**高精度捷联惯导算法PSINS C++核心程序**

**使用手册**

西北工业大学自动化学院 严恭敏

2021-04-24

目录

[1概述 3](#_Toc29920)

[2编译控制 3](#_Toc10338)

[3常用变量和符号的约定 4](#_Toc7583)

[4宏定义与全局变量 4](#_Toc17867)

[4.1宏定义与全局变量 4](#_Toc20493)

[4.2全局变量类 5](#_Toc6568)

[5基础数据类 6](#_Toc29089)

[5.1三维向量 6](#_Toc16781)

[5.2四元数 7](#_Toc10159)

[5.3三维矩阵 8](#_Toc6983)

[5.4多维向量 9](#_Toc6008)

[5.5多维矩阵 9](#_Toc813)

[6一些辅助数据处理类 11](#_Toc12194)

[6.1 Allan方差法统计序列的方差 11](#_Toc27192)

[6.2直接统计序列的均值和方差 11](#_Toc3720)

[6.3基于移入移出法的序列统计 12](#_Toc10638)

[6.4最大最小值统计 13](#_Toc29705)

[6.5 IIR数据滤波器 13](#_Toc26528)

[6.6环型数组存储类 14](#_Toc25271)

[7捷联惯导算法类 14](#_Toc17169)

[7.1地球参数计算 14](#_Toc29589)

[7.2惯性测量单元数据处理 15](#_Toc23040)

[7.3捷联惯导算法 15](#_Toc14542)

[7.4 AVP插值 16](#_Toc14438)

[8卡尔曼滤波类 16](#_Toc12036)

[9组合导航类 17](#_Toc20139)

[10初始对准算法类 18](#_Toc31266)

[10.1惯性系初始对准算法 18](#_Toc22946)

[10.2卡尔曼滤波初始对准算法 19](#_Toc20967)

[10.3基于航迹角的动态粗对准算法 19](#_Toc12158)

[10.4基于数据存储技术的初始对准算法 20](#_Toc25461)

[10.5“速度+姿态”匹配传递对准算法 21](#_Toc16785)

[11低成本MEMS航姿类 22](#_Toc22697)

[12文件读写 23](#_Toc24779)

[12.1通用文件读写 23](#_Toc31511)

[12.2用户定制文件读取 24](#_Toc845)

[13演示举例 24](#_Toc22477)

# 1概述

本PSINS C++核心代码在Windows\VC++6.0环境下编写且使用正常，在其它编译器下使用可能会遇到一些小的编译问题，但应该都不难修改，用户可自行处理。该 C++核心代码已具有成功移植应用于DSP\[TMS320C6713（CCS4.0编译器）和ARM\STM32F405（Keil uVision4编译器）的先例。](http://www.baidu.com/link?url=B303KN8MVjjRLQTRepHOfE4CQuxQJhnMxhKFKhfxcuq91AZZZb79p37hgVBOaveBO54KfyfmMVTZoOuH6oA_9gJ-JuBIL7jgCC4uYT5hmye) 作者将尽最大努力以完善代码的正确性和可靠性，但不承诺用户的实际应用总是有效的。

PSINS C++核心代码具有以下显著特点：

* 代码全部开源；
* 不依赖于其它外部的数学运算库函数，仅用标准C库即可；
* 编写了大量的四元数、三维向量、三维矩阵、多维向量、多维矩阵运算符重载函数库；
* 捷联惯导更新算法考虑了全面误差补偿细节，精度高，满足所有的实际应用场合；
* 具备很好的SINS/GNSS组合导航类设计，在解决该类型组合导航问题中扩展应用十分方便；
* 采用了时间分片Kalman滤波技术，能满足普通嵌入式单片机的实时应用；
* 除初始化外没有任何动态内存分配操作，实时运算时间可控。

PSINS C++核心代码可免费下载，供学习和研究使用，但不得应用于商业目的。

下载网址：[www.psins.org.cn](http://www.psins.org.cn) ，作者将不定期更新。

# 2编译控制

除了根据不同的编译器和实际需要对PSINS.h文件中的以下宏定义进行“//注释”外，用户不应该修改PSINS.h和PSINS.cpp中的任何代码，否则存在会被将来作者新发布版本覆盖的风险。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* compiling control !!! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define PSINS\_MATRIX\_MAX\_DIM 20 最大矩阵维数定义

//#define PSINS\_IO\_FILE 文件读写操作，在嵌入式中不用

//#define PSINS\_RMEMORY 自定义内存操作，可用于逆向导航

//#define PSINS\_AHRS\_MEMS 启用航姿仪算法

//#define PSINS\_psinsassert 是否启用断言函数，仅用于调试

//#define MAT\_COUNT\_STATISTIC 最大矩阵堆栈个数，仅用于调试

//#define PSINS\_STACK 堆栈大小统计，仅用于调试

//#define PSINS\_UART\_PUSH\_POP 是否启用UART收数操作

//#define PSINS\_VC\_AFX\_HEADER 是否采用VC60文件遍历操作，嵌入式中不用

# 3常用变量和符号的约定

* 惯性坐标系(i)、地球坐标系(e，即地心地固坐标系)。
* 导航坐标系(n)：东E-北N-天U。
* 载体坐标系(b)：右R-前F-上U。
* 角增量wm、速度增量vm：PSINS惯导算法里使用的陀螺和加速度计输入都统一是增量信息（对应单位rad或m/s），如果用户数据中是角速度/比力信息，则简单地乘以采样间隔ts处理即可。
* 姿态/欧拉角向量：att=[俯仰pitch; 横滚roll; 方位yaw]，俯仰角范围-pi/2~pi/2，机头朝上为正；横滚角范围-pi~pi，右倾为正；方位角范围-pi~pi，北偏西为正。
* 速度向量：vn=[vE; vN; vU]。
* 位置向量：pos=[纬度lat; 经度lon; 高度hgt]。
* 姿态速度位置导航参数：avp=[att; vn; pos; t]。
* 姿态阵：Cnb，一般遵从规律是从左到右从上到下书写，即表示C^n\_b，它表示从b系到n系的坐标变换矩阵。对应姿态四元数写为qnb。
* 角速度wnie：表示w^n\_{ie}即e系相对于i系的角速度在n系的投影；wnin和wnen等类似。
* 失准角误差phi=[phiE;phiN;phiU]；速度误差dvn；位置误差dpos=[dlat;dlon;dhgt]；陀螺漂移eb=[ebx;eby;ebz]；加表零偏db=[dbx;dby;dbz]；web陀螺角度随机游走/角速率白噪声；wdb加计速度随机游走/比力白噪声。
* 指北方位捷联导航解算类的实例sins；导航导航计算类实例eth；Kalman滤波及其派生类实例kf。

# 4宏定义与全局变量

## 4.1宏定义与全局变量

|  |  |
| --- | --- |
| **全局变量/函数与宏定义** |  |
| #define PI 3.14159265358979  #define PI\_2 (PI/2.0)  #define PI\_4 (PI/4.0)  #define \_2PI (2.0\*PI)  #define EPS (2.22044604925031e-16)  #define INF (3.402823466e+30)  #define INFp5 (INF\*0.5)  #define fEND (10.0\*INF)  #define fXYZU(X,Y,Z,U) 1.0\*(X)\*(U),1.0\*(Y)\*(U),1.0\*(Z)\*(U)  #define fXYZ(X,Y,Z) fXYZU(X,Y,Z,1.0) | PI  PI/2  PI/4  2\*PI  零阈值  无穷大  一半无穷大  浮点参数结束标志  三个一组浮点参数定义  等等，便于滤波器参数设置 |
| #define FRQxx xx  #define TSxx (1.0/FRQxx) | 常用采样频率定义  采样间隔（频率倒数） |
| #define MMD PSINS\_MATRIX\_MAX\_DIM  #define MMD2 (MMD\*MMD) | 向量/矩阵的最大维数定义（需根据滤波维数修改）  方阵的元素个数 |
| #define asinEx(x) asin(range(x, -1.0, 1.0))  #define acosEx(x) acos(range(x, -1.0, 1.0))  #define max(x,y) { (x)>=(y)?(x):(y); }  #define min(x,y) { (x)<=(y)?(x):(y); }  #define pow2(x)  #define CC180toC360(yaw)  #define C360toCC180(yaw)  double diffYaw(double yaw, double yaw0) | 扩展反正弦，防止输入绝对值大于1  扩展反余弦  求最大值  求最小值  平方  北偏西方位角(-PI~PI)转为北偏东方位角(0~2PI)  前一定义的逆  两方位角之间差值（限制在+-PI内） |
| double r2dm(double r)  double dm2r(double dm) | 弧度转换为度分（注：1234.56度分表示12度+34.56分）  度分转换为弧度 |
| BOOL assert(BOOL b);  int sign(double val, double eps=EPS);  double range(double val, double minVal, double maxVal);  double atan2Ex(double y, double x); | 断言验证  判断正负符号，以eps为阈值  限制变量范围 minVal<=val<=maxVal  与atan2相比，防止y==x==0 |
| #define LLH(latitude,longitude,height)  #define PRY(pitch,roll,yaw) | 纬经度以“度”表示的位置向量  以“度”表示的姿态向量 |
| double MKQt(double sR, double tau) | 由马尔可夫过程均方差和相关时间求激励噪声方差 |
| double MagYaw(const CVect3 &mag, const CVect3 &att,  double declination=0.0) | 由三轴磁传感器测量与惯导姿态角计算磁方位角 |
| const CVect3 I31, O31, Ipos;  const CQuat qI;  const CMat3 I33, O33;  const CVect On1, O1n;  const CGLV glv; | 三维元素均为1的向量，三维零向量，单位位置向量（纬经高方向均变化1m）  单位四元数  三维单位矩阵，三维零矩阵  nX1和1Xn纬零向量  全局变量结构体（Global Variable）  特别注意：除此处外，PSINS用户不宜/需小心定义更多其它类的全局变量（由于C++全局类变量初始化顺序不可控）！ |
| int psinslasterror | last-error |
| psinsassert | psins断言函数 |
| disp(i, FRQ, n) | 每个一定频率，控制台输出显示 |

