

1. 实验名称及目的

八旋翼模型 DLL 生成及 SIL/HIL 实验：在 Matlab 将 Simulink 文件编译生成的八旋翼 DLL 模型文件；并对生成的八旋翼模型进行软硬件在环仿真测试，通过本例程熟悉平台八旋翼模型的使用。

2. 实验原理

OctoX.slx 是构建八旋翼模型的模版。

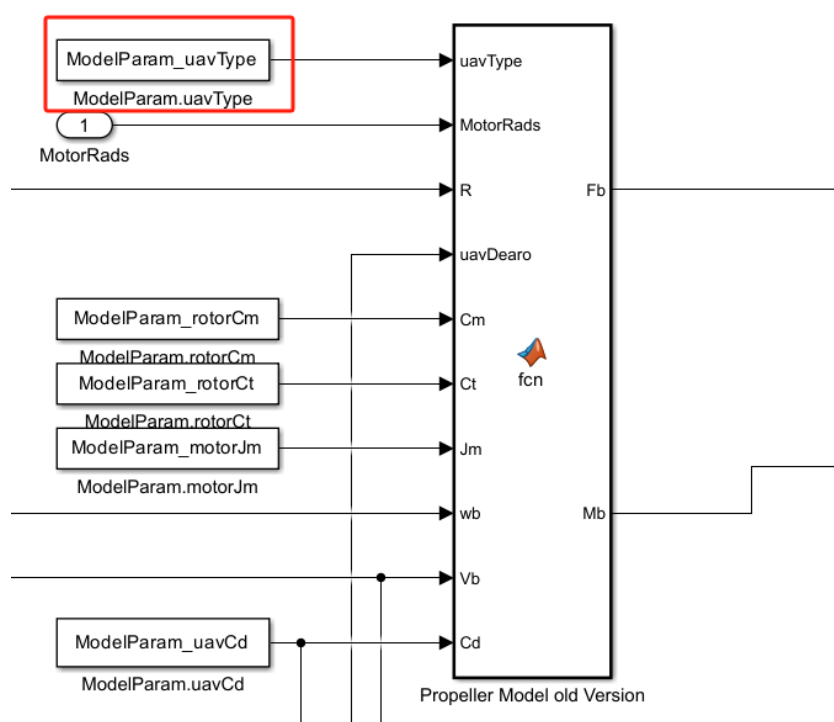
2.1. 模型参数介绍[1][2]

1) 重要参数[1]

OctoX_init.m 中定义了最大系统模型的各种参数，关键数据如下。

飞机的三维显示样式

`ModelParam_uavType = int16(8);` %机型设为八旋翼，这个参数决定了飞机的三维显示样式，需要和 RflySim3D 的 XML 文件中的 `ClassID` 相匹配；同时对于多旋翼飞行器的螺旋桨模型（该模型在力和力矩模块中可见），不同的机型，要对应不同的 `ID` 以计算机架配置和力矩分配



飞机的初始位姿参数

`ModelInit_PosE=[0,0,0];` %用于设置飞机的初始位置，对应了 CopterSim 上的 X 和 Y 初始值。Z 值利用 TerrainZ 实现了从 CopterSim 中读取当前地形高度数据，使得飞机可以初始化在复杂地形的地表面（例如 Grassland 地图）。

`ModelInit_AngEuler=[0,0,0];` %用于设定飞机的初始姿态。飞机姿态角的前两位（俯仰和滚转角）可以通过 `ModelInit_AngEuler` 参数来配置，但是偏航角需要在 CopterSim 中配置。针对导弹等竖直起飞的飞行器，需要设定合适的俯仰和滚转值。

QGC 中显示的地图坐标和高度原点（在 RflySim3D 的 Cesium 大场景中能任意指定飞机在地球三维场景中的坐标）

`ModelParam_GPSLatLong = [40.1540302 116.2593683];` %飞机初始的纬度和精度，单位度。

`ModelParam_envAltitude = -50;` %原点的海拔高度，竖直向下为正，高于海平面填负值，单位米。

执行器的初始参数

`ModelInit_Inputs = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];` 十六维输入向量，定义电机 PWM 初始值，默认全 0，对固定翼和小车需要修改，因为它们的油门在初始状态处于最小值（-1），见“Motor Model”模块

故障接口参数

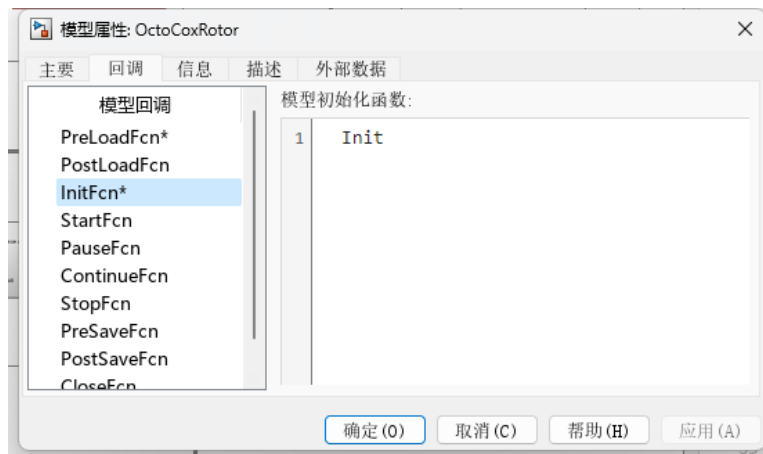
FaultInParams: 可通过外部消息动态改变的 32 维参数向量，在故障注入或者可变形的异构飞行器上有用，也可动态地调整传感器模型噪声等；与 `inSILInts` 和 `inSILFloats` 形成功能互补。

`FaultParamAPI.FaultInParams = zeros(32,1);` 定义了一个名为 `FaultInParams` 的 32 维向量，该向量被初始化为所有元素都为零。

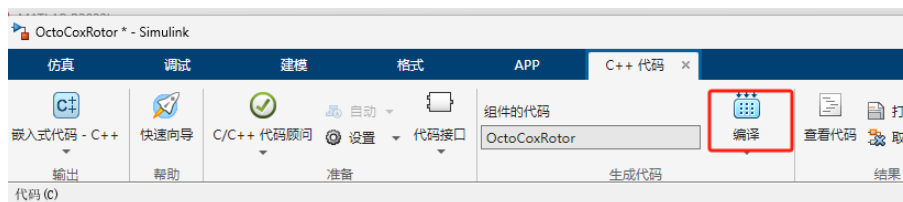
`FaultParamAPI.FaultInParams(3)=1;` 将 `FaultInParams` 向量的第三个元素设置为 1。

2) 参数调用过程[2]

`OctoX.slx` 是八旋翼 DLL 模型生成的模板，模型启动运行（编译）时会调用 `OctoX_init.m`



`OctoX_init.m` 中包含了八旋翼模型参数信息，本脚本会在 `OctoX.slx` 编译时被调用将参数载入 MATLAB 工作空间，也可以直接运行 `OctoX_init.m` 将参数载入工作空间。Simulink 模型会通过参数名称读取工作空间中的参数，故需要保证 simulink 模型中设置的参数名称与 `***_init.m` 中的参数名称相同。



名称	值
FaultParamAPI	1x1 struct
filepath	'F:\d2\4.RflySimModel\2.AdvExps\
HILGPS	1x1 Bus
MavLinkGPS	1x1 Bus
MavLinkSensor	1x1 Bus
MavLinkStateQuat	1x1 Bus
MavVehicleInfo	1x1 Bus
ModelInit_AngEuler	[0,0,0]
ModelInit_Inputs	1x16 double
ModelInit_PosE	[0,0,0]
ModelInit_RateB	[0,0,0]
ModelInit_RPM	0
ModelInit_VelB	[0,0,0]
ModelParam_envAltitude	-50
ModelParam_GPSLatLong	[40.1540,116.2594]
ModelParam_motorCr	718.4300
ModelParam_motorJm	8.0000e-05
ModelParam_motorMinThr	0.0500
ModelParam_motorT	0.0340
ModelParam_motorWb	108.7200
ModelParam_rotorCm	1.4170e-07
ModelParam_rotorCt	8.3800e-06
ModelParam_uavCCm	[0.0035,0.0039,0.0034]
ModelParam_uavCd	0.4720
ModelParam_uavDearo	0.1200
ModelParam_uavJ	[0.0609,0,0,0,0.0609,0,0,0,0.0117]
ModelParam_uavMass	1.5000
ModelParam_uavMotNumbs	8
ModelParam_uavR	0.3250
ModelParam_uavType	8

