{ib}^{b} - \boldsymbol{\Omega}_{in}^{b}$$

$$\boldsymbol{\Omega}_{in}^{b} = \boldsymbol{C}_{n}^{b}\boldsymbol{\Omega}_{in}^{n}\boldsymbol{C}_{b}^{n}$$

$$\boldsymbol{\omega}_{in} = \boldsymbol{\omega}_{ie} + \boldsymbol{\omega}_{en}$$

$$\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{n}} = [\Omega\cos L \quad 0 \quad -\Omega\sin L]^{\mathrm{T}}$$

$$\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}} = [\dot{\ell}\cos L \quad -\dot{L} \quad -\dot{\ell}\sin L]^{\mathrm{T}}$$

11.2.1 方向余弦法

$$\dot{\boldsymbol{C}}_{\mathrm{b}}^{\mathrm{n}} = \boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{\Omega}_{\mathrm{nb}}^{\mathrm{b}} \qquad \dot{\boldsymbol{C}}_{\mathrm{n}}^{\mathrm{e}} = \boldsymbol{C}_{\mathrm{n}}^{\mathrm{e}}\boldsymbol{\Omega}_{\mathrm{en}}^{\mathrm{n}}$$

- 为了更新在第 3 章里作了定义的方向余弦矩阵 C，需要求解下面形式的矩阵微分方程：

$$\dot{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{\Omega}$$

$$\boldsymbol{\Omega}_{\mathrm{nb}}^{\mathrm{b}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$$\boldsymbol{C}_{(t)} = \boldsymbol{C}_{(0)} + \int_0^t \boldsymbol{C}(t)\boldsymbol{\Omega}\,dt$$

$$\boldsymbol{C}_{(t)} = \boldsymbol{C}_{(0)} e^{\int_0^t \boldsymbol{\Omega} dt}$$

$$\boldsymbol{C}_{(t+\Delta t)} = \boldsymbol{C}_{(t)} e^{\int_t^{t+\Delta t} \boldsymbol{\Omega} dt}$$

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k \exp \int_{t_k}^{t_{k+1}} \boldsymbol{\Omega} \mathrm{d}t$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{\Omega} \mathrm{d}t = [\boldsymbol{\sigma} \times]$$

$$\boldsymbol{\sigma} \times = \begin{bmatrix} 0 & -\sigma_z & \sigma_y \\ \sigma_z & 0 & -\sigma_x \\ -\sigma_y & \sigma_x & 0 \end{bmatrix}$$

斜对称阵

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k \exp[\boldsymbol{\sigma} \times] = \boldsymbol{C}_k \boldsymbol{A}_k$$

$$\boldsymbol{\sigma} = \sqrt{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}$$

11.2.1 姿态计算-方向余弦算法

$$\boldsymbol{A}_k = \boldsymbol{I} + [\boldsymbol{\sigma}\times] + \frac{[\boldsymbol{\sigma}\times]^2}{2!} + \frac{[\boldsymbol{\sigma}\times]^3}{3!} + \frac{[\boldsymbol{\sigma}\times]^4}{4!} + \cdots$$

$$[\boldsymbol{\sigma}\times]^2 = \begin{bmatrix} -(\sigma_y^2+\sigma_z^2) & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & -(\sigma_x^2+\sigma_z^2) & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & -(\sigma_x^2+\sigma_y^2) \end{bmatrix}$$

对称阵

$$[\boldsymbol{\sigma}\times]^3 = -(\sigma_x^2+\sigma_y^2+\sigma_z^2)[\boldsymbol{\sigma}\times]$$

$$[\boldsymbol{\sigma}\times]^4 = -(\sigma_x^2+\sigma_y^2+\sigma_z^2)[\boldsymbol{\sigma}\times]^2$$

$$\boldsymbol{A}_k = \boldsymbol{I} + [\boldsymbol{\sigma}\times] + \frac{[\boldsymbol{\sigma}\times]^2}{2!} - \frac{\sigma^2[\boldsymbol{\sigma}\times]^2}{3!} - \frac{\sigma^2[\boldsymbol{\sigma}\times]^2}{4!} + \cdots =$$

$$\boldsymbol{I} + \left[1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} - \cdots\right][\boldsymbol{\sigma}\times] + \left[\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - \cdots\right][\boldsymbol{\sigma}\times]^2$$

$$\boldsymbol{A}_k = \boldsymbol{I} + \frac{\sin\sigma}{\sigma}[\boldsymbol{\sigma}\times] + \frac{(1-\cos\sigma)}{\sigma^2}[\boldsymbol{\sigma}\times]^2$$

11.2.1 姿态计算-方向余弦算法

$$\boldsymbol{A}_k = \boldsymbol{I} + \frac{\sin\sigma}{\sigma}[\boldsymbol{\sigma}\times] + \frac{(1-\cos\sigma)}{\sigma^2}[\boldsymbol{\sigma}\times]^2$$

如果可以完美地实施这个计算方程，它将会生成一个当需要对测量的比力矢量进行转换的时候才需要对它进行计算的正交矩阵。

$$\boldsymbol{A}_k = \boldsymbol{I} + a_1[\boldsymbol{\sigma}\times] + a_2[\boldsymbol{\sigma}\times]^2$$

式中

$$a_1 = 1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} - \cdots \qquad \boldsymbol{a_1} = \frac{\sin\sigma}{\sigma}$$

$$a_2 = \frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - \cdots \qquad \boldsymbol{a_2} = \frac{(1-\cos\sigma)}{\sigma^2}$$