## 4.2全局变量类

|  |  |
| --- | --- |
| **class CGLV** | **全局变量类** |
| *double Re, f, g0, wie* | 地球长半轴、扁率、重力、自转角速率 |
| *double e, e2* | 偏心率、偏心率平方 |
| *double mg, ug, deg, min, sec, hur, ppm, ppmpsh* | 毫g、微g、分钟、秒、小时、ppm、ppm每根号小时 |
| *double dph, dpsh, dphpsh, ugpsh, ugpsHz, mpsh, mpspsh, secpsh* | °/h、°/sqrt(h)、°/h/sqrt(h)、ug/ sqrt(h)、ug/ sqrt(Hz)  m/sqrt(h)、m/s/sqrt(h)、角秒/sqrt(h) |
| CGLV(double Re=6378137.0, double f=(1.0/298.257),  double wie0=7.2921151467e-5, double g0=9.7803267714) | 对地球模型参数初始化，默认为WGS-84模型 |

# 5基础数据类

## 5.1三维向量

|  |  |
| --- | --- |
| **class CVect3** | **三维向量类** |
| *double i, j, k* | 三维向量的三个分量 |
| CVect3(void) | 默认构造函数，不进行任何操作 |
| CVect3(double xyz) | 对三个分量初始化为相同值 |
| CVect3(double xx, double yy, double zz) | 对三个分量初始化 |
| CVect3(const double \*pdata) | 通过数组指针对三个分量初始化 |
| CVect3& operator=(double f) | 三个分量同于相同数值 |
| friend BOOL IsZero(const CVect3 &v, double eps=EPS) | 判断是否为零向量（三个元素均为零） |
| friend BOOL IsZeroXY(const CVect3 &v, double eps=EPS) | 判断X、Y分量是否均为零 |
| CVect3 operator+(const CVect3 &v) const | 加运算 |
| CVect3 operator-(const CVect3 &v) const | 减运算 |
| CVect3 operator\*(const CVect3 &v) const | 叉乘运算 |
| CVect3 operator\*(double f) const | 乘标量 |
| CVect3 operator\*(const CMat3 &m) const | 行向量乘矩阵 |
| CVect3 operator/(double f) const | 除标量 |
| CVect3 operator/(const CVect3 &v) const | 除向量（对应元素相除） |
| CVect3& operator+=(const CVect3 &v) | +=运算 |
| CVect3& operator-=(const CVect3 &v) | -=运算 |
| CVect3& operator\*=(double f) | \*=标量 |
| CVect3& operator/=(double f) | /=标量 |
| CVect3& operator/=(const CVect3 &v) | /=向量（对应元素相除） |
| friend CVect3 operator\*(double f, const CVect3 &v) | 标量乘向量 |
| friend CVect3 operator-(const CVect3 &v) | 向量取反 |
| friend double norm(const CVect3 &v) | 求模运算 |
| friend double normXY(const CVect3 &v) | 求XY分量模 |
| friend CVect3 sqrt(const CVect3 &v) | 分别对每一分量开方 |
| friend CVect3 pow(const CVect3 &v, int k=2) | 分别对每一分量求k次方 |
| friend double dot(const CVect3 &v1, const CVect3 &v2) | 点乘运算 |
| friend CVect3 dotmul(const CVect3 &v1, const CVect3 &v2) | 点乘，同Matlab的“.\*”运算功能 |
| friend CQuat rv2q(const CVect3 &v) | 等效旋转矢量转换为四元数 |
| friend CVect3 q2rv(const CQuat &q) | 变换四元数转换为等效旋转矢量 |
| friend CMat3 a2mat(const CVect3 &att) | 欧拉角转换成姿态阵（欧拉角按“右前上312”方式） |
| friend CVect3 m2att(const CMat3 &Cnb) | 姿态阵转换成欧拉角 |
| friend CQuat a2qua(double pitch, double roll, double yaw) | 欧拉角转换成四元数 |
| friend CQuat a2qua(const CVect3 &att) | 欧拉角转换成姿态四元数 |
| friend CVect3 q2att(const CQuat &qnb) | 姿态四元数转换成欧拉角 |
| friend CMat3 askew(const CVect3 &v) | 求向量的反对称阵 |
| friend CMat3 pos2Cen(const CVect3 &pos) | 由纬经度位置向量计算位置矩阵 |
| friend CVect3 pp2vn(const CVect3 &pos1, const CVect3 &pos0, double ts=1.0, CEarth \*pEth=NULL) | 由两点位置差分计算平均速度 |
| friend CVect3 MKQt(const CVect3 &sR, const CVect3 &tau) | 求一阶马尔可夫过程的激励噪声方差 |
| friend CMat3 dv2att(CVect3 &va1, const CVect3 &va2,  CVect3 &vb1, const CVect3 &vb2) | 由双矢量确定姿态阵 |
| friend CVect3 Alignsb(CVect3 wmm, CVect3 vmm, double latitude) | 静基座粗对准 |
| friend double MagYaw(const CVect3 &mag, const CVect3 &att, double declination=0) | 求磁航向角（利用att中水平姿态信息、可补偿磁偏角declination） |
| friend CVect3 xyz2blh(const CVect3 &xyz) | ECEF地心坐标转换为纬经高 |
| friend CVect3 blh2xyz(const CVect3 &blh) | 纬经高转换为ECEF地心坐标 |
| friend CVect3 Vxyz2enu(const CVect3 &Vxyz, const CVect3 &pos) | ECEF地心坐标速度转换为东北天速度 |
| friend CVect3 randn(const CVect3 &mu, const CVect3 &sigma) | 生成元素是正态分布的三维向量 |
| friend CVect3 sort(const CVect3 &v) | 元素从大到小排序 |

注：（1）大多数类的成员函数都定义成全局变量，虽然封装性不好，但方便操作；（2）很多友元函数可以定义成单独的函数，但放在类中只是为了方便归类。后同。

## 5.2四元数

|  |  |
| --- | --- |
| **class CQuat** | **四元数类** |
| *double q0, q1, q2, q3* | 四元数的四个分量，其中q0为标量系数 |
| CQuat(void) | 默认构造函数，不进行任何操作 |
| CQuat(double qq0, double qq1=0.0, double qq2=0.0, double qq3=0.0) | 对四个分量初始化 |
| CQuat(const double \*pdata) | 通过数组指针数据对三个分量初始化 |
| CQuat operator+(const CVect3 &phi) const | （真实）四元数加失准角误差 |
| CQuat operator-(const CVect3 &phi) const | （计算）四元数减失准角误差 |
| CVect3 operator-(CQuat &quat) const | 计算四元数减真实四元数得失准角误差 |
| CQuat operator\*(const CQuat &q) const | 两四元数相乘 |
| CVect3 operator\*(const CVect3 &v) const | 四元数乘矢量，进行坐标变换 |
| CQuat& operator\*=(const CQuat &q) | \*=运算 |
| CQuat& operator-=(const CVect3 &phi) | （计算）四元数-=失准角误差 |
| void normlize(CQuat \*q) | 归一化 |
| void SetYaw(double yaw=0.0) | 将四元数代表的方位角设置为指定值 |
| friend CQuat operator~(const CQuat &q) | 共轭/转置 |
| friend CVect3 qq2phi(const CQuat &qcalcu, const CQuat &qreal) | 计算四元数与真实四元数之间失准角 |
| friend CQuat UpDown(const CQuat &q) | 天向轴上下对调 |

