数字卫星开发文档

软件框架

功能分级

Attitude类：姿态动力学、姿态控制相关内容

Orbit类：轨道动力学、姿态控制相关内容

Environment类：引力场、地磁场、太阳等相关内容

SimTime类：加速倍率、系统时间戳相关内容

Gyro类：陀螺敏感器相关内容

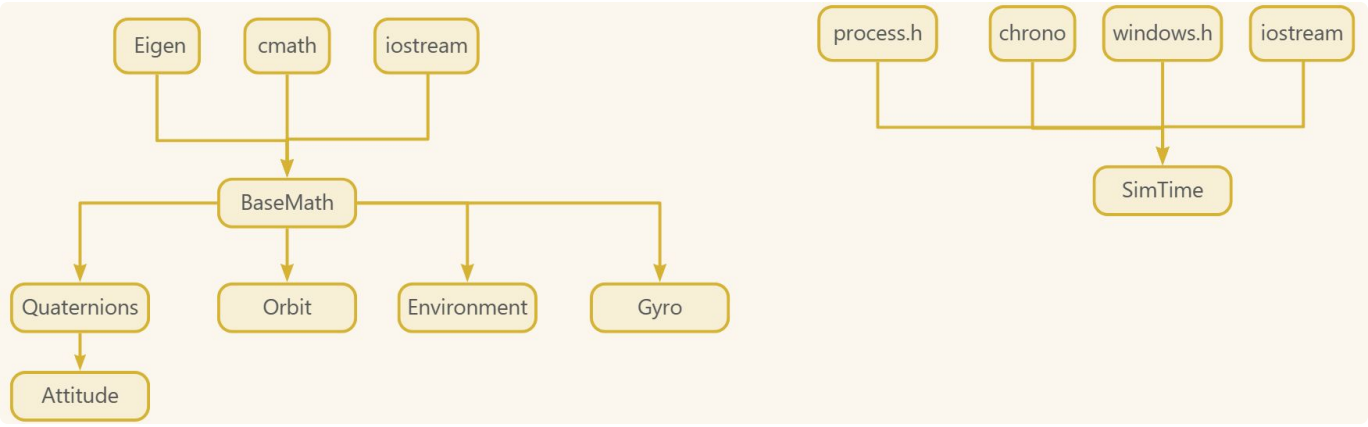
Satellite：卫星本体相关内容

BaseMath：基础数学常数、运算

Quaternions类：四元数相关定义和运算

Dcm类：旋转矩阵相关定义和运算

依赖关系



数学运算

四元数类 `Quat`

成员变量

`double QuatData[4]` 代表 q_0 ~ q_3 ，其数据一定具有如下性质[1:附录2]:

- JPL四元数，标部为 q_0 ，虚部为 q_1 ~ q_3
- $q_0 > 0$
- $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$
- 单位：无

成员函数

默认构造函数 `Quat()`

输入参数：无

返回：无

功能：以单位四元数 $(1, 0, 0, 0)$ 对Quat实例初始化

具体实现 C++

```
Quat::Quat() :QuatData{ 1,0,0,0 }
{}
```

四元数乘法 `Quat Quat::operator*(const Quat& _Quat) const`

输入参数：待相乘的四元数

返回：该实例四元数与_Quat实例做四元数乘法后的四元数

功能：记该实例对应的四元数为 (q_0, q_1, q_2, q_3) ,_Quat实例对应的四元数为 (p_0, p_1, p_2, p_3) ,返回的结果四元数为 (w, x, y, z) ，计算过程如下[1:附录2]:

