**PSINS高精度捷联惯导C++程序说明**

西北工业大学 严恭敏

版本时间2018-08-10，2020-04-09

**一、基础类文件（PSINS.h、PSINS.cpp）**

表1

|  |  |
| --- | --- |
| **宏定义、全局变量/函数** |  |
| #define PSINS\_MATRIX\_MAX\_DIM xx  #define MAT\_COUNT\_STATISTIC  #define PSINS\_IO\_FILE  #define PSINS\_RMEMORY  #define PSINS\_AHRS\_MEMS  #define PSINS\_LOW\_GRADE\_MEMS  #define CSINSGPSOD\_INIT\_BY\_USER  #define PSINS\_psinsassert  #define PSINS\_UART\_PUSH\_POP  #define PSINS\_VC\_AFX\_HEADER | 定义最大矩阵维数，使用滤波器时务必根据需要修改定义xx！  统计CMat类的栈调用个数，可用于分析堆栈是否够用  定义PSINS文件操作类，嵌入式系统中一般不用  定义PSINS内存操作类，一般用于回溯和数据存储  定义航姿类，多用于低精度MEMS航姿仪  定义低精度惯导，不考虑地球自转影响  SINS/GPS/OD组合类，需用户自己初始化  PSINS断言函数  PSINS串口组帧操作类  包含VC6.0的’afx.h’头文件  以上选择性编译宏定义，如果注释了则不进行相应的代码编译 |
| #define PI 3.14159265358979  #define PI\_2 (PI/2.0)  #define PI\_4 (PI/4.0)  #define \_2PI (2.0\*PI)  #define DEG (PI/180.0)  #define DPS (DEG/1.0) // deg/s  #define SEC (DEG/3600.0) // arcsec  #define DPH (DEG/3600.0) // deg/h  #define G0 9.7803267714  #define UG (G0/1.0e6) // ug  #define RE 6378137.0  #define PPM 1.0e-6  #define EPS 2.22044604925031e-16  #define INF 3.402823466e+30F | PI  PI/2  PI/4  2\*PI  度  度/秒  角秒  度/小时  1倍重力值  Ug  地球长半轴  ppm（百万分之一）  零阈值  无穷大 |
| #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  标量平方  北偏西方位角(-PI~PI)转为北偏东方位角(0~2PI)  前一定义的逆操作  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 psinsassert(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;  extern int psinslasterror | 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）  最后发生错误全局变量  特别注意：除此处外，用户不宜/小心定义更多其它可能包含这些变量的全局变量/函数（由于C++全局类变量初始化顺序不可控）！ |

表2

|  |  |
| --- | --- |
| **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 dps, dph, dpsh, dphpsh, ugpsh, ugpsHz,*  *ugpg2, mpsh, mpspsh, secpsh* | °/s、°/h、°/sqrt(h)、°/h/sqrt(h)、ug/ sqrt(h)、ug/ sqrt(Hz)  ug/g2、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 |

表3

|  |  |
| --- | --- |
| **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 CVect3 abs(const CVect3 &v)  friend double norm(const CVect3 &v)  double normInf(const CVect3 &v) | 各元素绝对值  求模运算  无穷范数（绝对值最大元素） |
| 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 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 |

表4

|  |  |
| --- | --- |
| **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)  friend CVect3 qq2phi(const CQuat &qcalcu, const CQuat &qreal); | 将四元数代表的方位角设置为指定值  计算四元数-参考四元数=失准角 |
| friend CQuat operator~(const CQuat &q) | 共轭/转置 |
| friend CQuat UpDown(const CQuat &q) | 四元数代表的天向轴上下对调 |

表5

|  |  |
| --- | --- |
| **class CMat3** | **三维矩阵类** |
| *double e00, e01, e02, e10, e11, e12, e20, e21, e22* | 定义矩阵的九个元素 |
| CMat3(void) | 默认构造函数，不进行任何操作 |
| CMat3(double xx, double xy, double xz,  double yx, double yy, double yz,  double zx, double zy, double zz ) | 对九个分量初始化 |
| CMat3(const CVect3 &v0, const CVect3 &v1, const CVect3 &v2) | 通过三个行向量对矩阵初始化 |
| 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 |
| 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 CQuat m2qua(const CMat3 &Cnb) | 姿态阵转四元数 |
| friend CMat3 q2mat(const CQuat &qnb)  friend double trace(const CMat3 &m) | 四元数转姿态阵  求迹 |
| friend double det(const CMat3 &m)  friend CMat3 adj(const CMat3 &m) | 计算矩阵行列式值  伴随阵 |
| friend CMat3 inv(const CMat3 &m) | 求逆 |
| friend CVect3 diag(const CMat3 &m) | 由矩阵对角线元素构造三维向量 |
| friend CMat3 diag(const CVect3 &v)  friend CMat3 pow(const CMat3 &m, int k)  friend CMat3 foam(const CMat3 &B, int iter=50) | 由三维向量构造对角矩阵  矩阵的幂方  姿态阵优化FOAM算法 |