## 5.3三维矩阵

|  |  |
| --- | --- |
| **class CMat3** | **三维矩阵类** |
| *double e00, e01, e02, e10, e11, e12, e20, e21, e22* | 定义矩阵的9个元素 |
| CMat3(void) | 默认构造函数，不进行任何操作 |
| CMat3(double xyz) | 将9矩阵9个元素都初始化xyz |
| CMat3(double xx, double yy, double zz) | 将对角线元素分别初始化，非对角线元素都为0 |
| CMat3(double xx, double xy, double xz,  double yx, double yy, double yz,  double zx, double zy, double zz ) | 对9个分量初始化 |
| CMat3(const CVect3 &v0, const CVect3 &v1, const CVect3 &v2,  BOOL isrow=1) | 通过三个向量对矩阵初始化（isrow=1则按行，否则按列） |
| CMat3 operator+(const CMat3 &m) const | 加运算 |
| CMat3 operator-(const CMat3 &m) const | 减运算 |
| CMat3 operator\*(const CMat3 &m) const | 乘运算 |
| CMat3 operator\*(double f) const | 矩阵乘标量 |
| CVect3 operator\*(const CVect3 &v) const | 乘三维矢量 |
| void SetRow(int i, CVect3 &v) | 将第i行(i=0,1,2)设置为向量 |
| void SetClm(int i, CVect3 &v) | 将第i列设置为向量 |
| CVect3 GetRow(int i) const | 取第i行向量 |
| CVect3 GetClm(int i) const | 取第i列向量 |
| friend CMat3 rcijk(const CMat3 &m, int ijk) | 重新按顺序排列行列 |
| friend CMat3 operator-(const CMat3 &m) | 矩阵取反 |
| friend CMat3 operator~(const CMat3 &m) | 矩阵转置 |
| friend CMat3 operator\*(double f, const CMat3 &m) | 标量乘矩阵 |
| friend CMat3 pow(const CMat3 &m, int k) | 幂方 |
| friend CQuat m2qua(const CMat3 &Cnb) | 姿态阵转四元数 |
| friend CMat3 q2mat(const CQuat &qnb) | 四元数转姿态阵 |
| friend double trace(const CMat3 &m) | 求迹 |
| friend double det(const CMat3 &m) | 计算矩阵行列式值 |
| friend CMat3 adj(const CMat3 &m) | 求伴随矩阵 |
| friend CMat3 inv(const CMat3 &m) | 求逆 |
| friend CVect3 diag(const CMat3 &m) | 提取矩阵对角线元素构成三维向量 |
| friend CMat3 diag(const CVect3 &v) | 由三维向量构造对角矩阵 |
| friend CMat3 MMT(const CMat3 &m1, const CMat3 &m2=I33) | 计算m1\*m2^T |
| friend CMat3 sfoam(const CMat3 &B, int iter=50) | 采用SFOAM算法进行单位正交化 |
| friend CMat3 randn(const CMat3 &mu, const CMat3 &sigma=CMat3(1.0)) | 生成元素是正态分布3x3矩阵 |

## 5.4多维向量

|  |  |
| --- | --- |
| **class CVect** | **N维向量类** |
| *int row, clm, rc*  *double dd[MMD]* | 向量行、列数目，总元素数目rc=row\*clm  向量元素存在数组中，元素数目最多为宏定义MMD |
| CVect(void); | 默认构造函数，不进行任何操作 |
| CVect(int row0, int clm0=1) | 对向量行、列数目初始化，默认为列向量 |
| CVect(int row0, double f) | 初始化列向量，所有元素值都赋值为f |
| CVect(int row0, double f, double f1, …) | 对列向量每一元素初始化 |
| CVect(int row0, const double \*pf) | 初始化列向量，并用数组赋值 |
| CVect(const CVect3 &v) | 用一个三维列向量初始化，结果三维列向量 |
| CVect(const CVect3 &v1, const CVect3 v2) | 用两个三维列向量初始化，结果六维列向量 |
| void Set(double f, ...) | 对每个元素赋值，注意：务必先初始化向量维数 |
| void Set2(double f, ...) | 对每个元素赋平方值，注意：务必先初始化向量维数 |
| CVect operator+(const CVect &v) const | 加运算 |
| CVect operator-(const CVect &v) const | 减运算 |
| CVect operator\*(double f) const | 乘标量运算 |
| CVect operator+=(const CVect &v) | +=运算（比加运算快） |
| CVect operator-=(const CVect &v) | -=运算（比减运算快） |
| CVect operator\*=(double f) | \*=标量运算（比乘标量运算快） |
| CVect operator\*(const CMat &m) const | 行向量乘矩阵 |
| CMat operator\*(const CVect &v) const | 行向量乘列向量、或列向量乘行向量 |
| double& operator()(int r) | 取第r个元素 |
| friend double dot(const CVect &v1, const CVect &v2) | 点乘运算 |
| friend CVect dotmul(const CVect &v1, const CVect &v2) | 点乘，同Matlab的“.\*”运算 |
| friend CVect operator~(const CVect &v) | 向量转置（行向量转列向量，或反之） |
| friend CVect abs(const CVect &v) | 每个元素求绝对值 |
| friend double norm(const CVect &v) | 求模 |
| friend double normInf(const CVect &v) | ∞范数，即返回绝对值最大的元素 |
| friend CVect pow(const CVect &v, int k=2) | 分别对每一分量求k次方 |
| friend CVect randn(const CVect &mu, const CVect &sigma=CVect(MMD,1.0)) | 生成元素是正态分布的向量 |
| friend CVect sort(const CVect &v) | 元素从大到小排序 |

## 5.5多维矩阵

|  |  |
| --- | --- |
| **class CMat** | **MXN维矩阵类** |
| *int row, clm, rc*  *double dd[MMD2]* | 矩阵行、列数，总元素数目rc=row\*clm  矩阵元素存在数组中，元素数目最多为宏定义MMD2=MMD\*MMD |
| CMat(void) | 默认构造函数，不进行任何操作 |
| CMat(int row0, int clm0) | 对矩阵行、列数目初始化 |
| CMat(int row0, int clm0, double f) | 初始化矩阵行、列数目，并将所有元素值都赋为f |
| CMat(int row0, int clm0, const double \*pf) | 初始化矩阵行、列数目，并用数组赋值 |
| void Clear(void) | 将所有元素清0 |
| void SetDiag(double f, ...) | 对方阵对角线元素赋值，注意：务必先初始化矩阵维数 |
| void SetDiag2(double f, ...) | 对方阵对角线元素赋平方值，注意：务必先初始化矩阵维数 |
| CMat operator+(const CMat &m) const | 加运算 |
| CMat operator-(const CMat &m) const | 减运算 |
| CMat operator\*(double f) const | 乘标量运算 |
| CVect operator\*(const CVect &v) const | 矩阵乘列向量 |
| CMat operator\*(const CMat &m) const | 矩阵乘法 |
| CMat& operator+=(const CMat &m0) | +=运算（比加运算快） |
| CMat& operator+=(const CVect &v) | +=对角阵运算（向量v转换为对角阵） |
| CMat& operator-=(const CMat &m0) | -=运算（比减运算快） |
| CMat& operator\*=(double f) | \*=标量运算（比乘标量运算快） |
| CMat& operator++() | 前置自增（加单位阵） |
| double& operator()(int r, int c) | 取第r行、c列元素 |
| void ZeroRow(int i) | 设置第i行向量为0 |
| void ZeroClm(int j) | 设置第j列向量为0 |
| oid SetRow(int i, double f, ...) | 设置第i行向量每一元素 |
| void SetRow(int i, const CVect &v) | 通过向量v设置第i行向量 |
| void SetClm(int j, double f, ...) | 设置第i列向量每一元素 |
| void SetClm(int j, const CVect &v) | 设置第j列向量 |
| CVect GetRow(int i) const | 取第i行向量 |
| CVect GetClm(int j) const | 取第j列向量 |
| void SetRowVect3(int i, int j, const CVect3 &v) | 设置i行j…(j+2)列元素为三维行向量v |
| void SetClmVect3(int i, int j, const CVect3 &v) | 设置i…(i+2)行j列元素为三维列向量v |
| void SetDiagVect3(int i, int j, const CVect3 &v) | 设置i…(i+2)行j…(j+2)列的对角线元素为三维列向量v |
| void SetMat3(int i, int j, const CMat3 &m) | 设置i…(i+2)行j…(j+2)列元素为三维矩阵m |
| CMat3 GetMat3(int i, int j) const | 获取i…(i+2)行j…(j+2)列元素为三维矩阵 |
| void SubAddMat3(int i, int j, const CMat3 &m) | i…(i+2)行j…(j+2)列元素加三维矩阵m |
| friend CMat operator~(const CMat &m) | 方阵转置 |
| friend void symmetry(CMat &m) | 方阵对称化 |
| friend double normInf(CMat &m) | ∞范数，即返回绝对值最大的元素 |
| friend CMat dotmul(const CMat &m1, const CMat &m2) | 点乘，同Matlab的“.\*”运算 |
| friend CVect diag(const CMat &m) | 取矩阵对角线元素 |
| friend CMat diag(const CVect &v) | 由向量构造对角矩阵 |
| friend void RowMul(CMat &m, const CMat &m0,  const CMat &m1, int r) | 仅计算前矩阵m0的第r行乘后矩阵m1，即m(r,:)=m0(r,:)\*m1 |
| friend void RowMulT(CMat &m, const CMat &m0,  const CMat &m1, int r) | 仅计算前矩阵m0的第r行乘后矩阵m1转置，即m(r,:)=m0(r,:)\*m1’ |
| friend void DVMDVafa(const CVect &V, CMat &M,  double afa=1.0) | 向量化对角阵再与矩阵乘再扩大afa倍，即diag(V)\*M\*diag(V)\*afa |