$$w = q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3$$

$$x = q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2$$

$$y = q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1$$

$$z = q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0$$

具体实现 C++

```cpp
Quat Quat::operator*(const Quat& _Quat) const
{
  double w = QuatData[0] * _Quat.QuatData[0] - QuatData[1] * _Quat.QuatData[1] - QuatData[2] * _Quat.QuatData[2] - QuatData[3] * _Quat.QuatData[3];
  double x = QuatData[0] * _Quat.QuatData[1] + QuatData[1] * _Quat.QuatData[0] + QuatData[2] * _Quat.QuatData[3] - QuatData[3] * _Quat.QuatData[2];
  double y = QuatData[0] * _Quat.QuatData[2] - QuatData[1] * _Quat.QuatData[3] + QuatData[2] * _Quat.QuatData[0] + QuatData[3] * _Quat.QuatData[1];
  double z = QuatData[0] * _Quat.QuatData[3] + QuatData[1] * _Quat.QuatData[2] - QuatData[2] * _Quat.QuatData[1] + QuatData[3] * _Quat.QuatData[0];
  return Quat(w,x,y,z);
}
```

动力学

轨道类COrbit

成员变量

成员变量 C++

```cpp
struct RV
{
Eigen::Vector3d Pos;//轨道的位置矢量，单位m
Eigen::Vector3d Vel;//轨道的速度矢量，单位m/s
};

struct OrbitElement
{
    double a;                //轨道半长轴(Semi-major Axis)
    double e;                //轨道偏心率(Eccentricity)
    double i;                //轨道倾角（rad） (Inclination)
    double RAAN;             //升交点赤经（rad）(RAAN)
    double omega;            //近地点幅角（rad）(Arg of Perigee)
    double M;                //轨道平近点角（rad）(Mean Anomaly)
    double f;                //真近点角（rad）(True Anomaly)
    double u;                //轨道幅角（纬度幅角）（rad）(Arg of Latitude)
    double E;                //轨道偏近点角（rad）(Eccentric Anomaly)
    double w;                //轨道平均角速度（rad/s）(Palstance  2PI/Period)
    double T;                //轨道周期（s）(Period)
};

class COrbit
{

public:
    RV J2000Inertial;//惯性系RV
    OrbitElement OrbitElements;//轨道根数
};
```

成员函数

二体轨道 `int COrbit::TwoBod(double Ts)`

输入参数：采样时间

返回：轨道参数递推结果

功能：记当前时刻惯性系位置矢量为 $r(t)$,速度矢量为 $v(t)$,_地球引力常数为 μ ， 采样时间为 T_s ，计算过程如下[2:p1]:

$$v(t+T_s)=v(t)+T_s\frac{-\mu}{|r|^3}r(t)$$

$$r(t+T_s)=r(t)+T_sv(t)$$

具体代码 C++

```cpp
int COrbit::TwoBod(double Ts)
{
  if (IsRV(J2000Inertial) == false)
  {
    printf("轨道RV不合法，R:%f(m) V:%f(m/s)\n", J2000Inertial.Pos.norm(), J2000Inertial.Vel.norm());
    return -1;
  }
  else
  {
    double tmp = 1/J2000Inertial.Pos.norm();
    double tmp2 = -EARTH_GRAVITATIONAL * tmp * tmp * tmp;
    J2000Inertial.Vel += Ts * tmp2 * J2000Inertial.Pos;
    J2000Inertial.Pos += Ts * J2000Inertial.Vel;
    return 0;
  }
}
```

姿态类CAttitude

成员变量

成员变量 C++

```cpp
class CAttitude
{
public:
Eigen::Vector3d Omega_b;//本体系角速度，单位rad/s
Quat Qib;//惯性系到本体系四元数
Eigen::Matrix3d SatInaMat;//本体系惯量矩阵，单位kgm2
Eigen::Vector3d WheelMomentum_b;//飞轮组在本体系下的角动量，单位Nms
Eigen::Vector3d TotalTorque;//Tf：干扰力矩：TB 磁力矩：Tw飞轮本体系力矩 TotalTorque=TB+Tf-Tw
};
```

