平台固定翼模型控制仿真实验原理

1. 文件目录.....1
2. 总体说明.....1
3. 关键子模块实现原理.....1
3.1. 固定翼碰撞检测的实现.....1
4. 相关文献.....6
附加资源.....6

1. 文件目录

例程目录：[安装目录]\RflySimAPIs\4.RflySimModel\2.AdvExps\e3_FWingModelCtrl\

文件夹/文件名称	说明
1.FixWingModelCtrlColl\Readme.pdf	带碰撞检测及响应的固定翼建模仿真实验步骤
2.FWPosCtrlAPI\Readme.pdf	固定翼航点控制实验步骤
3.FWAttCtrlAPI\Readme.pdf	固定翼俯仰角控制实验步骤
4.VelAltYawCtrlAPI_Py\Readme.pdf 5.VelAltYawCtrlAPI_Mat\Readme.pdf	固定翼速度/高度/偏航控制实验步骤
6.CopterSimSILNoPX4\Readme.pdf	固定翼综合模型仿真实验步骤
7.AdaptationTutorial\Readme.pdf	固定翼模型参数适配实验步骤

2. 总体说明

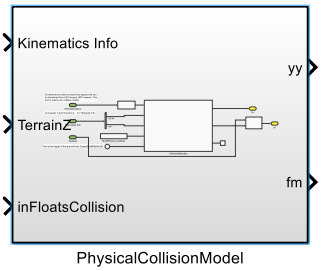
本例程库是相对于平台固定翼最小系统模型的进阶版版本，主要实验目的是在最小模板的基础上扩展功能模块和接口，以实现更复杂的仿真，学习本例程库之前应首先了解：

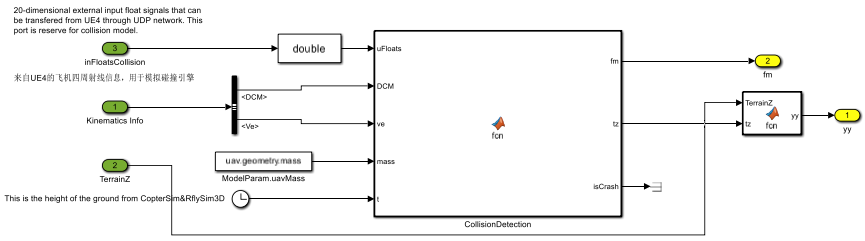
- 平台固定翼最小系统模型软硬件在环仿真相关例程：
  ..\..\1.BasicExps\e2_FixWingModelCtrl\Intro.pdf
- RflySim 载具建模模板之最小系统建模相关例程：..\e1_MinModelTemp\Intro.pdf
- Simulink 自主代码生成相关例程：..\..\0.ApiExps\2.UserDefinedC++\Intro.pdf
- ../../API.pdf 中环境配置和基本操作流程。
- 使用 RflySim 平台进行载具软硬件在环仿真时，需要将 DLL(windows 下)/SO（Linux 下）模型导入到 CopterSim，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。

3. 关键功能实现原理

3.1. 固定翼碰撞检测的实现

本模型相对于固定翼最小模板新增了输入接口 inFloatsCollision，可以回传 RflySim3D 中的飞机四周射线信息，通过新增模块 PhysicalCollisionModel 模拟了一个简单的碰撞引擎，可以根据计算出相对于机体坐标系的碰撞力和力矩以及发生碰撞时的地面高度。





inFloatsCollision 数据解析

CollisionDetection 碰撞检测及响应逻辑

详见四旋翼碰撞检测的实现：..\e2_MultiModelCtrl\Intro.pdf

模拟碰撞后坠机触地

详见四旋翼碰撞检测的实现：..\e2_MultiModelCtrl\Intro.pdf

3.2. 固定翼航点控制的实现

惯性系下（NED）航路寻迹接口 SendPosNED

通过外部控制接口（python）进行单机航点控制

单机控制脚本 AircraftMathworksController.py 中依次调用了 RflySim 平台飞机控制接口协议文件 PX4MavCtrlV4.py 中定义的以下接口函数