1x1 int16

GenerateModelDLLFile.p 是将 slx 模型转化为 DLL 模型文件的脚本，使用 RflySim 平台进行载具软硬件在环仿真时，需要将 DLL(windows 下)/SO(Linux 下)模型导入到 CopterSim，形成运动仿真模型，因此，在 Simulink 模型编译完成后，需要将模型对应的 C++文件打包成 DLL/SO 模型。

2.2. 输入信号[4]

1) 电机数据 inPwms

输入接口 inPWMs，16 维执行器控制量输入，已归一化到-1 到 1 尺度(通常电机是 0-1，舵机是 -1~1)，它的数据来自飞控回传的电机控制 MAVLink 消息 mavlink_hil_actuator_controls_t 的 controls，具体定义如下：

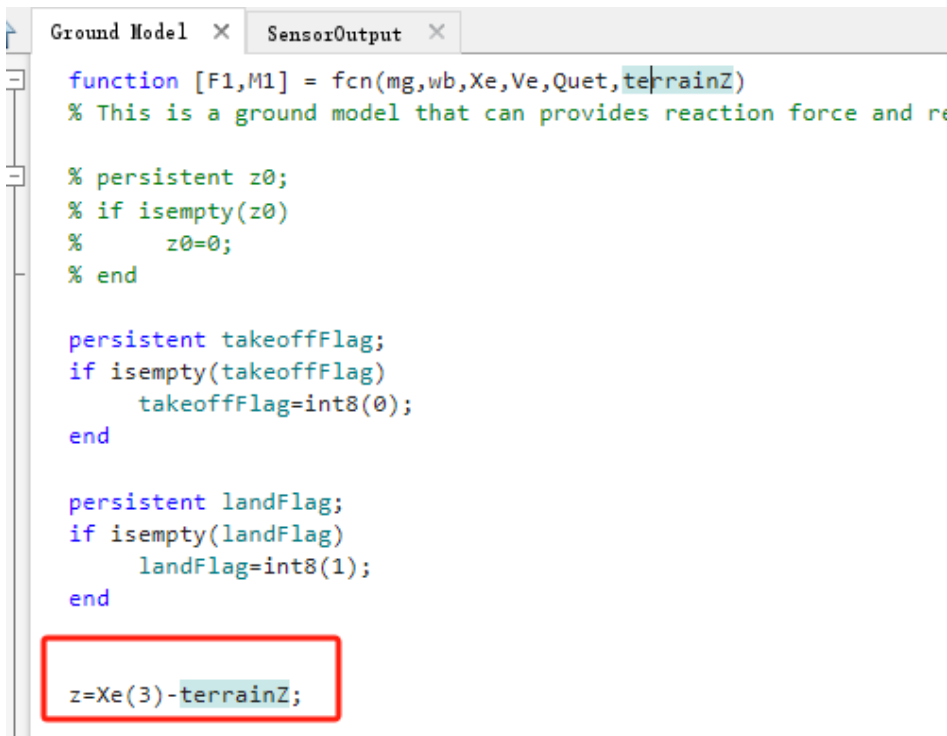
```
typedef struct __mavlink_hil_actuator_controls_t {
    uint64_t time_usec; //时间戳，从开机后的时间，单位 ms
    uint64_t flags; //标志位，用于显示当前的飞行状态
    float controls[16]; //控制量，16 维电机的控制量，发送到模型中，驱动飞机飞行
    uint8_t mode; //模型，用于显示飞机当前的飞行模式和是否上锁等信息 }
mavlink_hil_actuator_controls_t;
```

软件在环仿真时，电机控制指令从 PX4 SITL 控制器通过 TCP4561 系列端口以 MAVLink 协议发送到运动仿真模型的 inPWMs 接口，而硬件在环仿真时，该指令是从飞控通过串口以 MAVLink 协议发送到运动仿真模型的 inPWMs 接口。

2) 地形高度 terrainZ

最大模板可以利用 TerrainZ 实现从 CopterSim 中读取当前地形高度数据，使得飞机初始化在复杂地形的地表面（例如 RflySim3D 中的 Grassland 地图）。这个值是由 CopterSim 读取 DLL 模型初始位置参数 ModelInit_PosE 中的 xy 坐标，根据地形校准文件及高程信息解算出地形高度 TerrainZ，通过 Mavlink 消息传输给 DLL 模型的 TerrainZ 接口，在 DLL 模型

中通过 PhysicalCollisionModel/ GroundSupportModel/ Ground Model 函数中重新定义模型初始位置的高度，最后会通过 MavVehile3DInfo 接口传给 RflySim3D 中的三维显示模型。



```
function [F1,M1] = fcn(mg,wb,Xe,Ve,Quet,terrainZ)
% This is a ground model that can provides reaction force and r

% persistent z0;
% if isempty(z0)
%     z0=0;
% end

persistent takeoffFlag;
if isempty(takeoffFlag)
    takeoffFlag=int8(0);
end

persistent landFlag;
if isempty(landFlag)
    landFlag=int8(1);
end

z=Xe(3)-terrainZ;
```

3) 碰撞数据 inFloatsCollision

利用 inFloatsCollision 实现了一个简单地物理引擎，可以根据 RflySim3D 回传的四周距离数据，实现碰到障碍物的回弹、碰到其他飞机便坠毁等功能

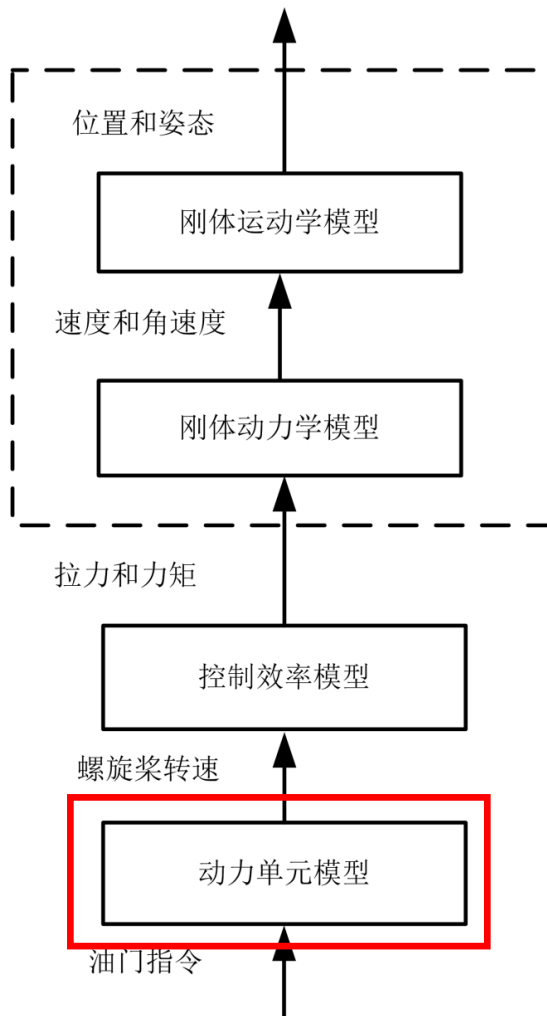
2.3. 模型模块[3]

表格 1 模型参数对照表

参数名称	公式中参数名称	.m 文件参数名称	参数值
总质量	m	ModelParam.uavMass	1.515(kg)
重力加速度	g	ModelParam.envGravityAcc	9.8(m/s ²)
转动惯量矩阵	J	ModelParam.uavJ	$\begin{bmatrix} 0.06087 & 0 & 0 \\ 0 & 0.06087 & 0 \\ 0 & 0 & 0.1166 \end{bmatrix}$

多旋翼机身半径	$\frac{d}{2}$	ModelParam.uavR	0.325(m)
螺旋桨拉力系数	c_T	ModelParam.rotorCt	8.380e-06 (N/(rad/s) ²)
螺旋桨力矩系数	c_M	ModelParam.rotorCm	1.417e-07(N.m/(rad/s) ³)
油门到电机稳态转速 曲线斜率	C_R	ModelParam.motorCr	718.43
油门到电机稳态转速 曲线零点	ω_b	ModelParam.motorWb	108.72(rad/s)
电机螺旋桨转动惯量	J_{RP}	ModelParam.motorJm	0.00008(kg/m ²)
电机响应时间常数	T_m	ModelParam.motorT	0.034(s)
阻力系数	C_d	ModelParam.uavCd	0.472(N/(m/s) ²)
阻尼力矩系数	C_{dm}	ModelParam.uavCCm	[0.0035 0.0039 0.0064] (N/(rad/s) ²)