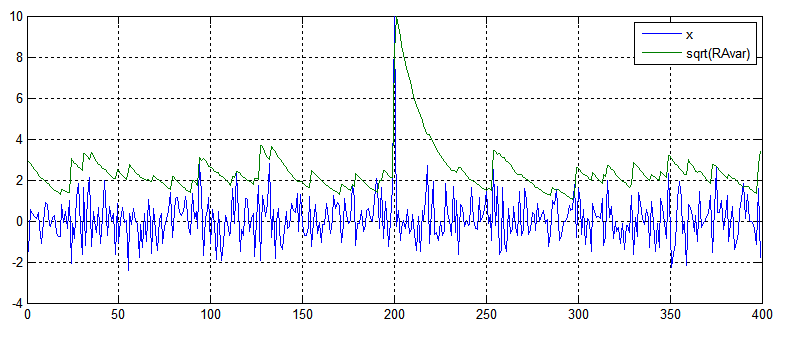
# 6一些辅助数据处理类

## 6.1 Allan方差法统计序列的方差

算法详见《惯性仪器测试与数据分析》式（9.4-29）、《捷联惯导算法与组合导航原理》式（6.6.10）。

|  |  |
| --- | --- |
| **class CRAvar** | **Allan方差法统计序列的方差** |
| *int nR0, maxCount, Rmaxflag[RAMAX];*  *double ts, R0[RAMAX], Rmax[RAMAX], Rmin[RAMAX],*  *tstau[RAMAX], r0[RAMAX];* | 方差数组的数目、超限计数、超限标记  序列采样周期、方差、方差上限、方差下限  序列采样周期/衰减时间比、保存上一时刻序列值 |
| CRAvar(int nR0, int maxCount0=2) | 初始化方差数组数目和超限计数 |
| void set(double r0, double tau, double rmax=0.0, double rmin=0.0, int i=0) | 设置第i个统计元素的方差、时间常数、方差上限、方差下限初值（默认方差上限=100\*初值、方差下限=0.01\*初值） |
| void set(const CVect3 &r0, const CVect3 &tau, const CVect3 &rmax=O31,  const CVect3 &rmin=O31) | 用三维向量设置统计量 |
| void set(const CVect &r0, const CVect &tau, const CVect &rmax=On1,  const CVect &rmin=On1) | 用n维向量设置统计量 |
| void Update(double r, double ts, int i=0) | 利用采样序列进行方差更新 |
| void Update(const CVect3 &r, double ts) | 同上（用三维向量） |
| void Update(const CVect &r, double ts) | 同上（用n维向量） |
| double operator()(int k) | 读取第k个方差元素 |

一个CRAvar ravr(1); ravr.set(3.0, 10.0, 10.0, 0.1)的示例如下图。

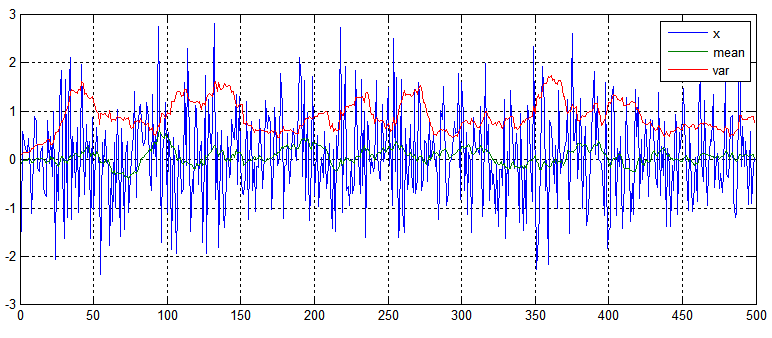


## 6.2直接统计序列的均值和方差

设置一个数组存储最近的n个数，每次更新时都重新计算一遍这n个数的均值和方差。

|  |  |
| --- | --- |
| **class CVAR** | **统计序列的均值和方差** |
| *float mean, var;* | 均值、方差 |
| CVAR(int imax0=10, float data0=0.0) | 初始化，设置数组元素最大个数和初值 |
| float Update(float data, BOOL isvar=TRUE) | 方差更新，方差是否更新标志（若不更新则只算均值）  注意：当元素个数比较多时计算量偏大 |

一个CVAR(20)的示例如下图。

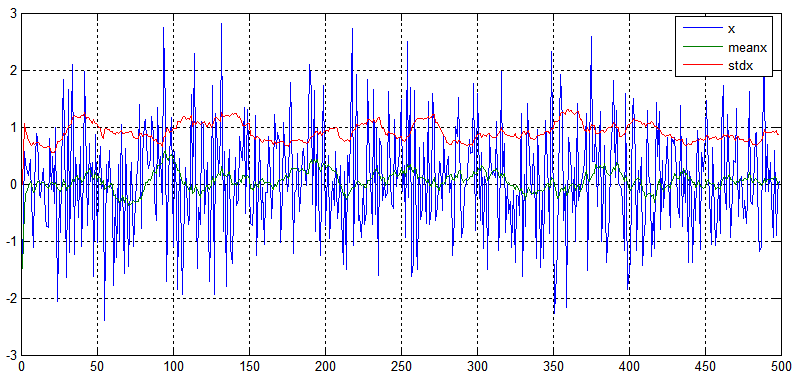


## 6.3基于移入移出法的序列统计

设置一个二维数组存储最近的row行\*clm列个数，每次更新时加入最近的数据统计值，同时减去最老的数据统计值，这一统计方法的计算量相对较小些。

|  |  |
| --- | --- |
| **class CVARn** | **统计序列** |
| *double \*Sx, \*Sx2, \*mx, \*stdx;* | 分别为sum(x), sum(x^2), mean(x), std(x) |
| CVARn(int row0, int clm0) | 初始化，分配row0行clm0列个double元素的内存，按列统计 |
| void Reset(void) | 所有内存和统计量重置为0 |
| BOOL Update(const double \*pd)  BOOL Update(double f, ...) | 更新，注意输入元素的绝对值不应大于1.0e5，否则作截断处理。  如果更新次数达到最大的row0返回1，否则返回0  注意：巨多次更新后可能会因舍入误差累积而偏真实统计值 |

一个CVARn(20,1)的示例如下图。



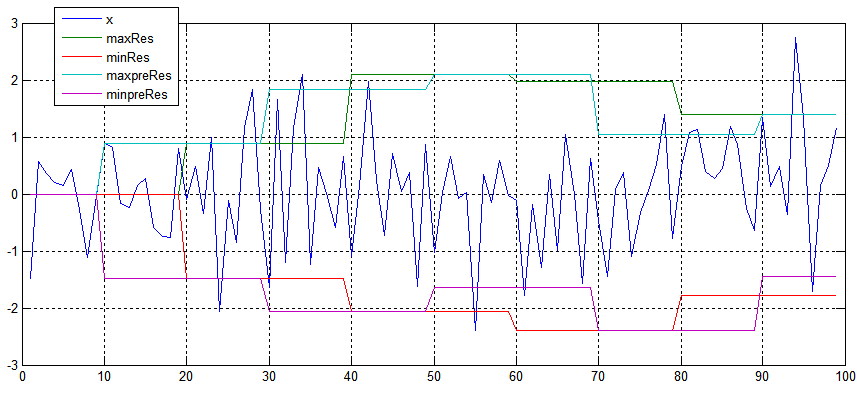
## 6.4最大最小值统计

给出最近n个数的最大最小值。

|  |  |
| --- | --- |
| **class CMaxMin** | **最大最小值统计** |
| *double maxRes, minRes, maxpreRes, minpreRes;* | 最大值，最小值，前一最大值，前一最小值 |
| CMaxMin(int cnt00=100, int pre00=0, double f0=0.0) | 初始化，统计最近的n个值，使n=cnt00，一般pre00=cnt00/2， |
| int Update(double f) | 数据更新  当返回1时，最大值/最小值可用  当返回-1时，前一最大值/前一最小值可用  当返回0时，统计值未更新 |

CMaxMinn是CMaxMin的多维版本。

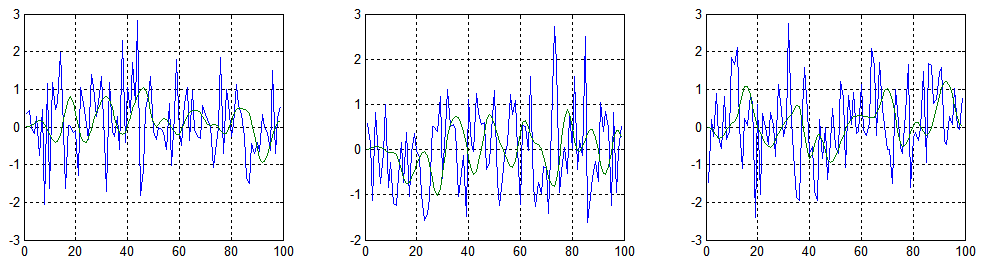
一个CMaxMin(20, 10)的示例如下图。当Update返回1时，过去的20个值都在minRes~maxRes之间变化；当返回-1时，过去的20个值都在minpreRes~maxpreRes之间变化。