当载体运动被捷联陀螺敏感时，用递归算法可以更新方向余弦短阵。

11.2.1 姿态计算-方向余弦算法

$$\dot{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{\Omega}$$

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k \exp[\boldsymbol{\sigma}\times] = \boldsymbol{C}_k \boldsymbol{A}_k$$

$$\boldsymbol{A}_k = \boldsymbol{I} + a_1[\boldsymbol{\sigma}\times] + a_2[\boldsymbol{\sigma}\times]^2$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{\Omega}\mathrm{d}t = [\boldsymbol{\sigma}\times]$$

$$\boldsymbol{\sigma}\times = \begin{bmatrix} 0 & -\sigma_z & \sigma_y \\ \sigma_z & 0 & -\sigma_x \\ -\sigma_y & \sigma_x & 0 \end{bmatrix}$$

$$[\boldsymbol{\sigma}\times]^2 = \begin{bmatrix} -(\sigma_y^2+\sigma_z^2) & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & -(\sigma_x^2+\sigma_z^2) & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & -(\sigma_x^2+\sigma_y^2) \end{bmatrix}$$

$a_1 = 1, a_2 = 0$ 为一阶算法

$a_1 = 1, a_2 = 0.5$ 为二阶算法

$a_1 = 1 - (\sigma^2/6), a_2 = 0.5$ 为三阶算法

$a_1 = 1 - (\sigma^2/6), a_2 = 0.5 - (\sigma^2/24)$ 为四阶算法

$$a_1 = \frac{\sin\sigma}{\sigma}$$

$$a_2 = \frac{(1-\cos\sigma)}{\sigma^2}$$

11.2.1 姿态计算-方向余弦算法

$$\dot{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{\Omega}$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{\Omega} \mathrm{d}t = [\boldsymbol{\sigma} \times]$$

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_k \exp[\boldsymbol{\sigma} \times] = \boldsymbol{C}_k \boldsymbol{A}_k$$

$$\boldsymbol{\sigma} \times = \begin{bmatrix} 0 & -\sigma_z & \sigma_y \\ \sigma_z & 0 & -\sigma_x \\ -\sigma_y & \sigma_x & 0 \end{bmatrix}$$

$$\boldsymbol{A}_k = \boldsymbol{I} + a_1[\boldsymbol{\sigma} \times] + a_2[\boldsymbol{\sigma} \times]^2$$

$$a_1 = \frac{\sin\sigma}{\sigma} \quad a_2 = \frac{(1-\cos\sigma)}{\sigma^2}$$

$$[\boldsymbol{\sigma} \times]^2 = \begin{bmatrix} -(\sigma_y^2 + \sigma_z^2) & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & -(\sigma_x^2 + \sigma_z^2) & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & -(\sigma_x^2 + \sigma_y^2) \end{bmatrix}$$

一阶算法 $\boldsymbol{C}_{k+1} = \boldsymbol{C}_k(\boldsymbol{I} + [\boldsymbol{\sigma}\times])$

二阶算法 $\boldsymbol{C}_{k+1} = \boldsymbol{C}_k(\boldsymbol{I} + [\boldsymbol{\sigma}\times] + \frac{1}{2}[\boldsymbol{\sigma}\times]^2)$

三阶算法 $\boldsymbol{C}_{k+1} = \boldsymbol{C}_k\{\boldsymbol{I} + (1 - \frac{\sigma^2}{6})[\boldsymbol{\sigma}\times] + \frac{1}{2}[\boldsymbol{\sigma}\times]^2\}$

四阶算法 $\boldsymbol{C}_{k+1} = \boldsymbol{C}_k\{\boldsymbol{I} + (1 - \frac{\sigma^2}{6})[\boldsymbol{\sigma}\times] + (\frac{1}{2} - \frac{\sigma^2}{24})[\boldsymbol{\sigma}\times]^2\}$

11.2.1 姿态计算-方向余弦算法-误差分析

这里计算的姿态矩阵写成 $\hat{A}$ ，它可以用真实姿态矩阵 **A** 和误差矩阵 **E** 来表示:

$$\hat{\boldsymbol{A}} = \boldsymbol{A}[\boldsymbol{I} + \boldsymbol{E}] \qquad \text{整理得} \qquad \boldsymbol{E} = \boldsymbol{A}^{\mathrm{T}}\hat{\boldsymbol{A}} - \boldsymbol{I}$$

代入上节中的**A**和$\hat{A}$

$$\boldsymbol{E} = \left[\boldsymbol{I} - \frac{\sin\sigma}{\sigma}[\boldsymbol{\sigma}\times] + \frac{(1-\cos\sigma)}{\sigma^2}[\boldsymbol{\sigma}\times]^2\right][\boldsymbol{I} + a_1[\boldsymbol{\sigma}\times] + a_2[\boldsymbol{\sigma}\times]^2] - \boldsymbol{I} =$$

$$(\sigma a_1\cos\sigma - \sin\sigma + \sigma^2 a_2\sin\sigma)\frac{[\boldsymbol{\sigma}\times]}{\sigma} +$$

$$(1 - \cos\sigma - \sigma a_1\sin\sigma + \sigma^2 a_2\cos\sigma)\frac{[\boldsymbol{\sigma}\times]^2}{\sigma^2} \qquad (11.1$$

记为 $\boldsymbol{E} = \boldsymbol{U} + \boldsymbol{S}$

$$(\sigma a_1\cos\sigma - \sin\sigma + \sigma^2 a_2\sin\sigma)\frac{[\boldsymbol{\sigma}\times]}{\sigma} \qquad (1 - \cos\sigma - \sigma a_1\sin\sigma + \sigma^2 a_2\cos\sigma)\frac{[\boldsymbol{\sigma}\times]^2}{\sigma^2}$$

