平台多旋翼模型控制仿真实验原理

1. 文件目录

例程目录：[安装目录]\RflySimAPIs\4.RflySimModel\2.AdvExps\e2_MultiModelCtrl\

文件夹/文件名称	说明
1.MultiModelCtrl\Readme.pdf 4.HexModelCtrl\Readme.pdf 5.OctoCoxRotor\Readme.pdf 6.OctoX\Readme.pdf	利用建模模版实现各种机型多旋翼建模仿真实验步骤
2.MultiModelCtrlColl\Readme.pdf	带碰撞检测及响应的四旋翼建模仿真实验步骤
3.CopterSimSILNoPX4\Readme.pdf 10.HexarotorNoPX4\readme.pdf 11.OctoCoxRotorNoPX4\readme.pdf 12.Octorotor_XNoPX4\Readme.pdf 13.MixedMultiRotorNoPX4_Mat\Readme.pdf 14.MixedMultiRotorNoPX4_Py\Readme.pdf	多旋翼综合模型仿真实验步骤
7.IdentificationModel\FX150\Readme.pdf 7.IdentificationModel\FX200\Readme.pdf 7.IdentificationModel\FX450\Readme.pdf	四旋翼系统辨识模型建模仿真实验步骤
8.AdaptationTutorial\Readme.pdf	多旋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最小系统载具建模模板的进阶版本，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：

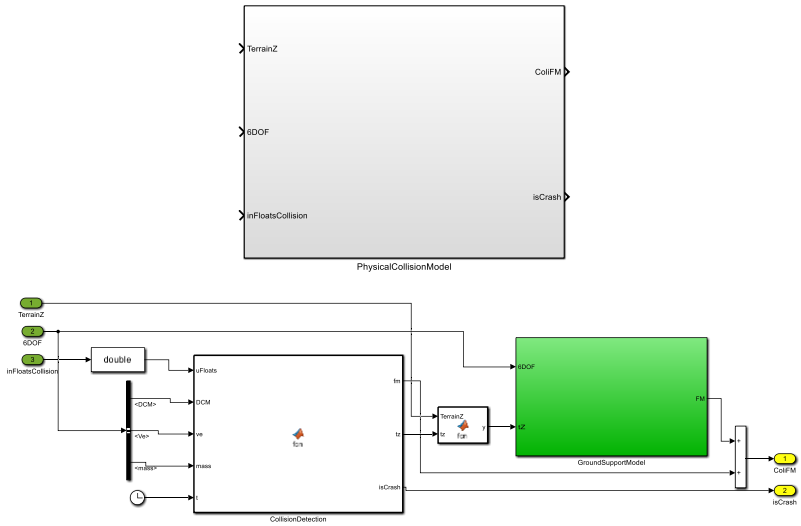
- RflySim 载具建模模板之最小系统建模相关例程：..\e1_MinModelTemp\Intro.pdf
- Simulink 自主代码生成相关例程：..\..\0.ApiExps\2.UserDefinedC++\Intro.pdf
- ../../API.pdf 中环境配置和基本操作流程。
- 使用 RflySim 平台进行载具软硬件在环仿真时，需要将 DLL(windows 下)/SO（Linux 下）模型导入到 CopterSim，形成运动仿真模型，因此，在 Simulink 模型编译完成后，需要将模型对应的 C++文件打包成 DLL/SO 模型。

3. 关键功能实现原理

3.1. 四旋翼模型碰撞检测及响应的实现

本模型相对于平台最小系统建模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 碰撞反馈力和力矩计算

声明和初始化持久变量

坠机标志位（isCol）：用于指示是否与非场景固有物体发生了碰撞。初始值为 0（表示没有碰撞），如果检测到碰撞，则设置为 1。

```
persistent isCol;
if isempty(isCol)
    isCol=int8(0);
end
```

输出力和力矩暂存向量（fOut）：存储因碰撞而产生的力和力矩的向量。初始化为零向量。

```
persistent fOut;
if isempty(fOut)
    fOut=[0;0;0;0;0;0];
end
```

碰撞冲量矩阵（mv0）：存储碰撞产生的初始冲量向量。初始化为零向量。

```
persistent mv0;
if isempty(mv0)
    mv0=[0;0;0];
end
```