## 6.5 IIR数据滤波器

|  |  |
| --- | --- |
| **class CIIR** | **IIR滤波器类**  （CIIRV3类为CIIR类的3维向量版） |
| *int n;*  *double b[10], a[10], x[10], y[10];* | 滤波器阶次  b,a滤波器系数，x,y存储滤波过程中间值 |
| CIIR(double \*b0, double \*a0, int n0) | 利用数组初始化滤波器系数，滤波器系数可通过Matlab/fdatool工具生成 |
| double Update(double x0) | 得到一个新的数据输入，进行滤波更新 |

对一个三维向量序列的滤波示例如下图（蓝色为标准正态噪声，绿色为滤波输出）。



## 6.6环型数组存储类

|  |  |
| --- | --- |
| **class CRMemory** | **环型数组存储类** |
| *BYTE \*pMemPush, \*pMemPop, pushBuf[128], popBuf[128]* | 存指针、读指针、存缓存、读缓存 |
| CRMemory(BYTE \*pMem, long memLen0, BYTE recordLen0=0) | pMem存储空间，memLen0存储空间长度（字节数）  recordLen0 =0每次读写为可变字节数  =n每次读写为固定字节数n |
| BOOL push(const BYTE \*p=NULL) | 存入数据，如果recordLen0 =0则p[0]为字节数；如果存入溢出则作覆盖处理并返回0 |
| BYTE pop(BYTE \*p=NULL) | 读出数据，如果recordLen0 =0则p[0]为字节数；如果没有数可读则返回0 |

# 7捷联惯导算法类

## 7.1地球参数计算

|  |  |
| --- | --- |
| **class CEarth** | **地球参数计算类** |
| *double a, b;*  *double f, e, e2;*  *double wie;*  *double sl, sl2, cl, tl, RMh, RNh, clRNh, f\_RMh, f\_RNh, f\_clRNh;*  *CVect3 pos, vn, wnie, wnen, wnin, gn, gcc, ganomaly;* | 长半轴、短半轴  扁率、偏心率/平方  自转角速率  纬度正弦、余弦等成员变量 |
| CEarth(double a0=glv.Re, double f0=glv.f, double g0=glv.g0) | 初始化 |
| void Update(const CVect3 &pos, const CVect3 &vn=O31) | 根据地理位置和惯导速度更新参数 |
| void SetAnomaly(const CVect3 &anomaly0=O31) | 设置重力异常 |
| CVect3 vn2dpos(const CVect3 &vn, double ts=1.0) | 由速度计算位置增量 |

## 7.2惯性测量单元数据处理

|  |  |
| --- | --- |
| **class CIMU** | **惯性测量单元类** |
| *CMat3 \*pgSens, gSens, \*pgSens2, gSens2, \*pgSensX, gSensX;*  *CVect3 \*pKa2, Ka2;*  *char \*prfu, rfu[3];*  *int nSamples;*  *CVect3 phim, dvbm, wm\_1, vm\_1;* | 陀螺重力敏感性参数矩阵  加计比例因数二次方项  IMU指向（如果不是右-前-上的话）  子样数  补偿后等效旋转矢量、比力增量、保存前一次角/速度增量采样 |
| CIMU(void) | 初始化 |
| void SetgSens(const CMat3 &gSens0,  const CMat3 &gSens20=O33, const CMat3 &gSensX0=O33) | 设置陀螺重力敏感性参数矩阵 |
| void SetKa2(const CVect3 &Ka20) | 设置加计比例因数二次方项 |
| void SetRFU(const char \*rfu0) | 设置IMU指向 |
| void Update(CVect3 \*wm, CVect3 \*vm, int nSamples) | 不可交换误差补偿更新 |
| friend void IMURFU(CVect3 \*pwm, CVect3 \*pvm, int nSamples,  const char \*str) | 将其它方向安装的IMU转换为载体“右前上”方向 |
| friend void IMUStatic(CVect3 &wm, CVect3 &vm,  CVect3 &att0, CVect3 &pos0, double ts=1.0) | 仿真静态IMU的陀螺/计加增量信息 |

## 7.3捷联惯导算法

|  |  |
| --- | --- |
| **class CSINS** | **捷联惯导算法类** |
| *double ts, nts, tk;*  *CEarth eth;*  *CIMU imu;*  *CQuat qnb;*  *CMat3 Cnb, Cnb0, Cbn, Kg, Ka;*  *CVect3 wib, fb, fn, an, web, wnb, att, vn, pos, eb, db, vb,*  *tauGyro, tauAcc, \_betaGyro, \_betaAcc;*  *CMat3 Maa, Mav, Map, Mva, Mvv, Mvp, Mpv, Mpp;*  *CVect3 vnL, posL; CMat3 CW, MpvCnb;*  *CQuat qnbE; CVect3 attE, vnE, posE;* | 相关成员变量 |
| CSINS(const CQuat &qnb0=qI, const CVect3 &vn0=O31,  const CVect3 &pos0=O31, double tk0=0.0) | 初始化 |
| void SetTauGA(const CVect3 &tauG, const CVect3 &tauA) | 陀螺零漂、加表零偏Markov相关时间 |
| void Update(CVect3 \*wm, CVect3 \*vm, int nSamples, double ts) | 惯导更新 |
| void Extrap(const CVect3 &wm=O31, const CVect3 &vm=O31,  double ts=0.0) | 子样内短时惯导外推 |
| void Extrap(double extts) | 利用最后子样采样进行短时惯导外推 |
| void lever(const CVect3 &dL=O31) | 计算杆臂参数、杆臂速度和杆臂位置 |
| void etm(void) | 计算惯导误差传播转移系数 |

## 7.4 AVP插值

|  |  |
| --- | --- |
| **class CAVPInterp** | **AVP插值** |
| *CVect3 att, vn, pos* | 插值姿态、速度、位置 |
| void Init(const CSINS &sins)  void Init(const CVect3 &att0, const CVect3 &vn0, const CVect3 &pos0, double ts) | 初始化 |
| void Push(const CSINS &sins)  void Push(const CVect3 &attk, const CVect3 &vnk=O31, const CVect3 &posk=O31) | 压入更新最新的AVP |
| int Interp(double tpast) | 插值计算过去tpast时刻的AVP |

# 8卡尔曼滤波类

|  |  |
| --- | --- |
| **class CKalman** | **卡尔曼滤波器**  （假设系统噪声和量测噪声均为对角阵） |
| *double kftk, zfdafa;*  *int nq, nr, measflag, measflaglog, measstop;*  *CMat Ft, Pk, Hk, Fading;*  *CVect Xk, Zk, Qt, Rt, rts, RtTau, measlost, Xmax, Pmax, Pmin, Zfd,*  *Rmax, Rmin, Rbeta, Rb,*  *FBTau, FBMax, FBXk, FBTotal* | 滤波计时、量测遗忘因子  状态维数、量测维数、量测标识、量测输出标识、停止量测计数  系统矩阵（连续时间）、方差阵、量测矩阵、遗忘矩阵  状态、量测、过程噪声（连续时间）、量测噪声、量测周期、量测遗忘时间、量测丢失时间、状态上限、方差阵上/下限、Zfd量测方差强跟踪门限  量测自适应方差上/下限、遗忘参数beta/b  反馈时间常数、反馈上限、当前反馈值、总反馈量 |
| CKalman(int nq0, int nr0) | 初始化滤波器维数 |
| virtual void Init(void) =0 | 初始化系统噪声\量测噪声\状态方差阵等，需重载 |
| void SetRmaxcount(int cnt=5) | 自适应量测方差超Rmax后，设置之后的连续量测无效个数 |
| virtual void SetFt(void) =0 | 设置系统矩阵，需重载 |
| virtual void SetHk(void) =0 | 设置量测矩阵，需重载 |
| virtual void SetMeas(void) =0 | 设置量测，需重载 |
| virtual void Feedback(double fbts) | 反馈修正 |
| void RtFading(int i, double fdts) | 量测方差遗忘 |
| void TimeUpdate(double kfts, int fback=1) | 时间更新 |
| void MeasUpdate(double fading=1.0) | 量测更新（采用序贯滤波，默认遗忘因子为1） |
| void RAdaptive(int i, double r, double Pr) | Sage-Husa量测自适应 |
| void RPkFading(int i) | 量测方差强跟踪 |
| void SetMeasFlag(int flag) | 设置量测标记 |
| void XPkConstrain(void) | 状态估计、方差阵限制（使-*Xmax* <=*Xk<= Xmax*、Pmin<=diag(Pk)<=Pmax） |
| friend void fusion(double \*x1, double \*p1,  const double \*x2, const double \*p2,  int n=9, double \*xf=NULL, double \*pf=NULL)  friend void fusion(CVect3 &x1, CVect3 &p1,  const CVect3 x2, const CVect3 p2)  friend void fusion(CVect3 &x1, CVect3 &p1,  const CVect3 x2, const CVect3 p2,  CVect3 &xf, CVect3 &pf) | 信息融合 |