11.2.1 姿态计算-方向余弦算法-误差分析

若 $A^{\mathrm{T}}A = I$，那么A是一个正交矩阵。对计算的矩阵，可以写为

$$\hat{A}^{\mathrm{T}}\hat{A} = [I + E]^{\mathrm{T}}[I + E]$$

忽略二阶和更高阶

$$\hat{A}^{\mathrm{T}}\hat{A} = I + E + E^{\mathrm{T}}$$

用它们的对称和斜对称元替代公式中的 E 和 E^{T}，并且注意到 $S^{\mathrm{T}} = S, U^{\mathrm{T}} = -U$，则

$$\hat{A}^{\mathrm{T}}\hat{A} = I - 2S$$

$$A^{\mathrm{T}}\hat{A} = I + S + U$$

S 表示矩阵 A 的正交偏差， S 的对角线元素被称为标度误差，而非对角线项表示扭斜误差。

如果 S 为零，$\hat{A}$ 就变成一个表示坐标系旋转的正交矩阵，它与由 A 定义的旋转不同。U 为这两个旋转之间的差。

11.2.1 姿态计算-方向余弦算法-误差分析

定义参数 **Ddc**为 **U** 的上下非对角线元素和的平方根。它除以计算机更新间隔 **δt** 可以用来度量计算姿态矩阵中的漂移。参数 **Ddc** 可以用来对各种阶次的姿态算法进行精度的评估。

定义： $$U = (\sigma a_1 \cos\sigma - \sin\sigma + \sigma^2 a_2 \sin\sigma)\frac{[\sigma\times]}{\sigma}$$

在单一的x轴旋转情况下 $\sigma = [\sigma \quad o \quad o]^T$

$$D_{dc} = \frac{1}{\delta t}(\sigma a_1 \cos\sigma - \sin\sigma + \sigma^2 a_2 \sin\sigma)$$

$a_1 = 1, a_2 = 0$	为一阶算法
$a_1 = 1, a_2 = 0.5$	为二阶算法
$a_1 = 1 - (\sigma^2/6), a_2 = 0.5$	为三阶算法
$a_1 = 1 - (\sigma^2/6), a_2 = 0.5 - (\sigma^2/24)$	为四阶算法

11.2.1 姿态计算-方向余弦算法-误差分析

例　考虑角增量的最大值(σ_{max})为 0.lrad 和 0.05rad 的情况。如果载体的最大角速率为10rad/s ，这些值就对应于 0.0ls 和 0.005s 的计算机更新间隔。表中列出了上述两种情况下计算姿态的漂移误差，这两种情况用的是不同阶次的算法。

算法阶次	姿态漂移误差（°/h）	
	σ_{max} =0.lrad	σ_{max} =0.05rad
1	6870	1720
2	3430	860
3	7	0.4
4	1.7	0.1

减少更新间隔可以显著改善计算精度。由于这种舍项舍去了漂移误差表达式中的三阶和四阶项，包含三阶项所得到的漂移率大大地降低了。在许多应用中，长时间的高旋转速率通常是不能持续的，所期望的平均速率会大大低于上面的分析中所考虑的10rad/s。在这种情况下，由不精确的姿态计算引起的平均漂移误差比表中列出的数据要小得多。因此在某些应用中采用一阶或二阶更新算法可能是可行的。

11.2.1 姿态计算-四元数算法-误差分析

例　考虑角增量的最大值(σ_{max})为 0.lrad 和 0.05rad 的情况。如果载体的最大角速率为10rad/s ，这些值就对应于 0.0ls 和 0.005s 的计算机更新间隔。表中列出了上述两种情况下计算姿态的漂移误差，这两种情况用的是不同阶次的算法。

表 11.2　用不同阶次的四元数算法计算的姿态漂移误差

算法阶次	姿态漂移误差/((°)/h)	
	σ_{max} =0.1rad	σ_{max} =0.05rad
1	1720	430
2	860	215
3	0.4	0.06
4	0.2	0.06

与 11.2.1 节的表 11.1 列出的结果相比较，可见四元数法的漂移值比用方向余弦法得到的结果要小。这主要是因为四元数方程包含有 $\sin(\sigma/2)$ 和 $\cos(\sigma/2)$ 项的展开式，而方向余弦方程则含有 σ 项的相似项。这也说明当 σ 为 0.1rad 时四元数法的漂移数与当 σ 为 0.05rad 时方向余弦法的漂移是一致的。因此，四元数表达式在单轴旋转及给定的舍位阶数下，得到更精确的姿态解。

姿态参数和姿态表达式计算的选择

四元数法由于给出了正交姿态矩阵而具有某些优点，但两种方法优劣的比较现在还无法给出定论。另外，在先前几节中给出的分析表明，用四元数法计算的姿态精度比用方向余弦表达式所能得到的精度要高。这些因素至少部分地说明为什么四元数法在近年比较流行。

然而，姿态算法的最终选择不大可能唯一依据计算精度来作出。在实际应用中，计算量和对存储器的要求将是确定算法的主要因素。在这种情况下，四元数方法仍是一个选择的趋向，这主要是因为它需要更新的参数较少，只要 4 个四元数而不是 9 个方向余弦。然而，当考虑到整个捷联计算任务（包括测量的比力矢量求解）时，两种方法的优劣就变得不是那么清晰了。

