PSINS捷联惯导C++程序说明

严恭敏，2017-05-04

**一、基础类文件（PSINS.h、PSINS.cpp）**

表1

|  |  |
| --- | --- |
| **全局变量与宏定义** |  |
| #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 123456789.0e100 | PI  PI/2  PI/4  2\*PI  零阈值  无穷大 |
| #define MMD 16  #define MMD2 (MMD\*MMD) | 向量/矩阵的最大维数定义（用户根据需要修改）  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 CC180toC360(yaw)  #define C360toCC180(yaw) | 扩展反正弦，防止输入绝对值大于1  扩展反余弦  求最大  求最小  北偏西方位角(-PI~PI)转为北偏东方位角(0~2PI)  前一定义的逆 |
| 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 |
| const CVect3 I31, O31;  const CQuat qI;  const CMat3 I33, O33;  const CGLV glv; | 三维元素均为1的向量，三维零向量  单位四元数  三维单位矩阵，三维零矩阵  全局变量 |

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 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) | 对地球参数初始化 |

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 xx, double yy=0.0, double zz=0.0) | 对三个分量初始化 |
| CVect3(const double \*pdata) | 通过数组数据对三个分量初始化 |
| BOOL IsZero(double eps=EPS) const | 判断是否为零向量（三个元素均为零） |
| BOOL IsZeroXY(double eps=EPS) const | 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/(double f) const | 除标量 |
| 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 |
| 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 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 m2att(const CMat3 &Cnb) | 姿态阵转换成欧拉角（欧拉角右前上312方式） |
| 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 |

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 |
| friend CQuat operator~(const CQuat &q) | 共轭/转置 |
| friend CQuat a2qua(double pitch, double roll, double yaw) | 欧拉角转换成四元数（欧拉角右前上312方式） |
| friend CQuat a2qua(const CVect3 &att) | 欧拉角转换成四元数（欧拉角右前上312方式） |
| friend CQuat m2qua(const CMat3 &Cnb) | 姿态阵转换成四元数 |
| friend CVect3 q2rv(const CQuat &q) | 变换四元数转换为等效旋转矢量 |

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 a2qua(double pitch, double roll, double yaw) | 欧拉角转四元数 |
| friend CMat3 a2mat(const CVect3 &att) | 欧拉角向量转姿态阵 |
| friend CQuat a2qua(const CVect3 &att) | 欧拉角向量转四元数 |
| friend CVect3 m2att(const CMat3 &Cnb) | 姿态阵转欧拉角 |
| friend CQuat m2qua(const CMat3 &Cnb) | 姿态阵转四元数 |
| friend CVect3 q2att(const CQuat &qnb) | 四元数转欧拉角 |
| friend CMat3 q2mat(const CQuat &qnb) | 四元数转姿态阵 |
| friend double det(const CMat3 &m) | 计算矩阵行列式值 |
| 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) | 由三维向量构造对角矩阵 |
| friend CMat3 a2mat(const CVect3 &att) | 欧拉角转换成姿态矩阵（欧拉角右前上312方式） |
| friend CMat3 q2mat(const CQuat &qnb) | 四元数转换成姿态阵 |
| friend CMat3 dv2att(CVect3 &va1, const CVect3 &va2,  CVect3 &vb1, const CVect3 &vb2) | 由双矢量确定姿态阵 |

表6

|  |  |
| --- | --- |
| **class CVect** | **N维向量类** |
| *int row, clm*  *double dd[MMD]* | 向量行、列数目  向量元素存在数组中，元素数目最多为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 |
| CVect(int row0, double f) | 初始化列向量，所有元素值都赋为f |
| CVect(const CVect3 &v) | 用一个三维列向量初始化，结果三维 |
| CVect(const CVect3 &v1, const CVect3 v2) | 用两个三维列向量初始化，结果六维 |
| void Set(double f, ...) | 对每个元素赋值，务必先初始化向量维数 |
| void Set2(double f, ...) | 对每个元素赋平方值，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 CVect operator~(const CVect &v) | 向量转置（行向量转列向量，或反之） |
| friend double norm(const CVect &v) | 求模 |

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MMD2 |
| 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 |
| void SetDiag(double f, ...) | 对对角线元素赋值，务必先初始化向量维数 |
| void SetDiag2(double f, ...) | 对对角线元素赋平方值，务必先初始化向量维数 |
| 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 SetRow(int i, const CVect &v) | 设置第i行向量 |
| 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 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 |
| void ZeroRow(int i) | 设置第i行向量为0 |
| void ZeroClm(int j) | 设置第i列向量为0 |
| friend CMat operator~(const CMat &m) | 转置 |
| friend void symmetry(CMat &m) | 方阵对称化 |
| friend double MaxAbs(CMat &m) | 求矩阵元素绝对值的对大值 |
| friend CVect diag(const CMat &m) | 取对角线元素 |
| 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 |