# 9组合导航类

|  |  |
| --- | --- |
| **class CSINSTDKF** | **捷联惯导组合时间分散滤波算法**  （继承自CSINSKF类，引入该类的目的为了使算法能在嵌入式系统上实时应用） |
| *double tdts;*  *int iter, ifn, measRes;*  *CMat Fk, Pk1;*  *CVect Pxz, Qk, Kk, Hi, tmeas;*  *CVect3 meanfn;* | 相关成员变量 |
| CSINSTDKF (int nq0, int nr0) | 初始化组合滤波器 |
| void Init(const CSINS &sins0) | 初始化时间分片组合滤波器 |
| void TDReset(void) | 滤波重置 |
| void MarkovGyro(const CVect3 &tauG, const CVect3 &sRG, int stateeb=9) | 陀螺马尔可夫过程设置 |
| void MarkovAcc(const CVect3 &tauA, const CVect3 &sRA, int statedb=12) | 加计马尔可夫过程设置 |
| void SetYaw(double yaw, int statephi=0, int statedvn=3) | 重置滤波器的方位角 |
| int TDUpdate(CVect3 \*wm, CVect3 \*vm,  int nSamples, double ts, int nStep=1) | 组合滤波器更新（含时间和量测更新：把滤波更新分散成2\*(nq+nr)+3个时间片/步，在每个惯导更新周期内需保证nStep步能运行完；每步大约运行0.5ms，但不同处理器间存在差异） |
| virtual void RTOutput(void) | 用户可重载导航实时输出，在惯导解算之后，滤波更新之前，实时性会好些 |

|  |  |
| --- | --- |
| **class CSINSGNSS** | **捷联惯导/GNSS组合导航组合** |
| *double posGNSSdelay, vnGNSSdelay, yawGNSSdelay, dtGNSSdelay, kfts*  *CVect3 lvGNSS;*  *CAVPInterp avpi;* | 相对于IMU的GNSS定位延迟、速度延迟、定向延迟，滤波器时间更新时间间隔  GNSS杆臂（右前上相对于IMU）  存储最近前几拍的惯导数据，用于后向插值 |
| CSINSGNSS(int nq0, int nr0, double ts) | 构造函数（适用于15-34维状态，1-6维量测通常为速度误差和位置误差）  方位角量测常用于低精度MEMS车载系统 |
| void Init(const CSINS &sins0, int grade=-1) | 初始化组合滤波器 |
| virtual void SetFt(int nnq);  virtual void SetHk(int nnq);  virtual void Feedback(int nnq, double fbts); | 构造系统阵  构造量测阵  滤波反馈 |
| virtual void SetMeas(void) | 被CSINSTDKF::TDUpdate调用 |
| void SetMeasGNSS(const CVect3 &pgnss=O31,  const CVect3 &vgnss=O31) | 设置位置、速度量测值（如果为0则表示无效） |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts) | 导航更新 |

|  |  |
| --- | --- |
| **class CSINSGNSSDR** | **捷联惯导/GNSS/DR组合导航组合** |
| *CVect3 posDR;*  *CMat3 Cbo*  *double Kod, gnsslost* | 航位推算位置  里程计安装矩阵（车体系相对于IMU系）  里程计标度系数，GPS丢失时间 |
| CSINSGNSSDR(double ts) | 一个21维的SINS/GNSS/DR组合导航构造函数例子（惯导15维状态+里程计3维+DR位置误差3维，13维量测，SINS/GNSS、SINS/DR、GNSS/DR两两组合） |
| void Init(const CSINS &sins0, int grade=-1) | 初始化组合滤波器 |
| virtual void SetFt(int nnq);  virtual void SetHk(int nnq);  virtual void Feedback(int nnq, double fbts); | 构造系统阵  构造量测阵  滤波反馈 |
| virtual void SetMeas(void) | 被CSINSTDKF::TDUpdate调用 |
| void SetMeasGNSS(const CVect3 &pgnss=O31,  const CVect3 &vgnss=O31, double yawgnss=0.0) | 设置GNSS位置、速度、双天线方位量测值（如果为0则表示无效） |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm, double dS,  int nSamples, double ts) | 导航更新，dS为里程仪位移增量 |

# 10初始对准算法类

## 10.1惯性系初始对准算法

惯性系初始对准方法可用于静态、干扰基座和载体运动状态下的初始对准。（1）静基座（微小干扰）下能达到较好的效果，可直接作为精对准使用；（2）大干扰基座下可能能达不到精对准目的（不是十分可靠，与干扰时机有关），最好只是作为粗对准使用，之后再利用Kalman滤波对准方法，或直接进行组合导航；（3）动基座下需要外界速度辅助，利用载体系速度（里程计/多普勒）或导航系速度（GNSS速度），通常也只是作为粗对准使用。

|  |  |
| --- | --- |
| **class CAligni0** | **惯性系初始对准类** |
| *double tk;*  *CQuat qnb0, qnb, qnbsb;* | 对准时间  对准初始时刻姿态阵（四元数），对准结束时刻姿态阵，解析对准方法的结果 |
| CAligni0(const CVect3 &pos0=O31, const CVect3 &vel0=O31,  int velAid=0)  void Init(const CVect3 &pos0=O31, const CVect3 &vel0=O31,  int velAid=0); | 初始化  pos0 初始位置，vel0 初始速度  velAid0 =0为无速度辅助（准静态）；  =1为n系速度辅助；  =2为b系速度辅助 |
| CQuat Update(const CVect3 \*pwm, const CVect3 \*pvm, int nSamples,  double ts, const CVect3 &vel=O31) | 对准更新  pwm 角增量，pvm速度增量，nSamples子样数，ts采样周期，vel辅助速度（辅助方式见velAid0）  返回对准姿态四元数qnb |

注：（1）如果对准时间短，小于100个更新周期则qnb输出CQuat(1.0)；小于1000个更新周期则qnb输出qnbsb；（2）也可以用与低精度的MEMS对准，但得到的只有水平姿态角是准的，方位角没意义。

## 10.2卡尔曼滤波初始对准算法

采用速度为量测的Kalman滤波初始对准算法，可用于静态和动基座。在动基座时需输入外界参考速度。

|  |  |
| --- | --- |
| **class CAlignkf** | **卡尔曼滤波初始对准类** |
| *int mvnk;*  *double mvnts, mvnT;*  *CVect3 mvn, pos0;* | 平均速度计数  平均速度时间  平均速度，对准的初始位置 |
| CAlignkf(double ts)  CAlignkf(const CSINS &sins0, double ts)  void Init(const CSINS &sins0) | 初始化，通过sins0设置粗对准姿态、初始速度和位置 |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts)  int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts, const CVect3 &vnr) | 静基座对准更新，pwm 角增量，pvm速度增量，nSamples子样数，ts采样周期。算法中静基座的速度量测更新频率为10Hz，量测值为100ms内的平均值（在振动时比采用瞬时速度效果好）  动基座对准更新，vnr为外界参考速度（比如GNSS速度），当无GNSS速度时需取vnr=O31 |

## 10.3基于航迹角的动态粗对准算法

针对车辆或固定翼飞机等运载体，假设水平姿态角不大，通过GNSS输出的参考速度（vnr）计算航迹方向近似为载体航向角，进行粗略动基座初始对准。

|  |  |
| --- | --- |
| **class CAligntrkang** | **动态粗对准算法** |
| *int cntYawOK;*  *CQuat qnb;* | 有效航迹计数（GNSS速度大于一定值2m/s，且当前航迹与惯导更新航迹差别不大<30°）  对准结果姿态四元数 |
| CAligntrkang(double vel00=1.0, double wz00=5.0\*DPS,  double dyaw00=5.0\*DEG);  void Init(const CSINS &sins0=CSINS(O31,O31,O31)); | 初始化，通过sins0设置粗对准姿态、初始速度和位置（通常为均为0即可） |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts, const CVect3 &vnr); | 对准更新，pwm 角增量，pvm速度增量，nSamples子样数，ts采样周期。vnr为GNSS量测速度，并通过该速度计算航迹角。  返回有效航迹计数cntYawOK，可用于判断粗对准完成情况 |