11.2.2 姿态计算-旋转角的计算

旋转角的计算

方向余弦矩阵更新精度的另一个限制因素是可以确定的旋转角 σ 的精度。首先考虑的情况是角速率矢量 ω 的方向在计算更新的整个间隔内在空间保持不变。在这种情况下可相当简单地对 ω 在 k 计算循环内进行积分即可求得 σ:

$$\boldsymbol{\sigma} = \int_{t_k}^{t_{k+1}} \boldsymbol{\omega} \mathrm{d}t$$

即 σ 是由某些陀螺在时间间隔 t_k 到 t_{k+1} 内直接提供的增量测量值之和。在这种情况下，σ 和 ω 之间的关系也可以表达成：

$$\mathrm{d}\boldsymbol{\sigma}/\mathrm{d}t = \boldsymbol{\omega}$$

11.2.2 姿态计算-旋转角的计算

然而，在一般情况下，简单地相加增量角的测量值来精确地确定 σ 是不可能的。如果 ω 矢量的方向不能在空间保持固定不变，而是在旋转(如锥形运动时)，那么根据博茨可以写成:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{\omega} + \dot{\boldsymbol{\varepsilon}}$$

在一般的运动条件下，即不限定于单轴运动，σ 的表达式可以通过对方程 $\boldsymbol{A}_k = \boldsymbol{I} + \frac{\sin\sigma}{\sigma}[\boldsymbol{\sigma}\times] + \frac{(1-\cos\sigma)}{\sigma^2}[\boldsymbol{\sigma}\times]^2$ 求导得到使 $\mathrm{d}\boldsymbol{A}/\mathrm{d}t = \boldsymbol{A}[\boldsymbol{\omega}\times]$ 求得，给出

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\sigma}\times\boldsymbol{\omega} + \frac{1}{\sigma^2}\left[1 - \frac{\sigma\sin\sigma}{2(1-\cos\sigma)}\right]\boldsymbol{\sigma}\times\boldsymbol{\sigma}\times\boldsymbol{\omega}$$

由于不可交换性的影响，在一系列的旋转之后，最后的方向取决于各次的旋转和旋转次序两个方面。上面的方程说明先前的各次旋转(σ)以及现时的角速率(ω) 是如何影响非交换性项(e) 的构成的。

11.2.2 姿态计算-旋转角的计算

在实际计算中，需要把上式的右边表达式作舍项处理。例如正弦和余弦项改写成级数展开式并且略去 σ 的高于主阶的项，则上面的方程可写成:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\sigma} \times \boldsymbol{\omega} + \frac{1}{12}\boldsymbol{\sigma} \times \boldsymbol{\sigma} \times \boldsymbol{\omega}$$

$$\delta\boldsymbol{\alpha}_{k+1} = \int_{t_k}^{t_{k+1}} \boldsymbol{\alpha} \times \boldsymbol{\omega} \mathrm{d}t$$

其中

$$\boldsymbol{\alpha} = \int_{t_k}^{t} \boldsymbol{\omega} \mathrm{d}t$$

$$\boldsymbol{\sigma} = \boldsymbol{\alpha}_{k+1} + \delta\boldsymbol{\alpha}_{k+1}$$

一般情况下，都存在相当程度的角振动，因此需要用比方向余弦矩阵更新速率更高的速率来求解 $\boldsymbol{\alpha}$，即用j 循环更新速率，且要选择一种足够快的j 循环更新速率来保证在最大载体旋转速率和振动出现时由 $\boldsymbol{\alpha}$ 获得的 σ 值能很好地符合方程的真值。

11.2.2 姿态计算-旋转矢量的补偿

旋转矢量的补偿

圆锥运动是指载体的一根轴在空间描绘出一个圆锥或近似一个圆锥时的运动。这种运动是由于系统两个正交轴同时施加角振荡的结果，而这两种角振荡不同相。

这里假定载体围绕x轴和 y 轴以频率f作振荡，其振幅分别为 和吧。另外，假定在两个通道中存在相位差 φ，那么，可以写出:

$$\boldsymbol{\omega} = 2\pi f[\theta_x \cos 2\pi ft \quad \theta_y \cos(2\pi ft + \phi) \quad 0]^{\mathrm{T}}$$

$$\boldsymbol{\alpha} = [\theta_x |\sin 2\pi ft - \sin 2\pi ft_k| \quad \theta_y |\sin(2\pi ft + \phi) - \sin(2\pi ft_k + \phi)| \quad 0]^{\mathrm{T}}$$

在 δα 方程中用 α 和 ω 两式代人，得

$$\delta\boldsymbol{\alpha} = \pi f \int_{t_k}^{t_{k+1}} \begin{bmatrix} \theta_x |\sin 2\pi ft - \sin 2\pi ft_k| \\ \theta_y |\sin(2\pi ft + \phi) - \sin(2\pi ft_k + \phi)| \\ 0 \end{bmatrix} \times \begin{bmatrix} \theta_x \cos 2\pi ft \\ \theta_y \cos(2\pi ft + \phi) \\ 0 \end{bmatrix} \mathrm{d}t$$

11.2.3 姿态计算-旋转矢量的补偿

由它生成一个 z 轴分量

$$\delta\alpha_z = \pi f\theta_x\theta_y\sin\phi\int_{t_k}^{t_{k+1}}\left|1-\cos 2\pi f(t-t_k)\right|\mathrm{d}t$$

在适当的范围内积分得

$$\delta\alpha_z = \pi f\theta_x\theta_y\sin\phi\left[t_{k+1}-t_k-\frac{\sin 2\pi f(t_{k+1}-t_k)}{2\pi f}\right]$$

设 $t_{k+1}-t_k=\delta t$

$$\delta\alpha_z = \pi f\theta_x\theta_y\delta t\sin\phi\left[1-\frac{\sin 2\pi f\delta t}{2\pi f\delta t}\right]$$

虽然 ω 是绕 z 轴和 y轴循环的，但 δα出现了 z 轴分量。它是一个常值且正比于相位角的正弦和运动的幅值。从上面的方程可以看出，当 φ=τ/2 时， δα最大。在这个条件下，由于 z 轴在空间的运动描绘出了一个圆锥形，因此载体的这种运动称为圆锥运动。