表8

|  |  |
| --- | --- |
| **class CRAvar** | **统计序列的方差** |
| *int nR0, Rmaxflag[RAMAX];*  *double ts, R0[RAMAX], Rmax[RAMAX], Rmin[RAMAX],*  *tstau[RAMAX], r0[RAMAX];* | 方差数目、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, double ts) | 初始化方差数目和序列采样周期 |
| void setR0(double r0, ...) | 设置方差初值（默认方差上限=100\*初值、方差下限=0.01\*初值） |
| void setTau(double tau, ...) | 设置衰减时间常数，计算tstau=ts/tau |
| void setRmax(double rmax, ...) | 设置方差上限 |
| void setRmin(double rmin, ...) | 设置方差下限 |
| void Update(double r, ...) | 利用采样序列进行方差更新 |
| double operator()(int k) | 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 tk;*  *int nq, nr, measflag;*  *CMat Ft, Pk, Hk;*  *CVect Xk, Zk, Qt, Rk, Pmax, Pmin;* | 时间  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  状态、量测、过程噪声（连续时间）、量测噪声、方差阵上/下限 |
| CKalman(int nq0, int nr0) | 初始化滤波器维数 |
| void SetFt(CSINS &sins) | 设置系统矩阵 |
| void SetHk(void) | 设置量测矩阵 |
| void TimeUpdate(double ts) | 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 SetMeasFlag(int flag) | 设置量测标记 |
| void PkConstrain(void) | 方差阵限制（使Pmin<=diag(Pk)<=Pmax） |
| void Feedback(CSINS &sins, double tauphi=INF,  double taudvn=INF, double taudpos=INF,  double taueb=INF, double taudb=INF) | 反馈修正 |

表10

|  |  |
| --- | --- |
| **class CSINSKF** | **捷联惯导组合类**  （） |
| *double tk;*  *int nq, nr, measflag;*  *CMat Ft, Pk, Hk;*  *CVect Xk, Zk, Qt, Rk, Pmax, Pmin;* | 时间  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  状态、量测、过程噪声（连续时间）、量测噪声、方差阵上/下限 |
| CKalman(int nq0, int nr0) | 初始化滤波器维数 |
| void SetFt(CSINS &sins) | 设置系统矩阵 |
| void SetHk(void) | 设置量测矩阵 |
| void TimeUpdate(double ts) | 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 SetMeasFlag(int flag) | 设置量测标记 |
| void PkConstrain(void) | 方差阵限制（使Pmin<=diag(Pk)<=Pmax） |
| void Feedback(CSINS &sins, double tauphi=INF,  double taudvn=INF, double taudpos=INF,  double taueb=INF, double taudb=INF) | 反馈修正 |

表10

|  |  |
| --- | --- |
| **class CTDKF** | **时间分散卡尔曼滤波算法**  （引入该类的目的为了使算法能在嵌入式系统上实时应用） |
| *CVect tmeas* | 最近一次量测到来的时间记录 |
| CTDKF(int nq0, int nr0) | 初始化滤波器维数 |
| int TDUpdate(CSINS &sins, double ts, int nStep=1) | 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 SetMeas(CSINS &sins) | 多种量测信息之间逻辑判断（需用户在继承类中具体实现） |

表11

|  |  |
| --- | --- |
| **class CEarth** | **地球参数计算类** |
| *double a, b;*  *double f, e, e2, ep, ep2;*  *double wie;*  *double sl, sl2, sl4, cl, tl, RMh, RNh, clRNh, f\_RMh, f\_RNh, f\_clRNh;*  *CVect3 pos, vn, wnie, wnen, wnin, gn, gcc;* | 长半轴、短半轴  扁率、偏心率/平方、第二偏心率/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=CVect3(0.0)) | 根据地理位置和惯导速度更新参数 |
| CVect3 vn2dpos(const CVect3 &vn, double ts=1.0) | 由速度计算位置增量 |

表12

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

表13

|  |  |
| --- | --- |
| **class CAligni0** | **惯性系初始对准类** |
| CAligni0(CVect3 &pos) | 初始化 |
| CQuat Update(CVect3 \*wm, CVect3 \*vm, int nSamples, double ts) | 对准更新 |

表14

|  |  |
| --- | --- |
| **class CSINS** | **捷联惯导算法类** |
| *double 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;;*  *CVect3 phim, dvbm, wm\_1, vm\_1;* | 相关成员变量 |
| CSINS(const CQuat &qnb0=qI, const CVect3 &vn0=O31, const CVect3 &pos0=O31) | 初始化 |
| void Update(CVect3 \*wm, CVect3 \*vm, int nSamples, double ts) | 惯导更新 |
| void etm(CMat3 &Maa, CMat3 &Mav, CMat3 &Map,  CMat3 &Mva, CMat3 &Mvv, CMat3 &Mvp,  CMat3 &Mpv, CMat3 &Mpp) | 计算惯导误差转移系数 |