创建通信示例

```
mav1 = PX4MavCtrl.PX4MavCtrler(1)
```

创建一架飞机的通信示例

启用 Mavlink 消息监听循环

```
mav1.InitMavLoop()
```

配置 CopterSim 通信模式，该函数的参数定义如下：

```
def InitMavLoop(self,UDPMode=2):
        """ Initialize MAVLink listen loop from CopterSim
            0 and 1 for UDP_Full and UDP_Simple Modes, 2 and 3 for MAVLink_Full and MAVLink_Simple modes, 4 for MAVLink_NoSend
            The default mode is MAVLink_Full
        """
```

默认通信模式为 **Mavlink_Full**：Python 直接发送 MAVLink 消息给 CopterSim，再转发给 PX4，数据量较大适合单机控制；适合单机或少量飞机仿真，无人机数量小于 4；

设定航路点

```
n = 30
r = 400
missionPoints=[]
for i in range(n):
    angle = 2*math.pi*i/n
    x=r*math.sin(angle)
    y=r*math.cos(angle)
    missionPoints.append([x,y,-100])
```

用一组离散的点模拟圆形运动轨迹，并在循环中通过 append 方法逐个将相应的轨迹点存入目标点列表（missionPoints）。missionPoints.append([x,y,-100])表示在 missionPoints 列表的末尾添加一个新的列表[x,y,-100]。

根据欧拉公式：

$$e^{ix} = \cos\ x + i\sin\ x$$

这些点将在 x-y 平面上形成一个圆形轨迹。

分阶段执行飞行任务

完成上述设置后，程序会通过检查一个 flag 变量的值来决定无人机应该执行哪些动作。

当 flag == 0 时，解锁飞机

解锁飞机

```
mav1.SendMavArm(True)
```

设定起飞目标点

```
targetPos=[200, 0, -100]
mav1.sendMavTakeOff(targetPos[0],targetPos[1],targetPos[2])
```

发送绝对的 GPS 坐标作为起飞目标点，使用 sendMavTakeOffGPS 命令，最后三位分别是经度、维度、和高度，会先从 uavPosGPSHome 向量中提取解锁 GPS 坐标，在此基础上用绝对坐标

当 flag == 1 时，无人机起飞和进入航路寻迹模式

位置检测

```
        curPos=mav1.uavPosNED
        dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

计算飞机当前位置和起飞目标位置的水平距离，用于判断是否到达目标位置，以开始下一阶段任务。

启动外部控制（offboard）

```
mav1.initOffboard()
```

使 px4 控制器进入外部控制模式，且以 30HZ 的频率发送 offboard 指令

航路寻迹模式

```
targetPos=missionPoints[flagI]
mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

会通过航路点索引 flagI 的值从 missionPoints 列表中读取相应的航点，并通过 SendPosNED 函数更新为下一个目标点。

当 flag == 2 时，无人机在航路点之间的飞行

```
targetPos=missionPoints[flagI]
```

更新目标位置为 missionPoints 列表中的下一个点。

```
          curPos=mav1.uavPosNED
            dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

再次计算无人机与目标位置之间的距离。

```
            if dis < 50:
                flagI=flagI+1
                spd = 10+flagI/3.0
```

```
                mav1.SendCruiseSpeed(spd)
```

如果距离小于 50 米，更新 flagI 以切换到下一个航路点，并调整无人机的巡航速度。

```
            else:
                mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

如果距离不满足条件，则继续向当前目标位置飞行。

通过外部控制接口（python）进行多机航点控制

多机控制脚本 AircraftMathworksController3.py 脚本的实现逻辑与单机控制相同，只是需要创建 3 架飞机，再将相同的控制指令复制 3 份

创建通信示例

```
VehilceNum = 3
mav=[]
for i in range(VehilceNum):
    mav=mav+[PX4MavCtrl.PX4MavCtrler(1+i)]
```

创建 3 架飞机的通信示例

启用 Mavlink 消息监听循环

```
for i in range(VehilceNum):
    mav[i].InitMavLoop()
```

配置 3 架飞机的 CopterSim 通信模式

3.3. 固定翼俯仰角控制的实现

通过外部控制接口（python）执行姿态控制

单机控制脚本 AircraftMathworksAttCtrl.py 中依次调用了 RflySim 平台飞机控制接口协议文件 PX4MavCtrlV4.py 中定义的以下接口函数