1) Motor Model 电机模块



在该模块中输入为 PWM 值（通过 inPWMs 接口获取），经过各电机的非线性动力学模型后得到各电机转速，该模块的输出分别为输入给力和力矩模型的电机转速（弧度每秒）；输入给 UE 的电机转速（转每分）

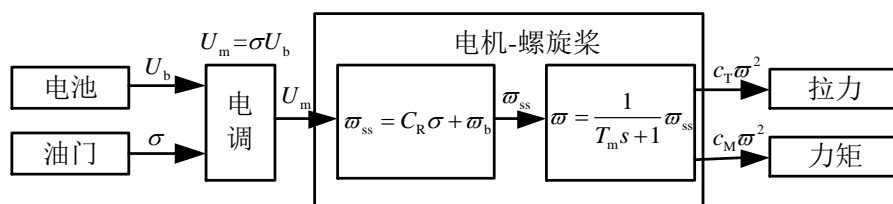
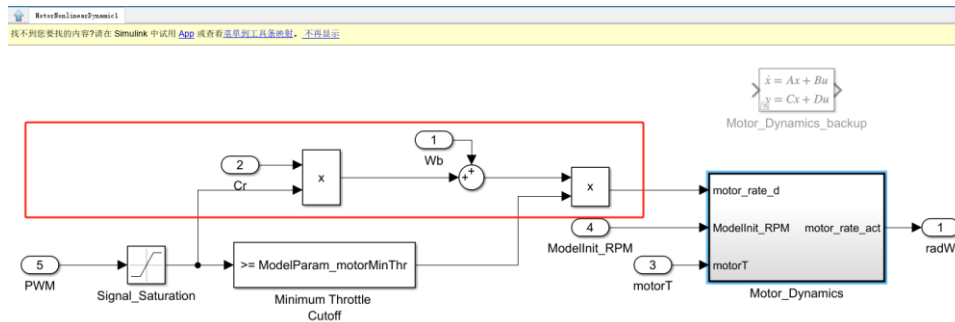


图 1 油门到螺旋桨转速

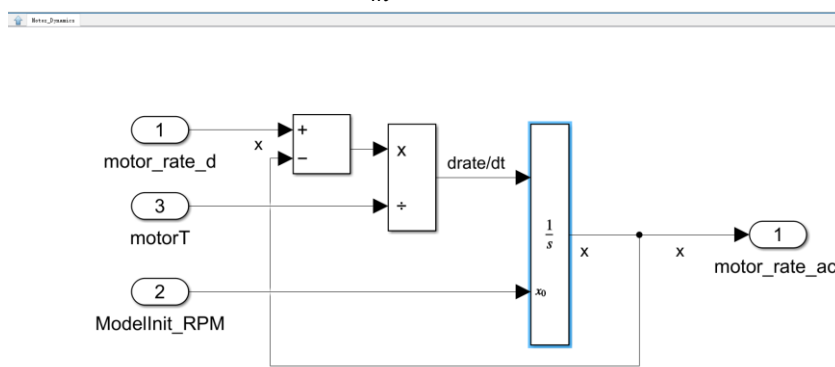
电调接收油门指令 σ 和电池输出电压 U_b 后产生等效平均电压为 $U_m = \sigma U_b$ 。首先输入一个电压信号，电机转动到一个稳态转速 ω_{ss} 。这种关系通常是线性的，令 $C_R = C_b U_b$ 得

$$\omega_{ss} = C_b U_m + \omega_b = C_R \sigma + \omega_b$$



其次，当给定一个油门指令，电机到达稳态转速 ω_{ss} 需要一段时间，该时间决定了电机的动态响应，记为 T_m 。在通常情况下，无刷直流电机的动态过程可以简化为一阶低通滤波器，其传递函数为

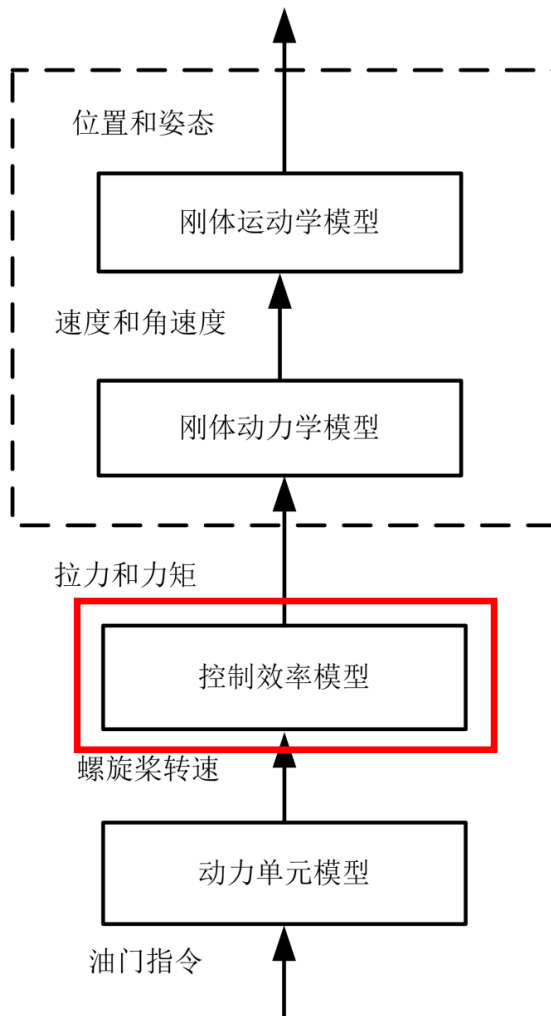
$$\omega = \frac{1}{T_m s + 1} \omega_{ss}$$



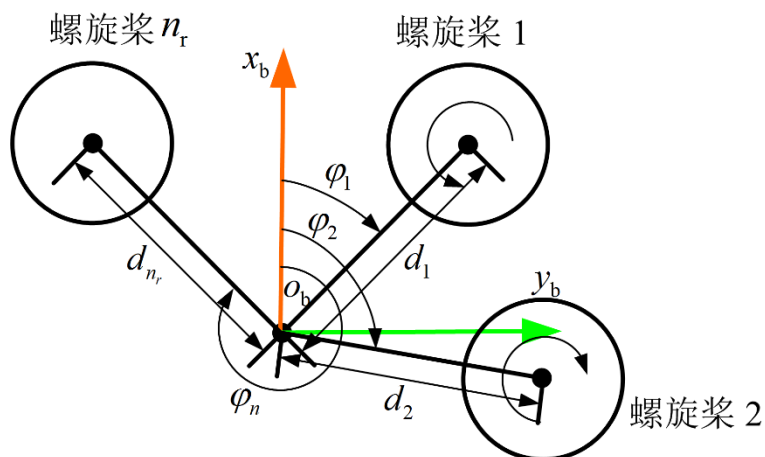
完整的动力单元模型为

$$\omega = \frac{1}{T_m s + 1} (C_R \sigma + \omega_b)$$

2) Force and Moment Model 力和力矩模块

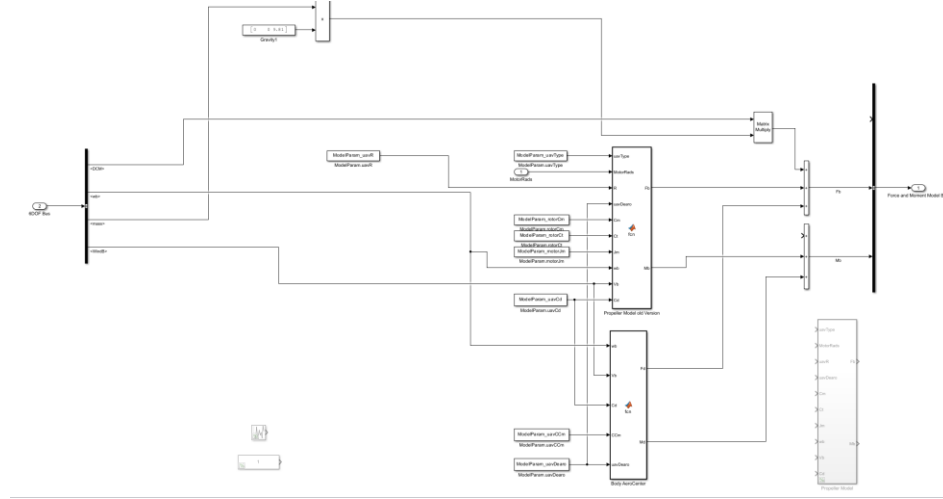


该模块输入为电机转速 MotorRads 、飞机运动学姿态 6DOF 和地形高度输入 TerrainZ ，输出为多旋翼合力、合力矩 $\text{Force and Moment Model Bus}$ 。对于桨数 $n_r \geq 5$ 的任意多旋翼，其拉力和力矩模型如下