11.2.3 姿态计算-旋转矢量的补偿

在时间间隔 kδt内，如果没有应用上面的修正项，那么计算姿态中的漂移误差可以表示成:

$$\delta\dot{\alpha}_z = \pi f\theta_x\theta_y\sin\phi\left[1 - \frac{\sin 2\pi f\delta t}{2\pi f\delta t}\right]$$

如果 上式与总的系统性能要求相比是一个小量，那么就不需要实施上述修正项。

例

考虑载体以 50Hz 作圆锥运动的情况。绕 x和y轴运动的角幅度取0.1°，如果姿态的更新频率为 100Hz，即δt =0. 01s，那么引起的漂移为 100°/h 。如果把计算频率提高到5 00Hz ，则漂移量就下降到6°/h。针对上述方程更一般的情况，在一给定振动频谱下，可计算出 $\delta\alpha_z$ 的均方根值

11.2.4 姿态计算-载体及导航系的旋转

载体及导航参考系的旋转

矢量 ω 表示载体相对于导航参考系的转动速率，当导航相对于当地地理坐标系时，该方程可用下面的形式(如第 3 章所述)

$$\dot{\boldsymbol{C}}_{\mathrm{b}}^{\mathrm{n}} = \boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{\Omega}_{\mathrm{ib}}^{\mathrm{b}} - \boldsymbol{\Omega}_{\mathrm{in}}^{\mathrm{n}}\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}$$

$$\dot{\boldsymbol{C}}_{b}^{n} = \boldsymbol{C}_{b}^{n}\boldsymbol{\Omega}_{nb}^{b}$$

$$\boldsymbol{\Omega}_{nb}^{b} = \boldsymbol{\Omega}_{ib}^{b} - \boldsymbol{\Omega}_{in}^{b}$$

$$\boldsymbol{\Omega}_{in}^{b} = \boldsymbol{C}_{n}^{b}\boldsymbol{\Omega}_{in}^{n}\boldsymbol{C}_{b}^{n}$$

上式右端第一项是载体角速率的函数，由捷联陀螺测得，而第二项 是较低的导航参考系角速率的函数。

如上所述，当考虑到载体运动时，方向余弦矩阵的更新，即方程$d\boldsymbol{C}_{b}^{n}/dt = \boldsymbol{C}_{b}^{n}\boldsymbol{\Omega}_{ib}^{b}$ 的求解可以用如下两个方程完成。

$$\boldsymbol{C}_{k+1} = \boldsymbol{C}_{k}\exp[\boldsymbol{\sigma}\times] = \boldsymbol{C}_{k}\boldsymbol{A}_{k}$$

$$\boldsymbol{A}_{k} = \boldsymbol{I} + a_1[\boldsymbol{\sigma}\times] + a_2[\boldsymbol{\sigma}\times]^2$$

11.2.4 姿态计算-载体及导航系的旋转

考虑导航系的旋转可以用一种类似的算法。采用如下方程对导航系旋转的方向余弦短阵进行更新

$$\boldsymbol{C}_{l+1} = \boldsymbol{B}_l \boldsymbol{C}_l$$

式中，$\boldsymbol{B}_l$表示把t_{l+1}时刻与t_l时刻的导航系相关联的方向余弦矩阵。$\boldsymbol{B}_l$可以用旋转矢量$\boldsymbol{\theta}$来表示：

$$\boldsymbol{B}_l = \boldsymbol{I} + \frac{\sin\theta}{\theta}[\boldsymbol{\theta}\times] + \frac{(1-\cos\theta)}{\theta^2}[\boldsymbol{\theta}\times]^2$$

这里$\boldsymbol{\theta}$是具有大小和方向的旋转矢量，表示导航系旋转了角度$\boldsymbol{\theta}$。而它的方向从t_l时刻的方向转到了在t_{l+1}时刻的位置。这个角$\boldsymbol{\theta}$可写成：

$$\boldsymbol{\theta} = \int_{t_l}^{t_{l+1}} \boldsymbol{\omega}_{in}^{n} \mathrm{d}t$$

11.2.4 姿态计算-载体及导航系的旋转

实际上，导航系转速通常比载体的转速要低很多，因此导航系旋转的方向余弦更新可以在低得多的速率上执行，即采用 l循环更新速率。另外，它的算法所基于的数学函数可以在较低阶次上舍项。

举例:

若计算 $\boldsymbol{B}_l$ 只考虑一阶项，在考虑地球转动的情况时导航系更新的时间间隔为1s，而漂移误差的净值能够保持在一个可以忽略的低水平。在某些近程导弹的应用中，角速率每小时几百度的测量精度是可以接受的。此时再考虑导航系的旋转(如地球15°/h的转动)就成为多余了。

11.2.7 姿态参数和姿态表达式计算的选择

11.2.7 姿态表达式的选择

采用方向余弦或者四元数参数来表达姿态的相关好处在出版的文献[3]~[5]中已经作了阐明。虽然四元数法由于给出了正交姿态矩阵而具有某些优点,但两种方法优劣的比较现在还无法给出定论。另外,在先前几节中给出的分析表明,用四元数法计算的姿态精度比用方向余弦表达式所能得到的精度要高。这些因素至少部分地说明为什么四元数法在近年比较流行。

然而,姿态算法的最终选择不大可能唯一依据计算精度来作出。在实际应用中,计算量和对存储器的要求将是确定算法的主要因素。在这种情况下,四元数方法仍是一个选择的趋向,这主要是因为它需要更新的参数较少,只要4个四元数而不是9个方向余弦。然而,当考虑到整个捷联计算任务(包括测量的比力矢量求解)时,两种方法的优劣就变得不是那么清晰了。