程序会通过检查一个 flag 变量的值来决定无人机应该执行哪些动作。

当 flag == 0 时，解锁飞机

解锁飞机

```
mav1.SendMavArm(True)
```

设定起飞目标点

```
targetPos=[200, 0, -100]
mav1.sendMavTakeOff(targetPos[0],targetPos[1],targetPos[2])
```

发送绝对的 GPS 坐标作为起飞目标点，使用 sendMavTakeOffGPS 命令，最后三位分别是经度、维度、和高度，会先从 uavPosGPSHome 向量中提取解锁 GPS 坐标，在此基础上用绝对坐标

当 flag == 1 时，无人机起飞和初始航迹

位置检测

```
        curPos=mav1.uavPosNED
        dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

计算飞机当前位置和起飞目标位置的水平距离，用于判断是否到达目标位置，以开始下一阶段任务。

启动外部控制（offboard）

```
mav1.initOffboard()
```

使 px4 控制器进入外部控制模式，且以 30HZ 的频率发送 offboard 指令

航路寻迹模式

```
targetPos=missionPoints[flagI]
mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

会通过航路点索引 flagI 的值从 missionPoints 列表中读取相应的航点，并通过 SendPosNED 函数更新为下一个目标点。

当 flag == 2 时，无人机姿态和油门的控制

```
mav1.SendAttPX4([0,10,0],mav1.uavThrust)
```

这里设置无人机的俯仰角为 10 度，并保持油门值为悬停油门。这是为了改变无人机的飞行方向。

3.4. 固定翼高度/速度/偏航控制的实现

通过外部控制接口（python）执行速度/高度/偏航控制

单机控制脚本 AircraftMathworksController.py 中依次调用了 RflySim 平台飞机控制接口协议文件 PX4MavCtrlV4.py 中定义的以下接口函数

程序会通过检查一个 flag 变量的值来决定无人机应该执行哪些动作。

当 flag == 0 时，解锁飞机

解锁飞机

```
mav1.SendMavArm(True)
```

设定起飞目标点

```
targetPos=[200, 0, -100]
mav1.sendMavTakeOff(targetPos[0],targetPos[1],targetPos[2])
```

发送绝对的 GPS 坐标作为起飞目标点，使用 sendMavTakeOffGPS 命令，最后三位分别是经度、维度、和高度，会先从 uavPosGPSHome 向量中提取解锁 GPS 坐标，在此基础上用绝对坐标

当 flag == 1 时，无人机起飞和进入航路寻迹模式

位置检测

```
        curPos=mav1.uavPosNED
        dis = math.sqrt((curPos[0]-targetPos[0])**2+(curPos[1]-targetPos[1])**2)
```

计算飞机当前位置和起飞目标位置的水平距离，用于判断是否到达目标位置，以开始下一阶段任务。

启动外部控制（offboard）

```
mav1.initOffboard()
```

使 px4 控制器进入外部控制模式，且以 30HZ 的频率发送 offboard 指令

航路寻迹模式

```
targetPos=missionPoints[flagI]
mav1.SendPosNED(targetPos[0], targetPos[1], targetPos[2])
```

会通过航路点索引 flagI 的值从 missionPoints 列表中读取相应的航点，并通过

SendPosNED 函数更新为下一个目标点。

当 flag == 2 时，发送期望速度、偏航和高度

```
mav1.SendVelYawAlt(10,math.pi/2,-150)
```

发送期望速度、偏航和高度，测试命令是否能正确运行，可以观测速度是否为 10，高度是否为 150

4. 相关文献

[1]. RflySim 载具建模模板之最小模板建模实验原理：..\e1_MinModelTemp\Intro.pdf

[2]. inCopterData 输入接口验证实验原理：..\..\0.ApiExps\14.inCopterData\Intro.pdf

附加资源

官方文档：RflySim 官方文档：https://rflysim.com/doc/zh/

社区交流：加入 RflySim 技术交流群：951534390