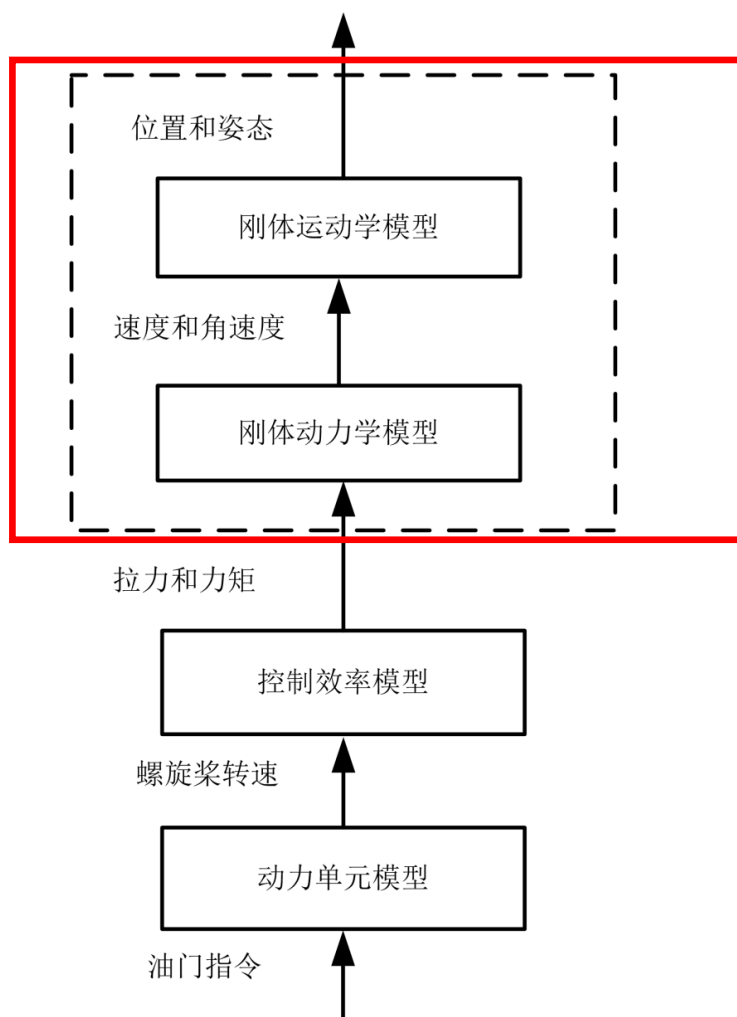


$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} c_T & c_T & \cdots & c_T \\ -d_1 c_T \sin \varphi_1 & -d_2 c_T \sin \varphi_2 & \cdots & -d_{n_r} c_T \sin \varphi_{n_r} \\ d_1 c_T \cos \varphi_1 & d_2 c_T \cos \varphi_2 & \cdots & d_{n_r} c_T \cos \varphi_{n_r} \\ c_M \delta_1 & c_M \delta_2 & \cdots & c_M \delta_{n_r} \end{bmatrix}}_{\mathbf{M}_{n_r}} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \vdots \\ \varpi_{n_r}^2 \end{bmatrix}$$

其中 f 为总拉力， τ_x 、 τ_y 、 τ_z 分别为 x、y、z 方向上的合力矩。



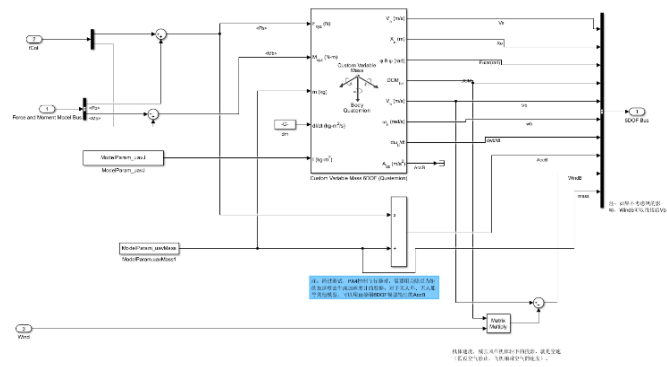
3) 6DOF 六自由度刚体模块



用于描述无人机在空中运动时的姿态和位置变化。考虑了无人机在三个坐标轴上的旋转运动（俯仰、横滚和偏航）以及机体与地球坐标系上的平移运动（前后、左右和上下）。

还可以根据实际需求对模型进行扩展，考虑更多的因素，如飞行器的非线性特性、气动力和惯性矩等。基于四元数模型如下

$$\left\{ \begin{array}{l} {}^e \dot{\mathbf{p}} = {}^e \mathbf{v} = \mathbf{R} \cdot {}^b \mathbf{v} \\ {}^b \dot{\mathbf{v}} = -[{}^b \boldsymbol{\omega}]_{\times} \cdot {}^b \mathbf{v} + {}^b \mathbf{F}/m \\ \dot{q}_0 = -\frac{1}{2} \mathbf{q}_v^T \cdot {}^b \boldsymbol{\omega} \\ \dot{\mathbf{q}}_v = \frac{1}{2} (q_0 \mathbf{I}_3 + [\mathbf{q}_v]_{\times}) {}^b \boldsymbol{\omega} \\ \mathbf{J} \cdot {}^b \dot{\boldsymbol{\omega}} = -{}^b \boldsymbol{\omega} \times (\mathbf{J} \cdot {}^b \boldsymbol{\omega}) + {}^b \mathbf{M} \end{array} \right.$$

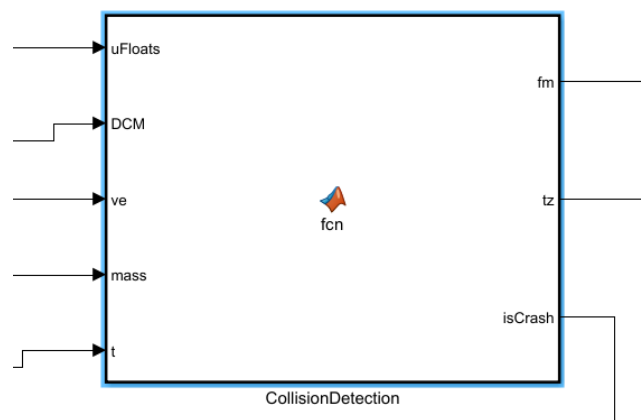


4) PhysicalCollisionModel 物理碰撞模块

碰撞模块能检测多旋翼飞行过程中是否碰撞到了物体,以及所碰撞物体的类型并做出符合物理规律的反应。开启碰撞模式后,在碰撞到飞机时多旋翼的速度满足冲量定理,在碰撞到房屋等固定物体时,多旋翼会朝着障碍物反向的反弹回去,反弹速度为原速度的 1/10。

由图可知,碰撞模块的输入为 $uFloats$ 、 DCM 、 ve 、 $mass$ 、 t ,输出为 fm 、 tz 、 $isCrash$ 。

其中 $uFloats$ 为 20 维外部输入浮点信号,该端口为碰撞模型预留,可以通过 UDP 网络从 UE4 传输。 DCM 为方向余弦矩阵, ve 为碰撞时的速度, $mass$ 为多旋翼质量, t 为时间戳。输出 fm 为力和力矩直接作用到 6DOF 对机体运动产生影响, tz 则表示多旋翼离地面的高度和 XYZ 的坐标, $isCrash$ 则为碰撞判断,如果碰撞发生则三个电机损坏。具体碰撞逻辑可以点击进入该模块详细了解。