11.3 加速度计矢量变换算法

用方向余弦进行加速度矢量的变换

测量的比力矢量$\mathbf{f^b}$在导航坐标系中表示如下：

$$f^n = C_b^n f^b \qquad (1)$$

式中：C_b^n是载体系到参考轴系的方向余弦阵。

对上式进行一次积分，可得

$$u^n = \int_{t_k}^{t_{k+1}} f^n dt \qquad (2)$$

式中：$\mathbf{u^n}$ 表示在导航系中速度经历了计算机周期$\mathbf{t_k}$到$\mathbf{t_{k+1}}$后的变化量。

矩阵 C_b^n 随更新间隔时间连续变化，并且可以用 C_b^n 在时刻 $\mathbf{t_k}$的值矩阵 $\mathbf{C_k}$ 和代表载体轴从t时刻变换到更新间隔的起始时刻 $\mathbf{t_k}$的矩阵 **A** 来描述，即

$$C_b^n = C_k A \qquad (3)$$

11.3 加速度计矢量变换算法

用方向余弦进行加速度矢量的变换

将式(3)代式(1)中得

$$u^n = C_k \int_{t_k}^{t_{k+1}} A f^b d_t \qquad (4)$$

A 可以表达成如下形式:

$$A = I + [\alpha\times] + 0.5[\alpha\times]^2 - \cdots \qquad (5)$$

式中

$$\alpha = \int_{t_k}^{t} \omega^b dt$$

综合式(4)和式(5)可得

$$u^n = C_k \int_{t_k}^{t_{k+1}} \left[f^b + \alpha \times f^b + 0.5[\alpha\times]^2 f^b - \cdots \right] dt \qquad (6)$$

忽略2阶小量，可以写成

$$u^n = C_k \left[\int_{t_k}^{t_{k+1}} f^b dt + \int_{t_k}^{t_{k+1}} \alpha \times f^b dt \right] \qquad (7)$$

11.3 加速度计矢量变换算法

用方向余弦进行加速度矢量的变换

如果现在写成: $v = \int_{t_k}^{t} f^b dt$

则可以得到: $\int_{t_k}^{t_{k+1}} f^b dt = v_{k+1}$ $\dot{v} = f^b$

其中，v_{k+1}表示加速度计在$[t_k, t_{k+1}]$时间段内输出的速度增量。

根据 $\alpha = \int_{t_k}^{t} \omega^b dt$ 可得 $\dot{\alpha} = \omega^b$

由于 $\frac{d}{dt}[\alpha \times v] = \alpha \times f^b - v \times \omega^b$

所以 $\alpha \times f^b = \frac{d}{dt}[\alpha \times v] + v \times \omega^b$

$$= \frac{1}{2}\frac{d}{dt}[\alpha \times v] + \frac{1}{2}[\alpha \times f^b + v \times \omega^b]$$

11.3 加速度计矢量变换算法

用方向余弦进行加速度矢量的变换

对上式两边积分得

$$\int_{t_k}^{t_{k+1}} \alpha \times f^b dt = \frac{1}{2}\alpha_{k+1} \times v_{k+1} + \frac{1}{2}\int_{t_k}^{t_{k+1}} \left[\alpha \times f^b + v \times \omega^b\right] dt$$

将上式代入式(7)可得

$$u^n = C_k\left(v_{k+1} + \frac{1}{2}\alpha_{k+1} \times v_{k+1} + \frac{1}{2}\int_{t_k}^{t_{k+1}} (\alpha \times f^b - \omega^b \times v)\,dt\right)$$

其中，α_{k+1}表示陀螺仪在$[t_k,t_{k+1}]$时间段内输出的角度增量。

11.3 加速度计矢量变换算法

$$\boldsymbol{u}^{\mathrm{n}} = \boldsymbol{C}_k\left(\boldsymbol{v}_{k+1} + \frac{1}{2}\boldsymbol{\alpha}_{k+1} \times \boldsymbol{v}_{k+1} + \frac{1}{2}\int_{t_k}^{t_{k+1}} (\boldsymbol{\alpha} \times \boldsymbol{f}^{\mathrm{b}} - \boldsymbol{\omega}^{\mathrm{b}} \times \boldsymbol{v})\,\mathrm{d}t\right)$$

方程(11.64)中 $\boldsymbol{u}^{\mathrm{n}}$ 包含 3 项：

(1) 由惯性测量单元产生的增量速度测量值之和；

(2) 从 t_k 到 t_{k+1} 时间间隔内累加的增量角和在同样时间间隔内速度增量变化的叉积(在文献中这项称为旋转修正)；

(3) 动态积分项。

如果 $\boldsymbol{f}^{\mathrm{b}}$ 和 $\boldsymbol{\omega}^{\mathrm{b}}$ 在更新间隔内保持常值，则 $\boldsymbol{\alpha} = \boldsymbol{\omega}^{\mathrm{b}}t$，$\boldsymbol{V} = \boldsymbol{f}^{\mathrm{b}}t$，代入方程(11.48)，则很容易看出，在这种条件下，积分项都为零。然而，如果 $\boldsymbol{f}^{\mathrm{b}}$ 和 $\boldsymbol{\omega}^{\mathrm{b}}$ 在更新间隔有很大的变化，为了提供动态运动的补偿，计算这个积分也许是必要的。在这种情况下，计算这个积分的速率需要大大超过动态运动的频率，即上节里所提到的 j 循环更新速率。表示这种效果