表6

|  |  |
| --- | --- |
| **class CVect** | **N维向量类** |
| *int row, clm, rc*  *double dd[MMD]* | 向量行、列数目（1xN或Nx1），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  CVect& operator=(double f)  CVect& operator=(const double \*pf) | 乘标量运算  等于同一数  等于一数组对应元素 |
| 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，不需赋值过程） |
| 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 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)  friend double norm(const CVect &v) | 各元素的绝对值  求模 |
| friend CVect pow(const CVect &v, int k=2) | 分别对每一分量求k次方 |

表7

|  |  |
| --- | --- |
| **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 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=-1) | 取第r行、c列元素（若c=-1则表示r行r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)  CVect3 GetDiagVect3(int i, int j=-1); | 设置i…(i+2)行j…(j+2)列的对角线元素为三维列向量v  获取i…(i+2)行j…(j+2)列对角线元素为三维向量 |
| 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) | 无穷范数（矩阵元素中绝对值最大者） |
| 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 |

表8

|  |  |
| --- | --- |
| **class CRAvar** | **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 |

表9

|  |  |
| --- | --- |
| **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;*  *int nq, nr, measflag;*  *CMat Ft, Pk, Hk, Fading;*  *CVect Xk, Zk, Qt, Rt, rts, Xmax, Pmax, Pmin, Zfd*  *Rmax, Rmin, Rbeta, Rb,*  *FBTau, FBMax, FBXk, FBTotal* | 滤波计时  状态维数、量测维数、量测标记  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、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 |
| 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 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 RtFading(int i, double fdts) | 量测噪声自遗忘 |
| 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（摆设） |
| 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） |
| void fusion(double \*x1, double \*p1, const double \*x2,  const double \*p2, int n, double \*xf, double \*pf) | 信息融合xf = (p2\*x1+p1\*x2)/(p1+p2), pf = p1\*p2/(p1+p2)  n<100则数组前三分量表示姿态；>100为普通量  注意不连续量的特殊处理，比如经度在+-180°附近跳变时 |

表10

|  |  |
| --- | --- |
| **class CSINSKF** | **捷联惯导组合**  （继承自CKalman类） |
| *CSINS sins* | 捷联惯导类成员 |
| CSINSKF(int nq0, int nr0) | 初始化组合滤波器 |
| virtual void Init(void) | 空初始化，不进行任何操作 |
| virtual void Init(const CSINS &sins0, int flag) | 初始化系统噪声\量测噪声\状态方差阵等，如有必要则重载 |
| virtual void SetFt(int nnq=15) | 设置系统矩阵，如有必要则重载 |
| virtual void SetHk(int nnq=15) | 设置量测矩阵，如有必要则重载 |
| virtual void Feedback(double fbts) | 反馈修正，如有必要则重载 |
| void Update(const CVect3 \*wm, const CVect3 \*vm,  int nSamples, double ts) | 组合更新 |
| void QtMarkovGA(const CVect3 &tauG, const CVect3 &sRG,  const CVect3 &tauA, const CVect3 &sRA) | 陀螺零漂、加表零偏建模成Markov过程 |

表11

|  |  |
| --- | --- |
| **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 TDReset(void) | 重置滤波器 |
| 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） |

表12

|  |  |
| --- | --- |
| **class CSINSGPSOD** | **SINS/GPS/OD组合导航滤波器** |
| *CVect3 posDR;*  *CMat3 Cbo, MpkD;*  *double Kod, tODInt;*  *BOOL measGPSvnValid, measGPSposValid, measODValid, measMAGyawValid;* | 相关成员变量 |
| CSINSGPSOD(void); | 构造函数（21维状态=惯导误差15维+DR位置误差3维+俯仰/方位安装误差2维+OD刻度系数误差1维，10维量测=GPS位置3维+GPS速度3维+DR位置3维+磁方位角1维，方位角量测常用于低精度MEMS车载系统） |
| virtual void Init(const CSINS &sins0, int grade=-1) | 初始化组合滤波器 |
| void SetMeasGPS(const CVect3 &pgps=O31,  const CVect3 &vgps=O31);  void SetMeasOD(double dSod, double ts);  void SetMeasYaw(double ymag); | 设置GPS位置、速度量测  设置里程计速度增量量测  设置磁航向量测 |
| int Update(const CVect3 \*pwm, const CVect3 \*pvm,  int nSamples, double ts); | 滤波更新 |

表13

|  |  |
| --- | --- |
| **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 |

表14

|  |  |
| --- | --- |
| **class CIMU** | **惯性测量单元类** |
| *int nSamples;*  *CVect3 phim, dvbm, wm\_1, vm\_1;* | 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 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 |

表15

|  |  |
| --- | --- |
| **class CAligni0** | **惯性系初始对准类** |
| *double tk;*  *CQuat qnb0, qnb;* | 对准时间  对准初始时刻姿态阵，对准结束时刻姿态阵 |
| CAligni0(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 \*wm, const CVect3 \*vm, int nSamples,  double ts, const CVect3 &vel=O31) | 对准更新  wm 角增量，vm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 |