5) SensorOutput 传感器输出模块

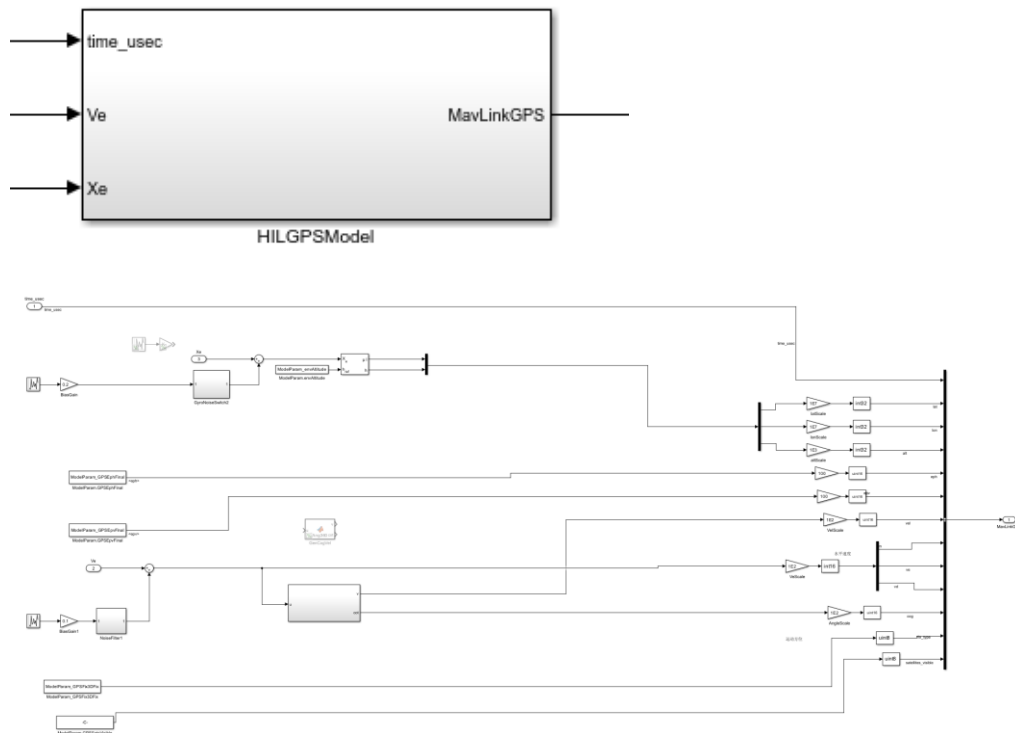
该模块中包括了环境模型、传感器模型和 GPS 模型

环境模型

环境模型对重力和大气压强对无人系统飞行产生的影响进行了模拟

GPS 模型

GPS 模型用于计算 GPS 数据，在仿真时反馈回 PX4 控制器



2.4. 输出信号[4]

`***_init.m` 会调用 `MavLinkStruct.mat` 导入四个输出结构体 bus（`MavLinkGPS`、`MavLinkSensor`、`MavLinkStateQuat` 以及 `MavVehileInfo`）的定义到工作空间。

```
load MavLinkStruct;
```

最大模型模版包含了 6 个输出信号，分别是 `MavHILSensor`、`MavHILGPS`、`MavVehile3Dinfo`、`outCopterData`、`ExtToUE4`、`ExtToPX4`。

1) MavHILSensor（传感器接口集合）

模型发送给 `RflySim3D` 的真实仿真数据，是平滑的理想值，这些数据可用于 `Simulink` 下的飞控与模型进行软件仿真测试。对应了 `MAVLink` 的 `mavlink_hil_sensor_t` 消息，本结构体包含了，加速度传感器的加速度值、陀螺仪传感器的角速度值、磁罗盘传感器的磁场值，气压和空速传感器的气压值等。这些传感器的值在仿真时由我们的模型提供，在真机飞行时由真实传感器芯片提供。

```
typedef struct __mavlink_hil_sensor_t {  
    uint64_t time_usec; /*时间戳，单位毫秒 ms*/  
    float xacc; /*机体坐标系 x 方向加速度，单位 m/s^2 */  
    float yacc; /*机体坐标系 y 方向加速度，单位 m/s^2 */  
    float zacc; /*机体坐标系 z 方向加速度，单位 m/s^2 */  
    float xgyro; /*机体坐标系 x 方向角加速度，单位 rad/s */  
    float ygyro; /*机体坐标系 y 方向角加速度，单位 rad/s */  
}
```

```

float zgyro; /*机体坐标系 z 方向角加速度, 单位 rad/s */
float xmag; /*机体坐标系 x 方向磁通量, 单位 Gauss =T/10000*/
float ymag; /*机体坐标系 y 方向磁通量, 单位 Gauss =T/10000*/
float zmag; /*机体坐标系 z 方向磁通量, 单位 Gauss =T/10000*/
float abs_pressure; /*绝对气压值, 单位 millibar=100Pa*/
float diff_pressure; /*相对气压值, 单位 millibar=100Pa*/
float pressure_alt; /*气压解算高度值, 单位 m*/
float temperature; /*温度, 单位摄氏度*/
uint32_t fields_updated; /*传感器参数初始化标志位, bit 0 = xacc, bit 12: temperature, bit 31:
全部重新初始化 */
} mavlink_hil_sensor_t;

```

2) MavHILGPS (GPS 接口)

模型发送给飞控的 GPS 数据值, 它对应了 MAVLink 消息的 `mavlink_hil_gps_t` 结构体。输出信号中包含了经纬高、水平竖直精度、地速、北东地的速度、偏航角、定位状态和卫星数量等数据。这些传感器的值在仿真时由我们的模型提供, 在真机飞行时由真实 GPS 模块提供。

```

typedef struct __mavlink_hil_gps_t {
    uint64_t time_usec; /*时间戳, 单位毫秒 ms*/
    int32_t lat; /*纬度(WGS84 地球模型), 单位度, 再乘以 1E7*/
    int32_t lon; /*经度(WGS84 地球模型), 单位度, 再乘以 1E7*/
    int32_t alt; /*高度 (AMSL 地球模型, 而不是 WGS84), 单位 m, 再乘以 1000 (向上为正)*/
    uint16_t eph; /*GPS 水平方向定位精度, 单位 cm, 如果不知道设为 65535*/
    uint16_t epv; /*GPS 垂直方向定位精度, 单位 cm, 如果不知道设为 65535*/
    uint16_t vel; /*GPS 地速, 单位 cm/s, 如果不知道设为 65535*/
    int16_t vn; /*GPS 地速朝北方向分量, 单位 cm/s */
    int16_t ve; /*GPS 地速朝东方向分量, 单位 cm/s */
    int16_t vd; /*GPS 地速朝下方向分量, 单位 cm/s */
    uint16_t cog; /*运动方向, 单位和范围 0~359.99 度, 再乘以 100 degrees * 100, 如果不知道设为 65535*/
    uint8_t fix_type; /*定位类型 0-1: no fix, 2: 2D fix, 3: 3D fix. */
    uint8_t satellites_visible; /*可见卫星数, 如果不知道设为 255*/
} mavlink_hil_gps_t;

```

注: GPS 数据的发送频率与真实传感器硬件基本相同为 10Hz, 因此飞控的实时位置并不能靠 GPS 直接提供, 需要与 IMU 等传感器进行融合滤波估计得到。

3) MavVehicle3Dinfo (真实仿真数据输出)

模型发送给飞控的各种传感器数据的集合, 对应了 MAVLink 的 `mavlink_hil_sensor_t` 消息。输出信号中包括了加速度传感器的加速度值、陀螺仪传感器的角速度值、磁罗盘传感器的磁场值, 气压和空速传感器的气压值等。

```

struct SOut2Simulator {
    int copterID; //飞机 ID, 用于区分局域网内不同飞机
    int vehicleType; //飞机样式, 区分同种飞机 (如四旋翼) 下的不同样式 (例如, 大疆、AR.Drone)
    double runnedTime; //时间戳, 当前时刻的时间, 单位毫秒
    float VelE[3]; //速度向量, 地球坐标系的 xyz 速度 (z 向下为正), 单位 m/s
    float PosE[3]; //位置向量, 地球坐标系下的 xyz 方向 (z 向下为正, 单位 m, 以起飞点为坐标原点
    float AngEuler[3]; //姿态角, 飞机的欧拉角, 定义于机体坐标系, 单位弧度
    float AngQuatern[4]; //四元数, 飞机姿态的四元数, 定义于机体坐标系
    float MotorRPMS[8]; //电机转速, 飞机的各个旋翼转速, 单位转每分
    float AccB[3]; //加速度, 飞机的运动加速度, 单位 m/s^2
    float RateB[3]; //角速度, 飞机的转动角速度, 单位 rad/s