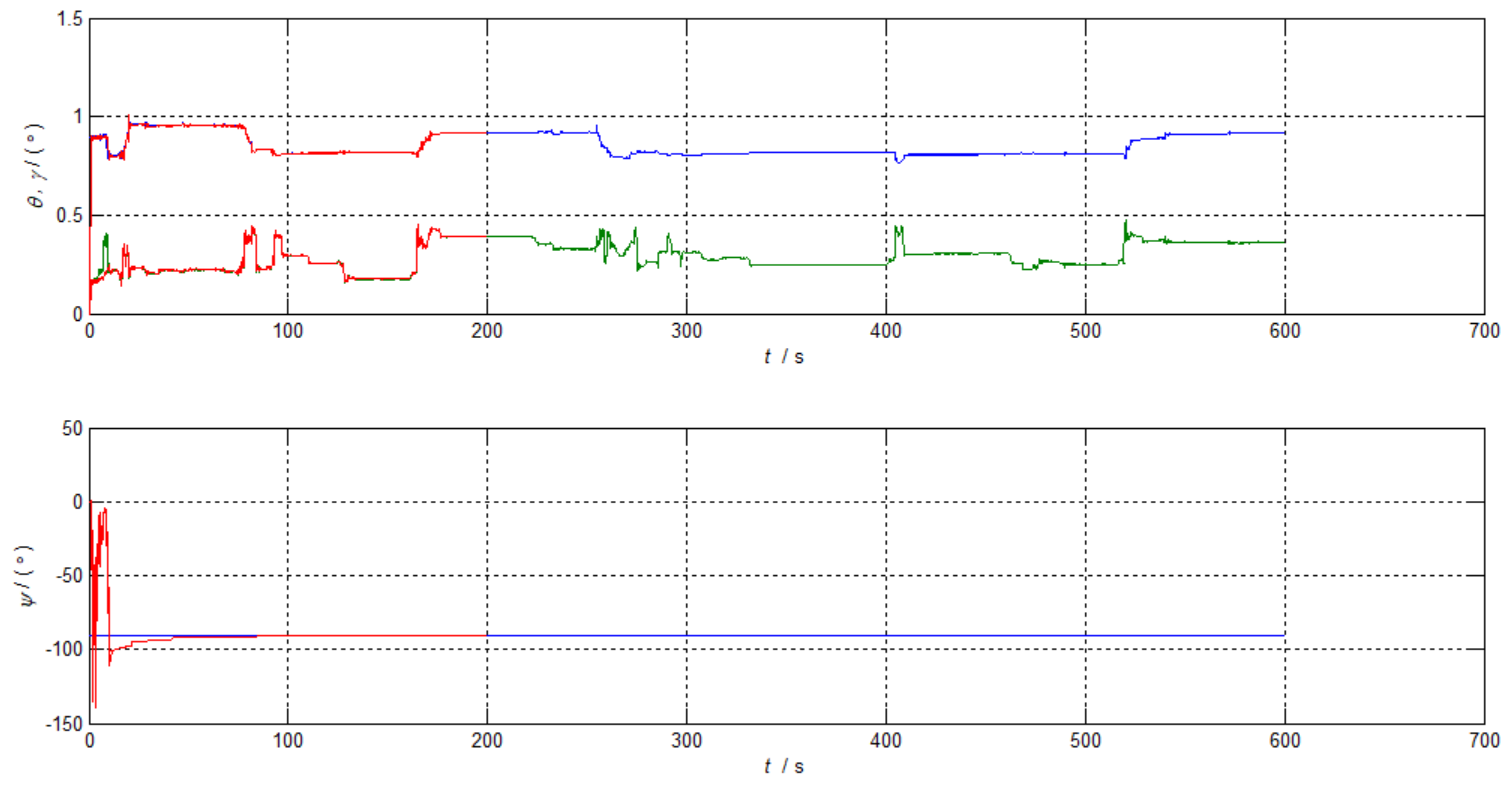
## 10.4基于数据存储技术的初始对准算法

先采用惯性系对准方法CAligni0粗对准T1时间，并纯储惯组采样数据，获得初始时刻的姿态四元数qnb0，再采用Kalman滤波对准方法CAlignkf从头开始精对准，慢慢追赶上最近的惯组采样数据T2时刻。

|  |  |
| --- | --- |
| **class CAlignsv** | **数据存储初始对准类** |
| *double t, tk, ts, T1, T2*  *BOOL alnkfinit*  *CAligni0 alni0*  *CRMemory \*pMem* | 内部计算时间t，与惯组采样时刻对应的对准时间tk，ts采样间隔，粗对准时间T1，精对准时间T2（注意10<T1<T2/2）  对准输出的姿态四元数  粗对准完成，进入kf精对准标识  惯性系粗对准方法  惯性数据存储指针 |
| CAlignsv(const CVect3 &pos, double ts,  double T2=300.0, double T1=0.0)  void Init(const CVect3 &pos, double ts,  double T2=300.0, double T1=0.0) | 初始化，pos为初始对准的地理位置，其它各参数含义见前 |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm) | 对准更新，pwm 角增量，pvm速度增量，这里只支持单子且采样间隔ts相同  当对准时间超过T2时，返回1，继续执行精对准过程。 |

注：采用数据存储技术的初始对准需要分配较大的内存，内存需求量为int(T1/ts)\*6\*sizeof(double) 字节，100Hz采样存储100s约为500k bytes，这在嵌入式系统中使用应当注意。

一个对准示例如下图，其中红色为粗对准结果。

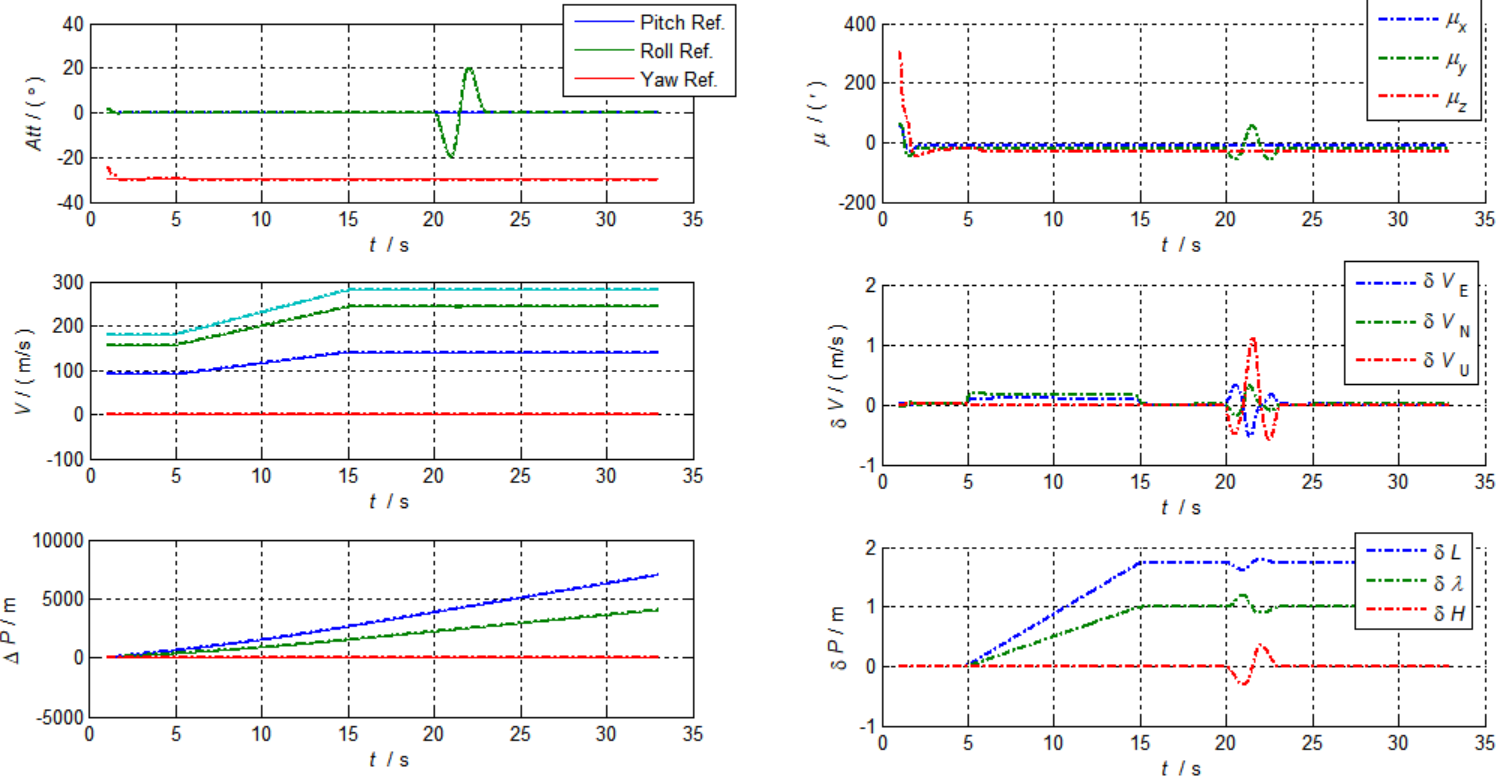


## 10.5“速度+姿态”匹配传递对准算法

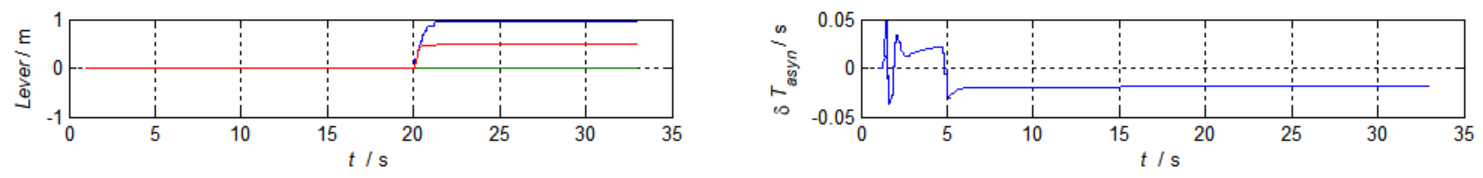
该类继承自CSINSGNSS，设计了19维状态：失准角3维、速度误差3维、主子惯导安装误差角3维、陀螺漂移3维、加表零偏3维、杆臂3维、时间不同步1维。特别注意：该类将安装角误差状态分量7-9替换了CSINSGNSS中的位置误差分量。

|  |  |
| --- | --- |
| **class CAligntf** | **数据存储初始对准类** |
| *CVect3 mu, lvMINS;*  *double dtMINSdelay;* | 主子惯导之间的安装误差角、杆臂  时间不同步（<0，主惯导总是滞后的） |
| CAligntf(double ts);  CAligntf(const CSINS &sins0, double ts);  void Init(const CSINS &sins0); | 初始化，通常以主惯导的参数直接初始化 |
| virtual void SetFt(int nnq);  virtual void SetHk(int nnq); | 建立系统矩阵  建立量测矩阵 |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts, int nSteps=5) | 时间更新 |
| void SetMeasVnAtt(const CVect3 &vnMINS=O31,  const CVect3 &attMINS=O31); | “速度+姿态”量测更新 |