成员函数

姿态动力学 `int CAttitude::AttitudeDynamicsRk4(double Ts)`

输入参数：采样时间

返回：本体系角速度递推结果

功能：记卫星本体系角速度为 ω_b ,卫星转动惯量为 I_{sc} ,反作用轮在本体系产生的角动量为 $h_{w,b}$ ，卫星所受的合外力矩为 τ_s（干扰力矩+磁力矩−飞轮力矩），采样时间为 T_s ，计算过程如下[3:p522]：

$$\dot{\omega}_b = f(\omega_b, I_{sc}, h_{w,b}, \tau_s) = I_{sc}^{-1}[\tau_s - \omega_b \times (I_{sc}\omega_b + h_{w,b})]$$

$$k_1 = f(\omega_b(t), I_{sc}, h_{w,b}(t), \tau_s(t))$$

$$k_2 = f(\omega_b(t) + 0.5T_s k_1, I_{sc}, h_{w,b}(t), \tau_s(t))$$

$$k_3 = f(\omega_b(t) + 0.5T_s k_2, I_{sc}, h_{w,b}(t), \tau_s(t))$$

$$k_4 = f(\omega_b(t) + T_s k_3, I_{sc}, h_{w,b}(t), \tau_s(t))$$

$$\omega_b(t + T_s) = \omega_b(t) + \frac{T_s}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

具体代码 C++

```cpp
Eigen::Vector3d AttDynamics(Eigen::Vector3d Omega_b, Eigen::Matrix3d& SatInaMat, Eigen::Vector3d& Hw, Eigen::Vector3d& Tau_s)
{
    Eigen::Vector3d tmp = Omega_b.cross(SatInaMat * Omega_b + Hw);
    Eigen::Vector3d result = SatInaMat.inverse() * (Tau_s - tmp);
    return result;
}

int CAttitude::AttitudeDynamicsRk4(double Ts)
{
    Eigen::Vector3d k1, k2, k3, k4;
    k1 = AttDynamics(Omega_b, SatInaMat, WheelMomentum_b, TotalTorque);
    k2 = AttDynamics(Omega_b + k1 * (0.5 * Ts), SatInaMat, WheelMomentum_b, TotalTorque);
    k3 = AttDynamics(Omega_b + k2 * (0.5 * Ts), SatInaMat, WheelMomentum_b, TotalTorque);
    k4 = AttDynamics(Omega_b + k3 * Ts, SatInaMat, WheelMomentum_b, TotalTorque);
    Omega_b += (k1 + k2 * 2 + k3 * 2 + k4) * (Ts / 6);
    return 0;
}
```

姿态运动学 `int CAttitude::AttitudeKinematics(double Ts)`

输入参数：采样时间

返回：惯性系到本体系四元数递推结果

功能：记卫星本体系角速度为 ω_b ,惯性系到本体系四元数为 q_{ib} ， 采样时间为 T_s ， 计算过程如下 [4:p21]

$$\theta = T_s\omega_b(t)/2$$

$$q_s = (cos(\theta), sin(\theta)\frac{\omega_b(t)}{|\omega_b(t)|})$$

$$q_{ib}(t+T_s) = q_{ib}(t) \otimes q_s$$

具体代码 C++

```cpp
Quat PlstToDeltaQuat(const Eigen::Vector3d Omega_b, double OfstSec)
{
    double PlstVal = Omega_b.norm();
    return Quat(PlstVal * OfstSec, Omega_b);
}

int CAttitude::AttitudeKinematics(double Ts)
{
    Quat QuatTemp;
    QuatTemp = PlstToDeltaQuat(Omega_b, Ts);
    Qib = Qib * QuatTemp;
    return 0;
}
```

控制算法

飞行环境

基本功能

数据库与显示

参考文献

1. 航天器姿态动力学与控制, 李立涛 荣思远
2. 卫星轨道姿态动力学与控制, 章仁为
3. Wertz J R. Spacecraft attitude determination and control(Book)[J]. Dordrecht, D. Reidel Publishing Co.(Astrophysics and Space Science Library., 1978, 73.
4. 李利亮. 航天器姿态确定系统的故障诊断方法研究[D]. 哈尔滨工业大学, 2020.