```

```
double PosGPS[3]; //GPS 坐标，飞机的经纬高坐标，单位度、度、米
};
```

3. 实验效果

实现八旋翼飞机 DLL 模型文件生成，以及完成八旋翼软硬件在环仿真。

4. 文件目录

文件夹/文件名称	说明
OctoX.slx	八旋翼飞机模型模板文件。
OctoX.dll	八旋翼飞机 DLL 模型
OctoXHITLRun.bat	硬件在环仿真批处理文件。
OctoXSITLRun.bat	软件在环仿真批处理文件。
GenerateModelDLLFile.p	DLL 格式转化文件。
Init.m	动力学模型相关参数。
Init_control.m	控制器初始化参数。
MavLinkStruct.mat	MavLink 数据结构体 mat 文件

5. 运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑 ^①	1
2	RflySim 平台免费版	PX4 飞控 ^②	1
3	MATLAB 2017B 及以上 ^③	数据线	1

① 推荐配置请见：<https://doc.rflysim.com>

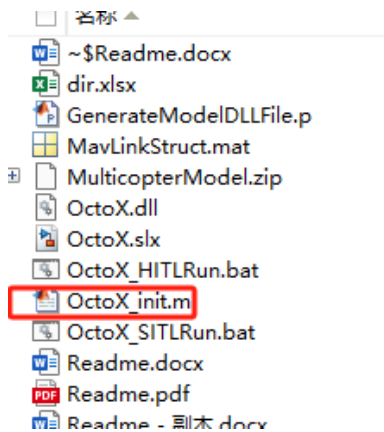
② 须保证平台安装时的编译命令为：px4_fmu-v6c_default，固件版本为：1.13.3。其他配套飞控请见：<http://doc.rflysim.com/hardware.html>。

6. 实验步骤

6.1. DLL 模型生成

Step 1:

打开“Init.m”文件并运行



```

42
43 %Initial condition
44
45
46 %% 6DOF模块相关参数
47 % 飞机质量:
48 ModelParam_uavMass=1.5;
49 % 转动惯量
50 ModelParam_uavJ= [0.06087,0,0;0,0.06087,0;0,0,0.01166];
51 ModelInit_VelB=[0,0,0];
52 ModelInit_RateB=[0,0,0];
53
54
55 %% 电机模型参数
56 ModelParam_uavMotNumbs = int8(8);
57 %ModelParam_ControlMode = int8(1); %整型 1表示Auto模式, 0表示Manual模式
58 ModelParam_motorMinThr=0.05;
59 ModelParam_motorCr=718.43;
60 ModelParam_motorWb=108.72;
61 ModelParam_motorT= 0.034;%0.0261;
62 ModelParam_motorJm =0.00008;
63 ModelParam_rotorCm=1.417e-07;
64 ModelParam_rotorCt=8.380e-06;
65 ModelInit_RPM = 0; %Initial motor speed (rad/s)
66 ModelInit_Inputs = [0 0 0 0 0 0 0 0 0 0 0 0 0];
67
68
69
70 %% 力和力矩模型参数
71 ModelParam_uavR=0.325;
72 %ModelParam_uavCtrlEn = int8(0);

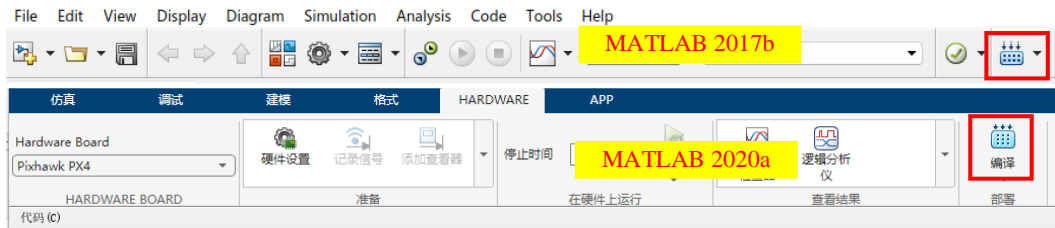
```

注意事项：从.m文件可以看出，与四旋翼模型相比，八旋翼模型主要不同点如下：

- 1) 电机数量：八旋翼无人机的电机数量为 8（`ModelParam_uavMotNumbs = int8(8)`），而四旋翼无人机的电机数量为 4（`ModelParam_uavMotNumbs = int8(4)`）。与四旋翼相比，八旋翼增加 4 个控制通道，多控制了 4 个电机来驱动旋翼，这可以提高其悬停稳定性和操控性。
- 2) 转动惯量：八旋翼无人机的在 x、y、z 方向的转动惯量(`ModelParam_uavJ`)分别为 0.06087、0.06087、0.01166，无量纲。
- 3) 电机时间常数：八旋翼无人机的电机时间常数为 0.034（`ModelParam_motorT= 0.034`），单位 s。

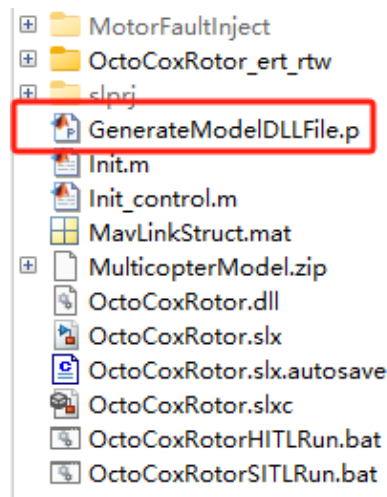
Step 2:

打开“OctoX.slx”Simulink 文件，点击 Build Model 按钮生成代码。



Step 3:

代码生成完毕后，在 Matlab 中右键“GenerateModelDLLFile.p”文件，点击运行，生成 OctoX.dll 文件。

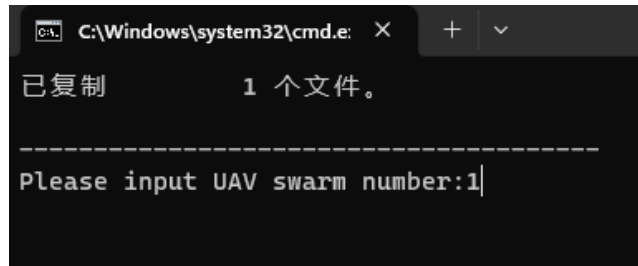


6.2. 软件在环仿真

Step 1:

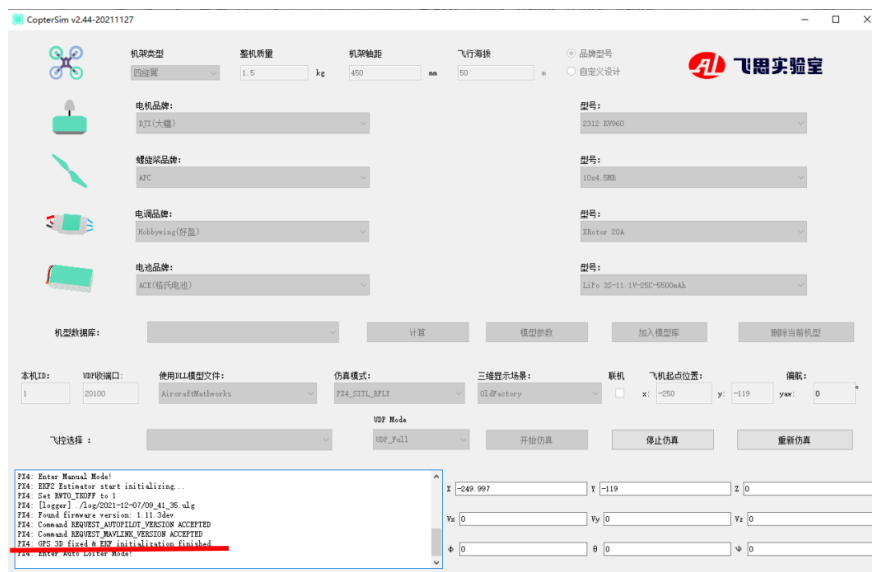
以管理员身份运行“OctoXSITLRun.bat”批处理文件，在弹出的终端窗口中输入 1，启动一架飞机的软件在环仿真。