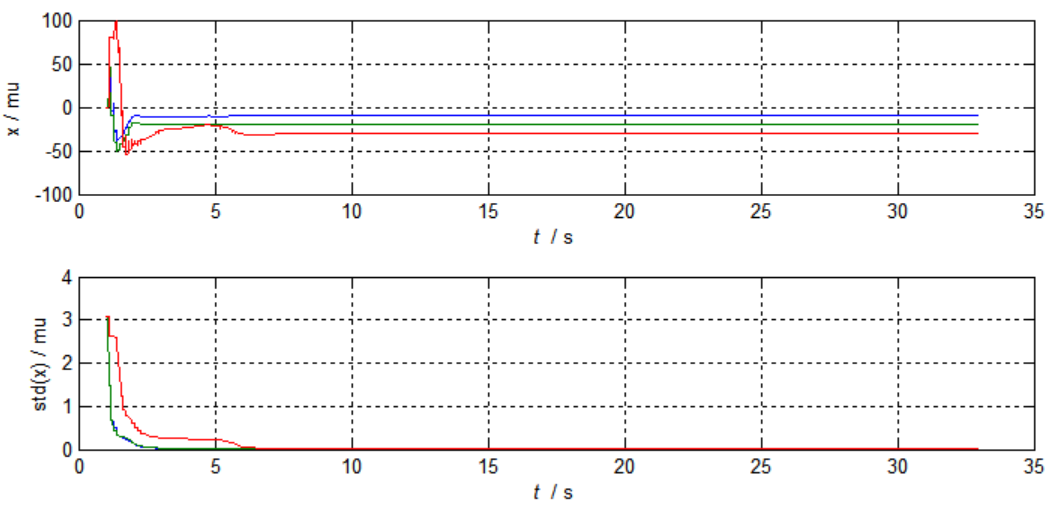
一个传递对准示例如下图，其中轨迹线做加速运动，再做摇翼运动。



传递对准运动轨迹及估计误差



杆臂及不同步误差估计



主子惯导安装误差估计及其均方差

# 11低成本MEMS航姿类

|  |  |
| --- | --- |
| **class CMahony** | **低成本MEMS-AHRS算法类** |
| *double tk, Kp, Ki;*  *CQuat qnb;*  *CMat3 Cnb;*  *CVect3 exyzInt;* | 相关成员变量 |
| CMahony(double tau=4.0) | 初始化，tau为控制算法的时间常数用于计算PI参数。 |
| void SetTau(double tau) | 设置时间参数，tau为控制算法的时间常数用于计算PI参数。 |
| void Update(const CVect3 &gyro, const CVect3 &acc,  const CVect3 &mag, double ts) | AHRS算法更新，注意：gyro的单位为°/s；acc和mag为归一化单位或任意单位 |

# 12文件读写

## 12.1通用文件读写

通用文件读写类用于文件输入输出处理。（1）用于读入时，读入文本文件或二进制double型数据记录文件，文件每行的记录数（列数）应当相同；（2）用于输出时，都统一当作double型数据输出，生成的数据文件可用Matlab的data=fread(fid,[column,inf],’doulbe’)’;语句进行读取。

文件读写类只在PC机中应用，不能应用于嵌入式系统。

|  |  |
| --- | --- |
| **class CFileRdWt** | **文件读写类** |
| static void Dir(const char \*dirI, const char \*dirO=NULL) | 设置文件读、写路径，若只设置一个参数则dirO=dirI |
| CFileRdWt(const char \*fname0, int columns0=0) | 文件初始化：  fname0文件名；  columns0=0写二进制文件；columns0>0读文本文件，columns0表示文件列数；columns0<0读二进制文件，-columns0表示文件列数。 |
| int load(int lines=1) | 读入一行（文本文件或double型二进制文件），lines>1时只读最后一行（即跳过前面lines-1行数据）。  读取成功返回1，否则文件结束返回0 |
| int loadf32(int lines=1) | 读入一行float32型二进制文件  读取成功返回1，否则文件结束返回0 |
| CFileRdWt& operator<<(double d) | 输出double型（二进制） |
| CFileRdWt& operator<<(const CVect3 &v) | 输出三维向量（二进制） |
| CFileRdWt& operator<<(const CVect &v) | 输出N维向量（二进制） |
| CFileRdWt& operator<<(const CMat &m) | 输出矩阵（二进制） |
| CFileRdWt& operator<<(const CRAvar &R) | 输出CRAvar型（二进制） |
| CFileRdWt& operator<<(const CAligni0 &aln) | 输出惯性系初始对准结果（二进制） |
| CFileRdWt& operator<<(const CSINS &sins) | 输出惯导的姿态、速度、位置、陀螺常漂、加计零偏估计 |
| CFileRdWt& operator<<(const CMahony &ahrs) | 输出航姿仪的姿态、陀螺漂移估计 |
| CFileRdWt& operator<<(const CQEAHRS &ahrs) | 输出航姿仪的姿态、陀螺漂移估计 |
| CFileRdWt& operator<<(const CKalman &kf) | 输出卡尔曼滤波器的状态Xk、方差阵Pk和Rk对角线元素、滤波标识位 |
| CFileRdWt& operator>>(double &d); | 读入double型（二进制） |
| CFileRdWt& operator>>(CVect3 &v); | 读入三维向量（二进制） |
| CFileRdWt& operator>>(CVect &v); | 读入N维向量（二进制） |
| CFileRdWt& operator>>(CMat &m); | 读入矩阵（二进制） |
| BOOL waitfor(int columnk, double val=0.0,  double eps=EPS) | 等待某列某数据值的出现 |
| void bwseek(int lines, int mod=SEEK\_CUR) | 后退指定行 |
| long filesize(int opt=1) | 获取文件大小 |

## 12.2用户定制文件读取

可用于定制用户特定的文件读取，比如不是统一按double/float类型纯储的二进制文件，或数据需要进行一些单位换算或坐标轴顺序调整等。

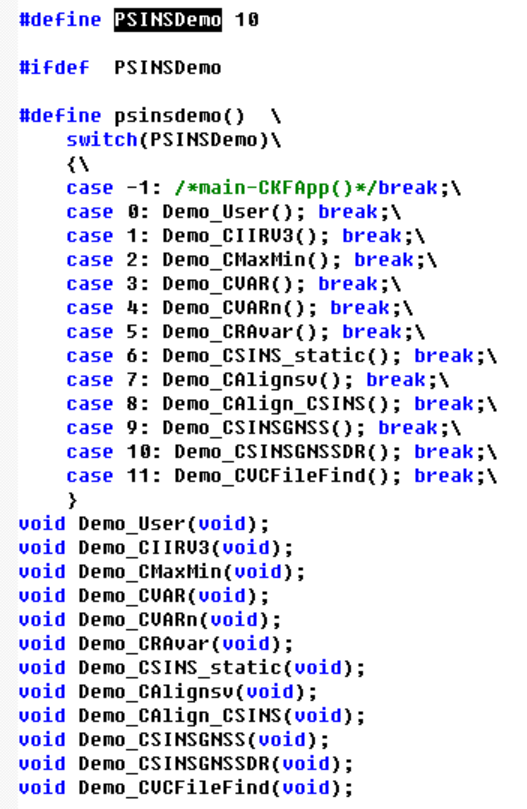
|  |  |
| --- | --- |
| **class CFileRdSr** | **用户定制文件读取类** |
| *DataSensor \*pDS* | 用户自定义的数据文件记录结构体 |
| int load(int lines=1, BOOL txtDelComma=1) | 须用户编写的特定读取方式 |

注：该类一般放置在用户的KFApp.h文件中，而不直接在PSINS.h文件中修改。

# 13演示举例

所有演示列子都放在PSINS\_Demo.cpp文件中，通过修改宏定义PSINSDemo可选择不同的例子，如果该宏定义小于0，则表示不使用该演示文件函数，而实用main\_SinsGps.cpp中的主函数。

C++运行完成后，其处理结果可通过PSINS\_Demo\_figure.m绘图，直观显示。



注：有些例子需要Data\lasergyro.imu数据支持（一组激光捷联惯导车载干扰基座下的采集数据）；运行完每个例子后，其结果输出保存在Data文件夹下，可以使用Data\PSINS\_Demo.m中的Matlab相应命令进行绘图查看。