11.3 导航算法

对导航计算的不同部分用不同计算速率的优越性。例如，含有地球转速的项，就不需要像含有载体速率项那样频繁估算。

速度方程 $$\boldsymbol{v}^{n} = \int_0^t \boldsymbol{f}^{n}\mathrm{d}t - \int_0^t [2\boldsymbol{\omega}_{ie} + \boldsymbol{\omega}_{en}] \times \boldsymbol{v}^{n}\mathrm{d}t + \int_0^t \boldsymbol{g}\mathrm{d}t$$

位置方程 $$\boldsymbol{x}^{n} = \int_0^t \boldsymbol{v}^{n}\mathrm{d}t$$

为了确定运载体在当地地理坐标系中的速度和位置，在导航处理器中需要估算积分项。在上面方程中的矢量项可以用分量形式表示为：

$$\boldsymbol{v}^{n} = [v_N \quad v_E \quad v_D]^T$$

$$\boldsymbol{x}^{n} = [x_N \quad x_E \quad -h]^T$$

$$\boldsymbol{\omega}_{ie} = [\Omega\cos L \quad 0 \quad -\Omega\sin L]^T$$

$$\boldsymbol{\omega}_{en} = \left[\frac{v_E}{R_0 + h} \quad \frac{-v_N}{R_0 + h} \quad \frac{-v_E\tan L}{R_0 + h}\right]^T$$

$$\boldsymbol{g} = [0 \quad 0 \quad g]^T$$

速度方程中的第一个积分项表示经过每一个更新周期后速度变化的和 为：

$$\boldsymbol{u}^{n} = \int_{t_k}^{t_{k+1}} \boldsymbol{f}^{n}\mathrm{d}t$$

11.3 导航算法

因为这一项是载体姿态的函数，$f^n = C_b^n f^b$，因此它必须考虑运载体动态运动的情况，需用足够高的速率进行计算。包括重力影响的速度矢量可以用下式在时间间隔 t_k 到 t_{k+1} 内进行计算:

$$v_{k+1}^n = v_k^n + u^n + g\delta t$$

速度方程中的第二个积分项含有哥氏修正项。在一般情况下，它对旷的贡献与方程的其他项相比比较小。因为哥氏项数值变化的速率相对较低，所以用 相对较低的 l 更新速率对它进行修正就可以了，即

$$v_{l+1}^n = [I - 2\Omega_{ie}\delta t_l - \Omega_{en}\delta t_l]v_l^n$$

$$\Omega_{ie} = [\omega_{ie} \times]$$

$$\Omega_{en} = [\omega_{en} \times]$$

$$v_l^n = \text{在 } t_l \text{ 时刻的速度矢量}$$

$$\delta t_l = t_{l+1} - t_l$$

最后，利用位置方程积分速度矢量得到位置。

11.3 导航算法

积分形式的选择取决于应用情况。对近程、低精度的应用，低阶方案(如矩形或梯形积分)很可能就可以了。在时间间隔 t_k 到 t_{k+1} 内计算位置更新的方程如下:

矩形积分:
$$\boldsymbol{x}_{k+1}^{n} = \boldsymbol{x}_{k}^{n} + \boldsymbol{v}_{k}^{n}\delta t$$

梯形积分:
$$\boldsymbol{x}_{k+1}^{n} = \boldsymbol{x}_{k}^{n} + \left(\frac{\boldsymbol{v}_{k}^{n} + \boldsymbol{v}_{k+1}^{n}}{2}\right)\delta t$$

对机载和船载惯导系统，性能要求更高，可能需要较高阶次的权分方案，如辛普森法则或四阶龙格一库塔积分。

辛普森法则:
$$\boldsymbol{x}_{k+1}^{n} = \boldsymbol{x}_{k}^{n} + \left(\frac{\boldsymbol{v}_{k-1}^{n} + 4\boldsymbol{v}_{k}^{n} + \boldsymbol{v}_{k+1}^{n}}{3}\right)\delta t$$

11.3 导航算法

常常需要确定相对于地球的海拔位置和现时当地垂直导航参考坐标系相对地球坐标系的角度方向，这通常用经纬度表示。为避免数学上的奇异性，角位置参数可以用导航系(n) 相对地球系 (e) 的方向余弦矩阵来表达，位置方向余弦矩阵及高度的变化由下列微分方程表示:

$$\dot{h} = -v_D$$

$$\dot{C}_n^e = C_n^e \Omega_{en}^n$$

$$\Omega_{en}^n = [\omega_{en}^n x], \omega_{en}^n = [\dot{l}\cos L \quad -\dot{L} \quad -\dot{l}\sin L]^T$$

导航软件执行的算法可以依据上面方程的积分来建立:

$$h_l = h_{l-1} + \Delta h_l$$

$$C_{nl+1}^e = C_{nl}^e \Delta C_{l+1}^l$$

11.3 导航算法

$\Delta \boldsymbol{C}_{l+1}^{l}$ 表示导航轴系在 t_{l+1} 时刻与在 t_l 时刻相关的方向余弦矩阵。它可以用旋转矢量。表示如下：

$$\Delta \boldsymbol{C}_{l+1}^{l} = \boldsymbol{I} + \frac{\sin\zeta}{\zeta}[\boldsymbol{\zeta}\times] + \frac{(1-\cos\zeta)}{\zeta^2}[\boldsymbol{\zeta}\times]^2$$

是具有幅值和方向的旋转矢量，即导航系绕 $\boldsymbol{\zeta}$ 旋转，其转角等于 $\boldsymbol{\zeta}$ 的幅值，此时导航系将从 t_l 时刻的方向转到 t_{l+1} 时刻的位置。

$$\boldsymbol{\zeta} \approx \int_{t_l}^{t_{l+1}} \boldsymbol{\omega}_{en}^{n} \mathrm{d}t$$