碰撞时间（tColi）：记录碰撞发生的时间。初始值为 0。

```
persistent tColi;
if isempty(tColi)
    tColi=0;
end
```

碰撞检测逻辑

检查是否收到碰撞消息：通过检查输入向量 uFloats （来自 inFloatsCollision 接口）的第

一个元素是否接近 12345（数据校验位）来确定是否收到碰撞消息。uFloats(2) 大于 0.5 表示被碰撞物体非场景固有物体。

```
       if abs(uFloats(1)-12345)<1 && uFloats(2)>0.5 %如果收到碰撞消息，12345 是数据校验位，uFloats(2)是碰撞到飞机的 ID
            isCol=int8(1);%设置为碰撞模式
```

碰撞类型：如果检测到碰撞，根据碰撞对象是移动物体（如其他飞行器）还是静止物体（如树或房子）来计算新的速度向量 veNew。

碰撞到移动物体：使用冲量定理计算新的速度。

```
    if uFloats(3)>0 %被碰撞物与本物体的质量比有正值，说明碰撞物也是飞机，需要用冲量定理
                massOb=mass*uFloats(3)^2;
                veOb = uFloats(7:9);
                veNew=(mass*ve+massOb*veOb)/(mass+massOb);
```

碰撞到静止物体（非地形）：以原速度的十分之一反向弹回。

```
    else %被碰撞物为树、房子等固定物体，质量无穷大，直接反弹回去（1/10 的速度）
                veNew=-ve/10.0;
```

记录碰撞冲量和时间：计算碰撞产生的冲量 mv0 并记录碰撞时间 tColi。

```
    mv0=mass*(veNew-ve);
    tColi=t;%记录碰撞时间
```

移动物体碰撞响应

碰撞后的冷却时间：如果自碰撞以来的时间超过 0.05 秒，则将输出力和力矩向量 fOut 重置为零向量。

```
    if (t-tColi)>0.05
            fOut=[0;0;0;0;0;0];
```

在碰撞冷却时间内的处理：如果在冷却时间内（0.05 秒以内），根据碰撞冲量矩阵 mv0 计算瞬时作用力，以模拟碰撞冲量的作用。随机因素（rand() 函数）被用来引入一定的变异，使碰撞响应更加真实。

```
    else
            %用瞬时的作用力模拟冲量作用（0.05s）
            mv=DCM*mv0;
            mv(1)=mv(1)*(0.7+0.3*rand());
            mv(2)=mv(2)*(0.7+0.3*rand());
            mv(3)=mv(3)*(0.7+0.3*rand());
            fOut(1:3) = mv/0.05;    %冲量除以时间得到力
            fOut(4:6) = rand(3,1)*mass;
```

场景固定物体（如地形）碰撞响应

初始化力和力矩向量 (fm)，这是基于之前计算出的碰撞响应力和力矩 (fOut)。

```
    fm = fOut;
```

初始化与地面和障碍物的距离信息 (tz) **和障碍物碰撞反馈力向量 (ddm)。**tz 初始值表明飞机默认高度为 100 单位，且位于原点。

```
    tz=[100;0;0;0];
    ddm=[0;0;0;0;0;0];
```

碰撞检测

是否发生碰撞数据校验：通过比较 uFloats(1) 与 12345 的接近度来确认是否接收到有效

的碰撞消息。

```
if abs(uFloats(1)-12345)<1 %如果收到碰撞消息
```

场景固定物体碰撞处理：如果检测到碰撞的对象是场景固定物体（uFloats(2) < 0.5），则更新飞机的坐标以及距离地面的高度。

```
if uFloats(2)<0.5 %如果碰撞的是场景固定物体
    %向下照射射线返回长度
    tz(1)=uFloats(15);
    %飞机在 UE4 中的坐标
    tz(2:4)=uFloats(4:6);
end
```