GenerateModelDLLFile.p	2023/11/28 16:50	MATLAB P-code	6 KB
MavLinkStruct.mat	2023/11/28 16:50	Access.Shortcut....	5 KB
MulticopterModel.zip	2023/12/27 14:23	Bandizip.zip	104 KB
OctoX.dll	2023/12/26 19:03	应用程序扩展	232 KB
OctoX.slx	2023/12/27 14:23	Simulink Model	79 KB
OctoX_HITLRun.bat	2023/12/26 18:56	Windows 批处理...	6 KB
OctoX_init.m	2023/12/28 10:26	MATLAB Code	3 KB
OctoX_SITLRun.bat	2023/12/26 18:56	Windows 批处理...	6 KB
Readme - 副本.docx	2023/12/28 9:46	Microsoft Word ...	3,693 KB
Readme.docx	2023/12/28 10:52	Microsoft Word ...	3,365 KB
Readme.pdf	2023/12/26 19:00	Microsoft Edge ...	2,161 KB



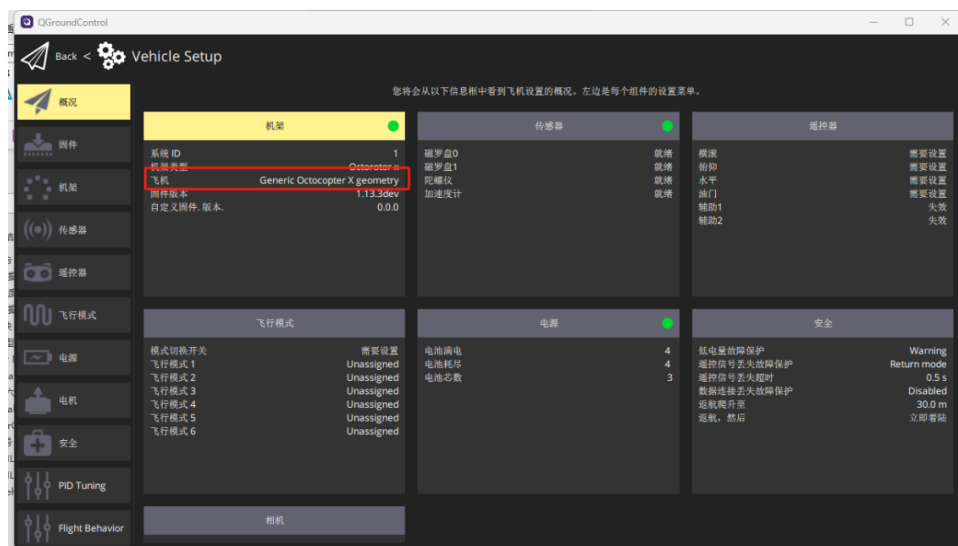
Step 2:

等待 CopterSim 中显示连接上 UE4。



Step 3:

确保 QGC 中机架设置如图



在 QGC 中点击起飞按钮，并设置一定的起飞高度，之后滑动确认。



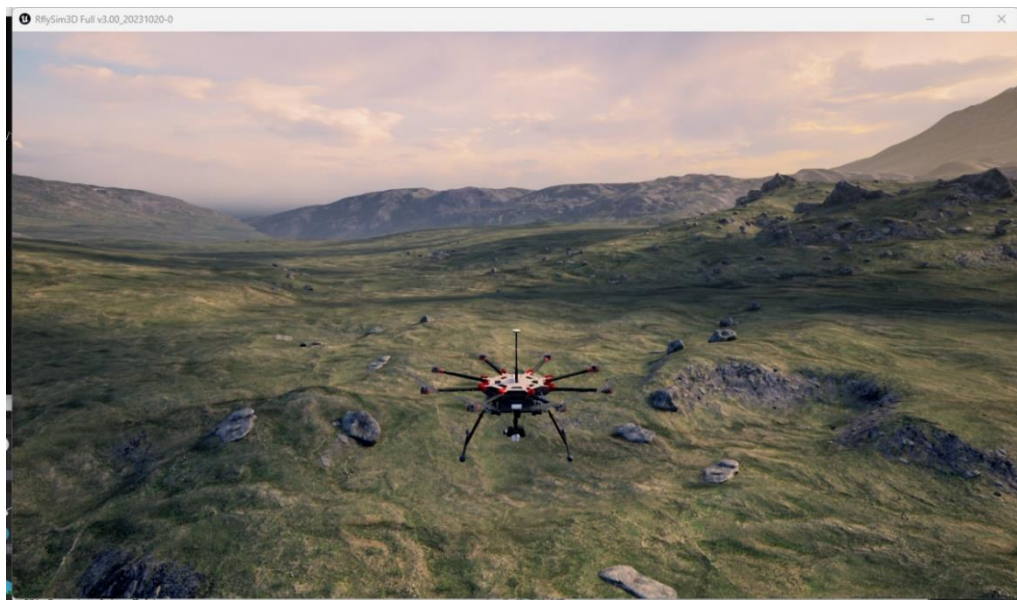
注意事项：从 OctoXSITLRun.bat 文件中可以找到定义机架类型的语句

```
set PX4SITLFrame=octo_x
```

八旋翼模型的机架配置在
 \PX4PSP\Firmware\ROMFS\px4fmu_common\init.d\airframes\octo_x 中定义

Step 4:

在 RflySim3D 中观察是否正常飞行。



6.3. 硬件在环仿真












Step 1:

按下图所示将飞控与计算机连接。

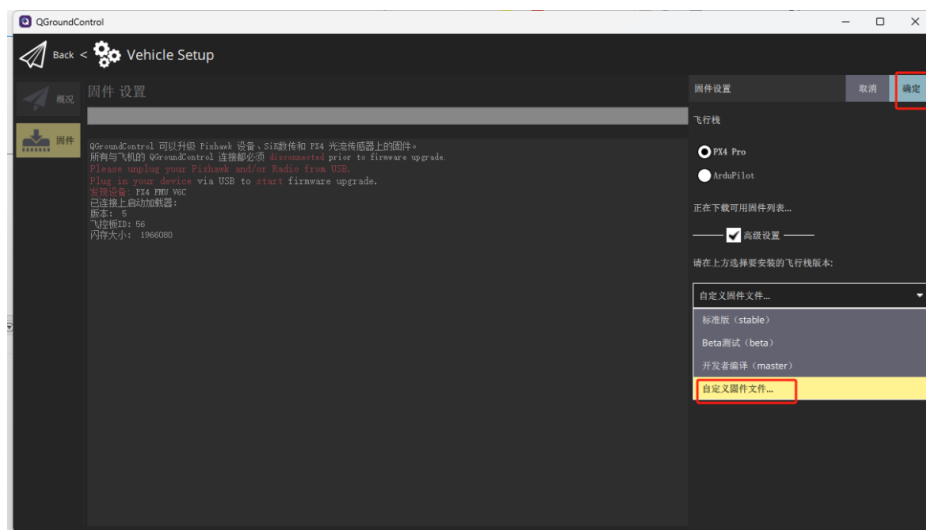


Step 2:

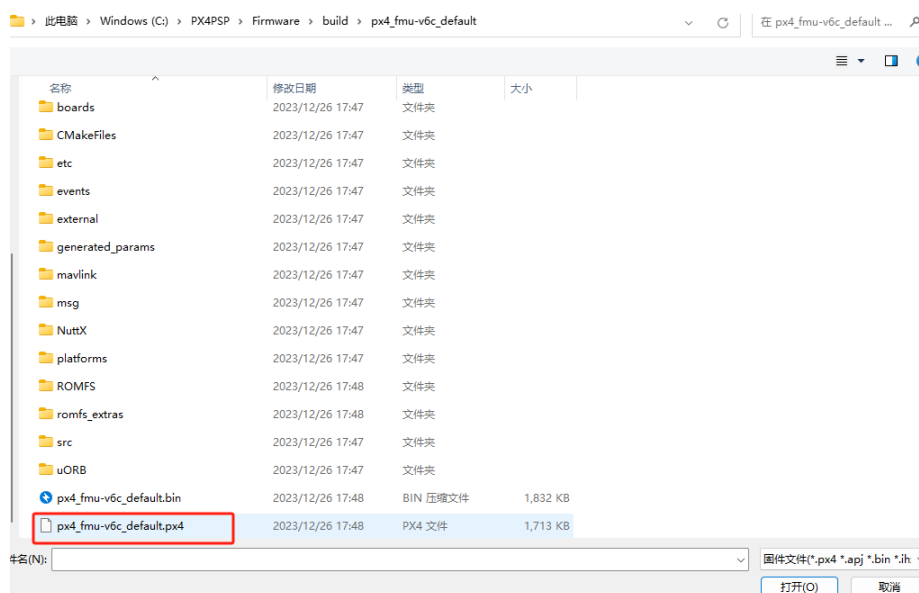
在 Rflytools 文件夹中打开 QGC 地面站。

	3DDisplay	2023/7/27 15:02	快捷方式	1 KB
	CopterSim	2023/7/27 15:02	快捷方式	1 KB
	FlightGear-F450	2023/7/27 15:02	快捷方式	2 KB
	HITLRun	2023/7/27 15:02	快捷方式	2 KB
	Python38Env	2023/7/27 15:02	快捷方式	2 KB
	QGroundControl	2023/7/27 15:02	快捷方式	1 KB
	RflySim3D	2023/7/27 15:02	快捷方式	1 KB
	RflySimAPIs	2023/7/27 15:02	快捷方式	1 KB
	RflySimUE5	2023/7/27 15:02	快捷方式	1 KB
	SITLRun	2023/7/27 15:02	快捷方式	2 KB
	Win10WSL	2023/7/27 15:02	快捷方式	2 KB