在位置矢量从计算机循环 l 到 $l+1$ 发生旋转的情况下，为了获得旋转角矢量 $\boldsymbol{\xi}$ 的精确更新，可采用一种用于速度更新计算形式的算法。导出的算法包括了位置矢量旋转和角速率与比力组合动态影响的补偿项。这里的动态效应会导致校正误差。

11.4 小结

捷联导航计算涉及由紧密安装在运载体上的惯性仪表测得的角速率和比力来确定运载体的姿态、速度和位置。把惯性敏感器的测量值用于各种方程中求得所要求的导航信息。

这一章中，描述了一些在惯性导航系统处理器内的计算方程。基于给定的方程可以开发各种算法，算法将直接接受和处理增量形式的惯性测量值。为了获得各种算法的实时解，建议捷联计算分成如下的低、中和高频部分。

11.4 小结

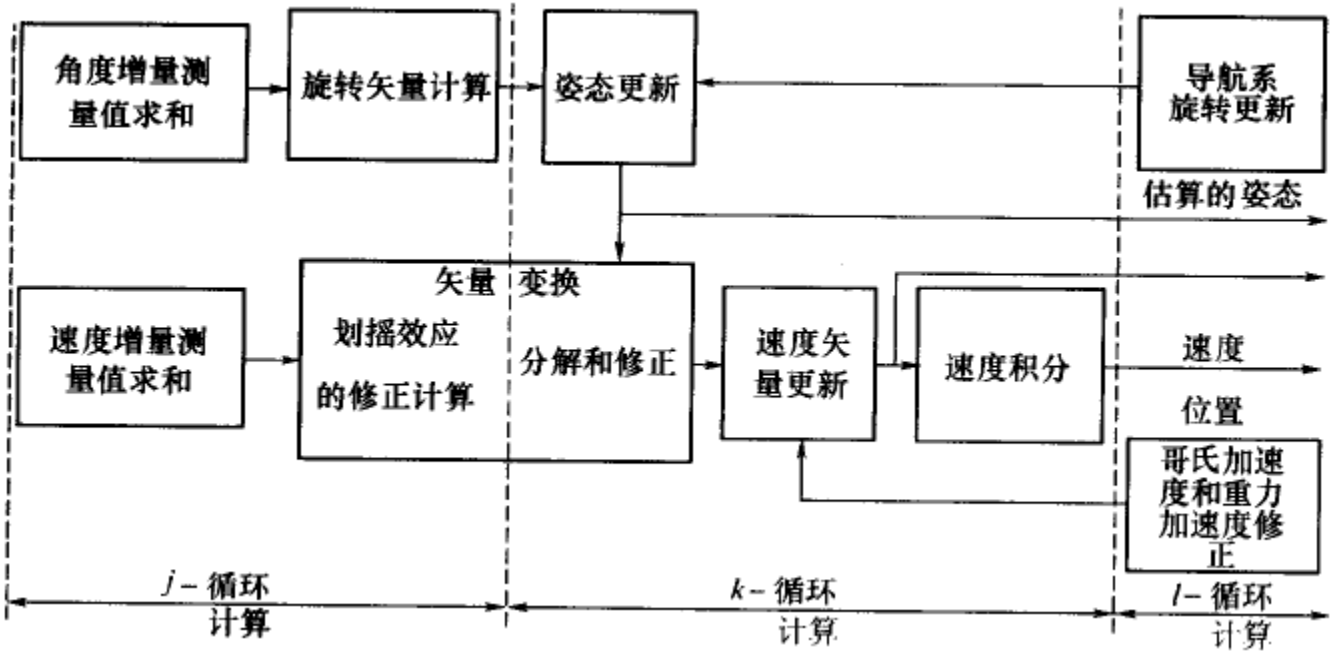
(1)低频计算 (l- 循环)。捷联导航方程的某些部分涉及一些随时间变化非常慢速的项，如“地球转速”。因此，用相对较低的速率来执行算法的这些部分是足够的。特别对导航坐标系旋转情况下的计算姿态更新及导航计算中哥氏修正的应用都可以在低速率上进行。另外，如果需要姿态的正交化和归一化的算法，也可以在低速率上执行。

(2) 中频计算（k-循环)。大量的计算应该是在中速率上执行。为了处理由于运载体机动而产生的大幅度动态运动的计算，需要选用中速率。这包括四元数或方向余弦的计算、加速度矢量的变换以及导航方程主要部分的求解。

(3) 高频计算 (j -循环)。为了处理周期运动(例如圆锥运动及划摇运动)，需要在高速率上进行某些相对简单的计算。这可以对在较低频率更新周期内发生的角速率和线加速度的变化进行补偿。

11.4 小结

假定惯性测量装置具有与所要求的最高计算频率相一致的输出能力。需要执行的各种计算功能简要地列于下图。



捷联计算任务总结

11.4 小结

在现代的捷联系统中，研究重点集中在为求解本章所描述的姿态和导航微分方程更确切的积分解析解而使用的更精确的算法上。

计算精度的改善使捷联算法更容易验证，因为在给定的运动条件下，试验结果能够很好地符合解析解。通过现代计算机技术的应用，更精确的算法会得到更广泛应用；计算机的快速处理速度加上长的浮点字长使得这样的算法能够执行。

思考题

- ✓ 1、画图并说明SINS导航计算的主要任务。
- ✓ 2、推导方向余弦矩阵微分方程的递推计算表达式，并给出二阶算法和四阶算法的表达式。
- ✓ 3、推导并给出用方向余弦进行加速度变换的表达式。
- ✓ 4、推导并给出速度计算和位置计算的递推表达式。