表15

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

表16

|  |  |
| --- | --- |
| **class CFileRdWt** | **文件读写类** |
| CFileRdWt(char \*dir0) | 设置文件读写路径 |
| CFileRdWt(char \*fname0, int columns) | 文件初始化：columns=-1读二进制文件；columns=0写二进制文件；columns>0读文本文件 |
| int load(int lines=1) | 读入一行（字符文件） |
| int IsEOF(void) | 检测文件是否结束 |
| 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 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与方差阵Pk对角线元素 |
| 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） |

**二、应用类文件（KFApp.h、KFApp.cpp）**

主要针对特定应用场合供用户编写具体应用类。目前提供的示例为低成本车载航姿仪类（CarAHRS）。车载航姿仪可供利用的传感器有IMU、GPS（可含双天线方位角）、磁强计、气压高度计和里程计等，还有载车静止零速修正条件（ZUPT，Zero velocity UpDaTe）和运动约束（MC，Moving Constraint）信息可利用，其中MC指载车行驶过程中车体右向和天向速度为零、车体倒车（后向）速度一般小于1m/s、前行速度一般小于39m/s（即140km/hur）。

数据处理的基本思路：

（1）只要载车在短时间内（t=5s）速度变化不大，可以采用惯性系对准方法粗略找到水平姿态角。假设5s前后载车速度变化不大于1m/s，则水平对准误差对大约为1/(g\*t)=1.2°。即使不用水平对准，由于载车一般停放时水平角都会小于10°，因而初始水平角直接赋零问题也可以。

（2）在水平对准后，如果磁强计测量信息可以（总磁场/水平分量测量值均与当地地磁大小偏差不大），可用地磁测量计算粗略方位角；或者，如果载车行驶速度大于2m/s，可利用速度计算载车行驶的航迹向，作为初始方位；如果有双天线GPS，那方位直接装定就可以了。

（3）行驶过程中如果GPS速度和定位可用，则进行INS/GPS量测组合；如果GPS不可以，则可用陀螺和加速度的方差（CRavar类）判断运动状态，停车时利用ZUPT测量，行驶中可利用CM约束。运动中不论使用何种量测，最后在机动相对较小时（加速度和转弯角速度不大）使用，有利于降低时间不同步和杆臂等误差影响。

要实现一个实用而有效的航姿仪算法并不容易，需要充分综合利用各种信息，只能通过实践不断完善。

表17

|  |  |
| --- | --- |
| **class CSensors** | **传感器类** |
| *int imuValid, nn; double imut, ts, nts; CVect3 wm[5], vm[5];*  *int gpsPosValid, satNum; double gpspt, PDOP; CVect3 gpsPos;*  *int gpsVnValid; double gpsvt; CVect3 gpsVn;*  *int gpsYawValid; double gpsyt, gpsYaw;*  *int odValid; double odt, dS;*  *int magValid; double magt; CVect3 mag;*  *int barValid; double bart, barHgt;* | 各类传感器的有效性标记、采集时间、数据等参数 |
| CSensors(void) | 初始化传感器类 |
| void clear(void) | 清除传感器采样标记 |
| int set(void) | 获取传感器采样值，需用户根据实际情况编写 |
| int set(CFileRdWt &f) | 读取文件获取传感器采样值 |

表18

|  |  |
| --- | --- |
| **class CCarAHRS** | **车载航姿仪类** |
| CCarAHRS(double ts) | 初始化滤波器，滤波器状态包括：失准角、速度误差、位置误差、陀螺常漂、加计常偏、z轴陀螺比例系数误差 |
| virtual void SetFt(CSINS &sins) | 设置系统矩阵 |
| void PreMeas(void) | 量测预处理 |
| virtual void SetMeas(CSINS &sins) | 设置/综合判断各种量测的使用 |
| void Update(CVect3 &wm, CVect3 &vm, double ts) | 航姿仪捷联解算和滤波更新 |

**三、与C的接口函数及主函数（IO2C.cpp、PSINSMain.cpp）**

表19

|  |  |
| --- | --- |
| **IO2C.cpp、PSINSMain.cpp** | **与C语言的接口函数及主函数** |
| extern "C" void PSINSInit(void)  extern "C" void PSINSSetIMU(double \*ga, double ts)  …  extern "C" void PSINSUpdate(int nStep)  extern "C" void PSINSOut(double \*avp) | 在Keil等平台中的c语言中调用PSINS算法，输入传感器输出并输出导航结果，用户根据需要编写 |
| int main(void) | PSINS库函数可作为为相对独立算法模块，如果在嵌入式Keil等平台中应用，主函数一般为c文件中的main函数，需将这里的main函数重新命名，比如main1，避免冲突 |



感谢“西安精准测控有限责任公司”提供实验测试数据，数据格式见数据文件的第一行文字说明。