障碍物碰撞反馈力的计算

计算障碍物碰撞反馈力：通过 uFloats(10) 到 uFloats(14) 分别判断与前、后、左、右、上障碍物的距离。如果距离小于 0（意味着已经发生碰撞），则根据两者碰撞距离相重叠的部分的具体值计算反馈力。力的计算根据重叠距离的范围分为三级，重叠距离越大，施加的反馈力越大。

```
%前后左右是以地面坐标系的北东地为准
%前障碍物碰撞反馈力
if uFloats(10)<0 %距前墙面距离，小于 0 说明已经接触
    z=uFloats(10);
    if z>-0.01
        ddm(1)=-10*ve(1);
    elseif z>-0.04
        ddm(1)=200*z-20*ve(1);
    else
        ddm(1)=500*z-50*ve(1);
    end
end
%后障碍物碰撞反馈力
if uFloats(11)<0
    z=uFloats(11);
    if z>-0.01
        ddm(1)=-10*ve(1);
    elseif z>-0.04
        ddm(1)=-200*z-20*ve(1);
    else
        ddm(1)=-500*z-50*ve(1);
    end

end
%左障碍物碰撞反馈力
if uFloats(12)<0
    z=uFloats(12);
    if z>-0.01
        ddm(2)=-10*ve(2);
    elseif z>-0.04
        ddm(2)=-200*z-20*ve(2);
    else
        ddm(2)=-500*z-50*ve(2);
    end
end
%右障碍物碰撞反馈力
```

```
        if uFloats(13)<0
            z=uFloats(13);
            if z>-0.01
                ddm(2)=-10*ve(2);
            elseif z>-0.04
                ddm(2)=200*z-20*ve(2);
            else
                ddm(2)=500*z-50*ve(2);
            end
        end
        %上障碍物碰撞反馈力
        if uFloats(14)<0
            z=uFloats(14);
            if z>-0.01
                ddm(3)=-10*ve(3);
            elseif z>-0.04
                ddm(3)=-200*z-20*ve(3);
            else
                ddm(3)=-500*z-50*ve(3);
            end
        end
```

力的坐标映射

地面坐标系到机体坐标系的映射：使用方向余弦矩阵（DCM）将计算出的反馈力（ddm）从地面坐标系转换到机体坐标系，以便正确地应用这些力。

```
    ddmm=DCM*ddm(1:3);
```

更新力和力矩向量 (fm)：将计算并映射后的碰撞反馈力加到之前的力和力矩向量上。

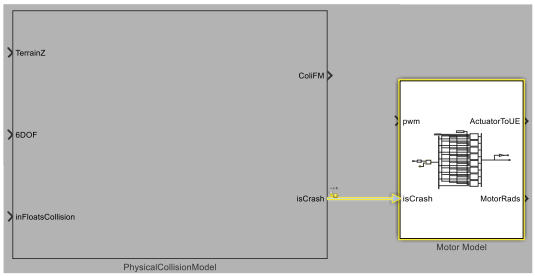
```
    fm=fm+[ddmm;0;0;0];
```

isCrash 坠机标志位

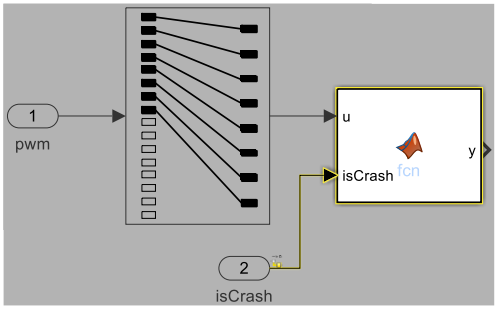
```
    isCrash=isCol;
```

与**非地形**之类场景固有物体发生碰撞后坠机标志位 isCrash 会赋值为 1。

isCrash 坠机标志位触发电机故障



电机模块（Motor Model）故障触发逻辑实现如下：



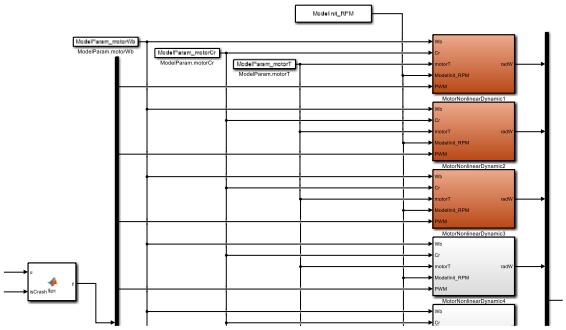

```
function y = fcn(u,isCrash)
y = u;
if isCrash>0.5
    y(1)=rand()/2.0;
    y(2)=rand()/5.0;
    y(3)=rand()/10.0+u(3)*0.9;
end
```