表16

|  |  |
| --- | --- |
| **class CAlignkf** | **Kalman滤波初始对准类** |
| *int mvnk;*  *double mvnts;*  *CVect3 mvn, pos0;* | 速度平均时间计数  量测速度平均时间间隔  平均速度、初始位置 |
| CAlignkf(void);  virtual void Init(const CSINS &sins0); | 初始化 |
| 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) | 对准更新  wm 角增量，vm速度增量，nSamples子样数，ts采样周期  vnr为参考速度，可用于动基座初始对准 |

表17

|  |  |
| --- | --- |
| **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 lever(const CVect3 &dL=O31) | 计算杆臂参数、杆臂速度和杆臂位置 |
| void etm(void) | 计算惯导误差转移系数 |

表18

|  |  |
| --- | --- |
| **class CAVPInterp** | **AVP插值类**  （多用于SINS/GPS组合，量测做差需要数十毫秒以前的惯导AVP姿态/速度/定位数据） |
| *#define AVPINUM 10*  *public:*  *double ts;*  *int ipush;*  *CVect3 atti[AVPINUM], vni[AVPINUM], posi[AVPINUM];*  *CVect3 att, vn, pos;* | 缓存数据组数，根据需要定义长度  相关成员变量 |
| CAVPInterp(double ts=0.01) | 初始化，ts为数据组之间的间隔（必须为等间隔） |
| void Push(const CVect3 &attk, const CVect3 &vnk=O31,  const CVect3 &posk=O31) | 压AVP入缓存 |
| int Interp(double tpast); | 求解以前tpast时刻的AVP |

表19

|  |  |
| --- | --- |
| **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 |
| void Update(const CVect3 &wm, const CVect3 &vm,  double ts, const CVect3 &mag=O31); | 功能同上，但此处wm和vm均为增量信息 |

表20

|  |  |
| --- | --- |
| **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 |
| CFileRdWt(void) | 构造函数（空） |
| 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。 |
| void Init(const char \*fname0, int columns0=0) | 功能同上 |
| 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 |
| long load(BYTE \*buf, long bufsize) | 往内存中读入一整块文件数据 |
| int getl(void) | 读入一行（文本文件） |
| long filesize(int opt=1) | 返回文件大小字节数，opt=1剩余未读大小/opt=0全文件大小 |
| 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 CQuat &q) | 输出四元数（二进制） |
| CFileRdWt& operator<<(const CMat3 &m) | 输出三维方矩阵（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) | 输出滤波器的状态Xk、方差阵Pk对角线元素、滤波状态 |
| CFileRdWt& operator<<(const CRMemory &m) | 输出内存数据 |
| CFileRdWt& operator<<(const CUartPP &uart) | 输出串口接收到的数据 |
| 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>>(CQuat &q) | 读入四元数（二进制） |
| CFileRdWt& operator>>(CMat3 &m) | 读入三维方矩阵（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） |
| char\* time2fname(void) | 返回一个以当前时间命名的文件名字符串，形如：  “PSINSyymmdd\_HHMMSS.bin” |

表21

|  |  |
| --- | --- |
| **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(long recordNum, int recordLen0) | 初始化recordNum条记录，每条记录recordLen0个字节 |
| 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 |
| BYTE\* get(int iframe) | 在固定字节数帧方式中，返回第iframe帧首地址 |

表22

|  |  |
| --- | --- |
| **class CIIR**  **class CIIRV3** | **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的fdatool工具生成 |
| double Update(double x0) | 输入一个新的数据，进行滤波更新 |

表23

|  |  |
| --- | --- |
| **class CUartPP** | **串口接收组包类**  （可用于嵌入式串口应用中的IMU固定帧长数据接收组包） |
| *#define UARTFRMLEN (50\*4)*  *#define UARTBUFLEN (UARTFRMLEN\*20)*  *unsigned char head[2], popbuf[UARTFRMLEN], buf[UARTBUFLEN], chksum;*  *int pushIdx, popIdx, frameLen, overflow, getframe;*  *int csflag, cs0, cs1, css;* |  |
| CUartPP(int frameLen0, unsigned short head0=0x55aa) | 初始化（帧长度，二字节帧头） |
| BOOL checksum(const unsigned char \*pc) | 帧校验 |
| int push(const unsigned char \*buf0, int len) | 接收到len个字节，入栈 |
| int pop(unsigned char \*buf0=(unsigned char\*)NULL) | 查找是否有完整一帧数据，有则出栈 |

**二、应用类文件（KFApp.h、KFApp.cpp）**

一般继承自CSINSTDKF类，由用户自行定义并实现……

**三、算法参考资料**

[1]严恭敏.捷联惯导算法与组合导航原理,西北工业大学出版社,2019.

[2]严恭敏.传统组合导航中的实用Kalman滤波技术评述,导航定位与授时,2020.