点击进入左侧“固件”界面后，勾选下方“高级设置”选择自定义固件文件(选用 v6c 飞控，固件版本为 1.13.3)。

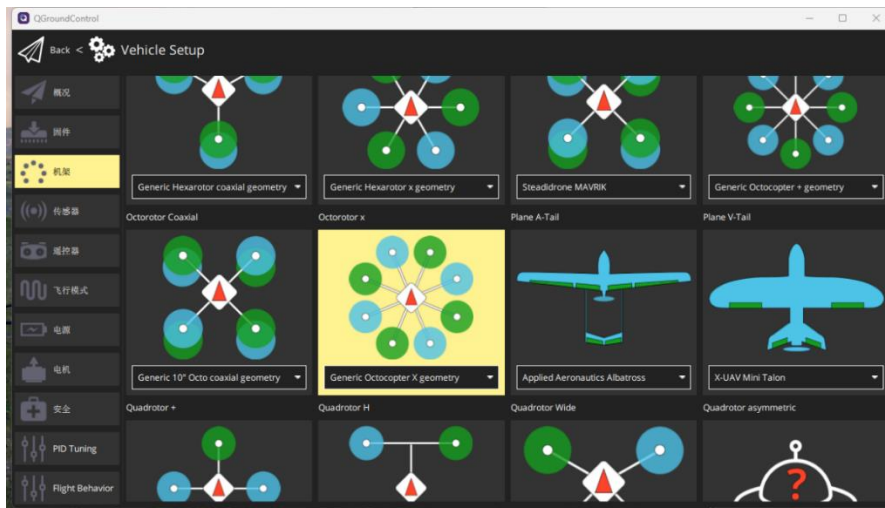


在 `C:\PX4PSP\Firmware\build\px4_fmu-v6c_default` 这个路径下选择确认 `px4_fmu-v6c_default.px4` 文件。



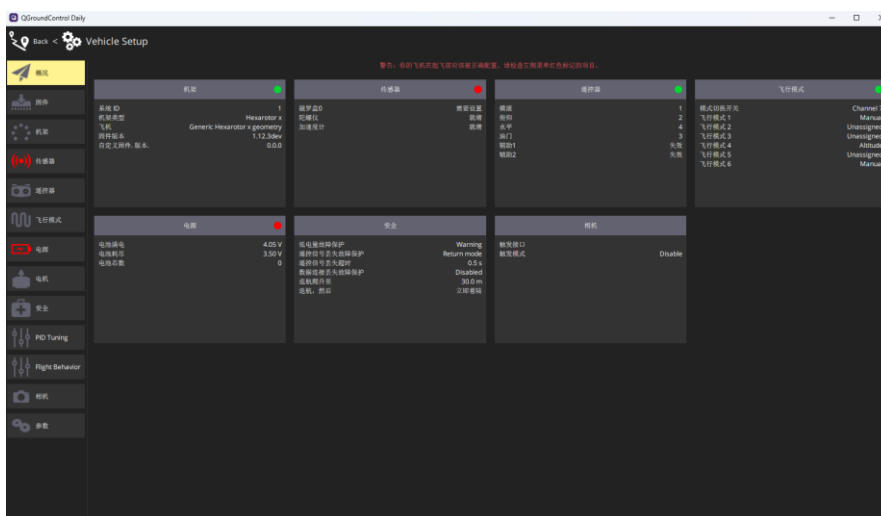
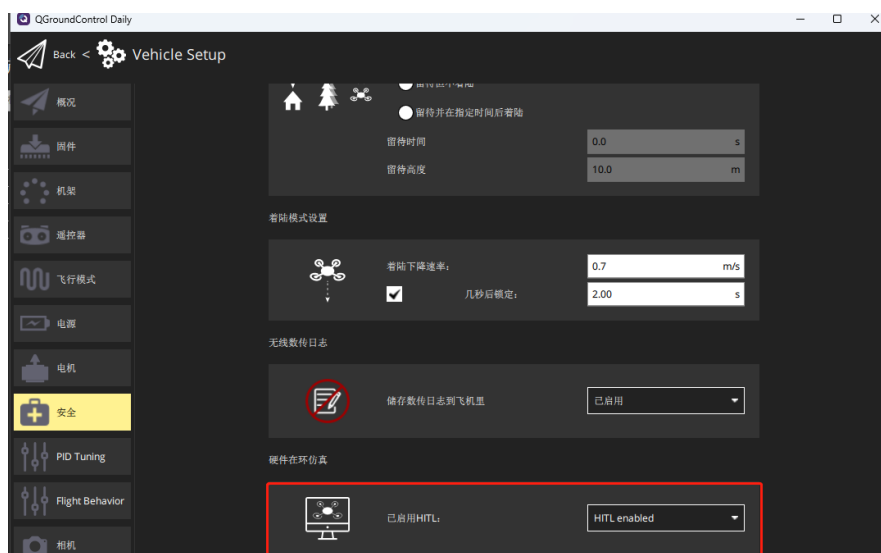
Step 4:

在机架界面设置机架型号为“Generic Octocopter X geometry”，设置完毕后点击右侧“应用并重启”。













Step 5:

在“安全”界面，选择“HITL enabled”启动硬件在环仿真，之后在概况界面中确认配置完成后，重新插拔飞控完成设置。



Step 6:

右键以管理员身份运行“OctoXHITLRun.bat”批处理文件，在弹出的终端窗口中根据串口提示输入串口号 5，启动一架飞机的硬件在环仿真。

	MavLinkStruct.mat	2023/11/28 16:50	Access.Shortcut...	5 KB
	MulticopterModel.zip	2023/12/27 14:23	Bandizip.zip	104 KB
	OctoX.dll	2023/12/26 19:03	应用程序扩展	232 KB
	OctoX.slx	2023/12/27 14:23	Simulink Model	79 KB
	OctoX_HITLRun.bat	2023/12/26 18:56	Windows 批处理...	6 KB
	OctoX_init.m	2023/12/28 10:26	MATLAB Code	3 KB
	OctoX_SITLRun.bat	2023/12/26 18:56	Windows 批处理...	6 KB
	Readme - 副本.docx	2023/12/28 9:46	Microsoft Word ...	3,693 KB
	Readme.docx	2023/12/28 10:52	Microsoft Word ...	3,365 KB
	Readme.pdf	2023/12/26 19:00	Microsoft Edge ...	2,161 KB

```
C:\Windows\system32\cmd.e  X  +  v
已复制      1 个文件。

-----
Please input the Pixhawk COM port list for HIL
Use ',' as the separator if more than one Pixhawk
E.g., input 3 for COM3 of Pixhawk on the computer
Input 3,6,7 for COM3, COM6 and COM7 of Pixhawks

Available COM ports on this computer are:
COM3: ??????????
COM4: ??????????
COM5: USB ????
Recommended COM list input is: 3,4,5

-----
My COM list for HITL simulation is:5
```

Step 7:

之后测试步骤与软件在环的 Step3 到 Step4 相同，运行之后在 RflySim3D 中观察是否按指令飞行。

7. 参考资料

- [1]. [API.pdf 中 DLL/SO 模型与通信接口的重要参数部分](#)。
- [2]. [API.pdf 中的环境配置](#)
- [3]. [API.pdf 中的 Simulink 建模模板介绍](#)
- [4]. [API.pdf 中 DLL/SO 模型与通信接口的数据协议部分](#)

8. 常见问题

Q1: ****

A1: ****