初始化输出向量：y = u;首先将输入向量 u（来自 pwm 值的前 8 位）直接赋值给输出向量 y，此时 y 和 u 是相同的。

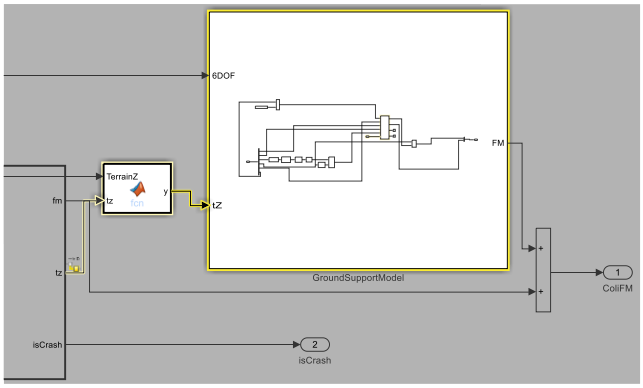
判断是否发生碰撞：if isCrash > 0.5 这一行判断是否发生了碰撞。isCrash 参数是一个标志变量，当其值大于 0.5 时，表示飞机发生了相撞。

调整电机性能：如果发生碰撞（isCrash > 0.5），则执行以下调整：

- y(1) = rand()/2.0;：第一个电机的性能随机降低到原来的 0 到 50%之间。
- y(2) = rand()/5.0;：第二个电机的性能随机降低到原来的 0 到 20%之间。
- y(3) = rand()/10.0 + u(3) * 0.9;：第三个电机的性能随机降低到原来的 0 到 10%之间，但保留了原始性能（由 u(3)给出）的 90%作为基础。



模拟碰撞后坠机触地反弹



计算发生碰撞位置的地形高度

```
function y = fcn(TerrainZ, tz)
y=TerrainZ;
if tz(1)<=99
    y=tz(4)+tz(1);
end
```

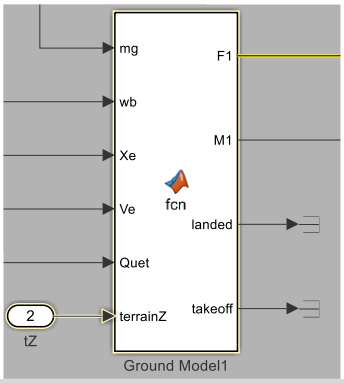
初始化输出：y=TerrainZ;首先将地形的高度值 TerrainZ 赋值给输出变量 y。这意味着在不进入 if 语句的情况下，函数默认返回地形的高度值。

飞机位置判断：if tz(1)<=99 判断了飞机距离地面高度是否小于或等于 99m。如果条件成立，说明飞机已经发生了碰撞，此时需要计算碰撞点处的地形高度。

调整输出值：y=tz(4)+tz(1);如果飞机的高度小于或等于 99m，则执行 y=tz(4)+tz(1)。这里，tz(4)是 tz 向量的第四个元素，代表飞机在 Z 轴（NED 地面系，正方向位负值）上的坐标。函数将飞机的 Z 坐标和相对地面的高度相加，然后将这个结果赋给 y。

Ground Model 计算地面反作用力

这个计算考虑了无人机与地面接触的软硬程度和速度，以模拟不同的接触条件。具体实现如下：




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

计算飞机距离地面的高度，这里 terrainZ 来自碰撞点的地形高度

```
F1=[0;0;0];
```

F1 初始化为零向量，代表反作用力。

```
if z>=0
    if z<0.05
         F1(3)=-mg(3)-200*z-200*Ve(3);
    elseif z<0.1
        F1(3)=-mg(3)-500*z-500*Ve(3);
    else
         F1(3)=-mg(3)-1000*z-1000*Ve(3);
    end
    F1(1)=-50*Ve(1);
    F1(2)=-50*Ve(2);
end
```

当 z>=0 时，即无人机处于地面下方时，根据无人机的离地面高度 z（此时表示无人机与地形碰撞产生的形变大小，用于模拟无人机与地面接触的软硬程度）和垂直速度 Ve(3)来计算反作用力 F1(3)。

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