1. 实验名称及目的

1.1. 实验名称

dll 模型带碰撞检测及电机故障触发模块的四旋翼模型 SIL/HIL 实验

1.2. 实验目的

该例程中的四旋翼模型具备碰撞检测模块，同时具备电机故障触发模块，当飞机在飞行过程中撞上另一架飞机时，该飞机会电机损坏并坠毁，通过该例程熟悉碰撞检测功能的使用。

1.3. 关键知识点

1.3.1 碰撞数据来源

RflySim3D 中根据载具模型的物理资产包围盒，以 6 面射线相交测试获得的距离数据，快速回传给 CopterSim 中 dll 模型的 InFloatsCollision 接口

InFloatsCollision 接口数据解析如下

1 校验位 12345

2 被碰物体 id

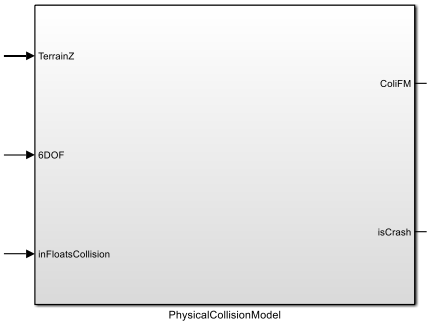
3 质量比

4~6 被碰物体位置

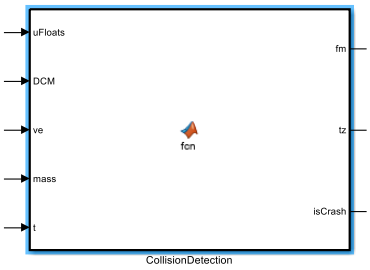
7~9 被碰物体速度

10~15 六面射线返回距离

1.3.2 四旋翼模型碰撞检测及响应的实现



碰撞检测和响应的功能主要通过 PhysicalCollisionModel 模块实现，其中的模块模拟了与移动物体和场景固定物体的交互



移动物体反作用力计算如下：

```
massOb=mass*uFloats(3)^2; 计算被碰撞物的广义质量，uFloats(3)表示质量比
veOb = uFloats(7:9); 获取被碰撞物的速度
veNew=(mass*ve+massOb*veOb)/(mass+massOb); 计算碰撞后本飞机的新速度，本飞机和被碰撞物的动量之和除以它们的质量之和。
mv0=mass*(veNew-ve); 计算冲量，本飞机的质量乘以速度的变化
mv=DCM*mv0; 将冲量旋转到体坐标系下。
mv(1)=mv(1)*(0.7+0.3*rand());：计算 xb 方向上的冲量。原始冲量乘以一个介于 0.7 和 1 之间的随机数以模拟不确定性。
mv(2)=mv(2)*(0.7+0.3*rand());：计算 yb 方向上的冲量，处理方式与 x 方向相同。
mv(3)=mv(3)*(0.7+0.3*rand());：计算 zb 方向上的冲量，处理方式与 x 方向相同。
fOut(1:3) = mv/0.05;     冲量除以时间得到反作用力
```

场景固定物体反作用力计算如下：

```
%后障碍物碰撞反馈力
        z=uFloats(11);获取与后障碍物的碰撞距离重叠部分
根据重叠距离不同调整 kn 和 kc 计算F_n = k_n δ_n^α + k_c δ̇_n
        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
```

1.3.3 Python 脚本控制 2 架四旋翼相撞

关键代码如下：

```
ue = UE4CtrlAPI.UE4CtrlAPI()
```

为打开的 RflySim3D 创建 1 个通信实例

```
mav = PX4MavCtrl.PX4MavCtrler(1)
mav2 = PX4MavCtrl.PX4MavCtrler(2)
```

为打开的 2 个 CopterSim 创建 2 个通信实例

```
mav.SendPosNED(0, 0, -5, 0)
mav2.SendPosNED(2, -2, -5, 0)
```

利用位置控制接口为两架飞机设置目标航点

```
mav.SendMavArm(True)
mav2.SendMavArm(True)
```

解锁飞机，理论上此时飞机才开始起飞并向目标点运动，但这里的 dll 模型的输入没有解锁标志位，所以这里没有实际作用

```
ue.sendUE4Cmd('RflyChangeViewKeyCmd P',0)  #开启碰撞引擎
```

开启 RflySim3D 的碰撞模式

```
mav.SendVelNED(1, 0, 0, 0)
```

限制 1 号飞机的 x 方向速度最大为 1m/s

下列代码主要用于检测并处理两个飞行器（简称为 V1 和 V2）的碰撞情况。

初始化变量

```
hasV1Crashed=False
hasV2Crashed=False
startTime = time.time()
lastTime = time.time()
```

```
timeInterval = 1/30.0 # time interval of the timer
```

- hasV1Crashed 和 hasV2Crashed：这两个布尔变量用于追踪每个飞行器是否已经发生了碰撞。初始值都设置为 False，表示开始时没有碰撞发生。
- startTime 和 lastTime：这两个变量记录了时间信息，用于控制循环的执行频率。startTime 在这段代码中没有被使用。
- timeInterval：定义了时间间隔，这里设置为 1/30.0，即每秒 30 帧的频率。这意味着代码的执行间隔大约是 0.033333 秒。

循环检测飞行器的碰撞状态

```
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()

    if (not hasV1Crashed) and mav.isVehicleCrash:
        print('Vehicle #1 Crashed with vehicle #',mav.isVehicleCrashID)
        hasV1Crashed=True

    if (not hasV2Crashed) and mav2.isVehicleCrash:
        print('Vehicle #2 Crashed with vehicle #',mav2.isVehicleCrashID)
        hasV2Crashed=True

    if hasV2Crashed and hasV1Crashed:
        time.sleep(4)
        break
```

2. 实验效果

将带碰撞模块的四旋翼模型生成 DLL 文件后，通过指定脚本启动 PX4 软件在环仿真，运行 CollisionDemo.py，即可在 RflySim3D 中看到两架四旋翼在初始化后起飞到空中、开启碰撞检测、相撞后触发电机故障坠地。电机损坏，飞机再也无法起飞

3. 文件目录

例程目录：[安装目录]\RflySimAPIs\4.RflySimModel\0.ApiExps\10.InCollisionAPI\1.inFloatsCollision\3.inFloatsCollisionMotorFail\

文件夹/文件名称	说明
..\Intro.pdf	基于 InFloatsCollision 接口的碰撞引擎实现实现原理
inFloatsCollisionMotorFail.slx	带碰撞及电机故障模块的四旋翼飞机模型文件。
inFloatsCollisionApiDemo_SITL.bat	软件在环仿真批处理文件。
GenerateModelDLLFile.p	DLL 格式转化文件。
inFloatsCollisionApiDemo_init.m	动力学模型相关参数。

MavLinkStruct.mat	MavLink 数据结构体 mat 文件
CollisionDemo.py	碰撞检测验证脚本
Python38Run.bat	Python 程序运行脚本

4. 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 工具链	Pixhawk 6X 或其它飞控②	1
3	MATLAB 2017b 及以上③	数据线	1

① ：推荐配置请见：https://rflysim.com/

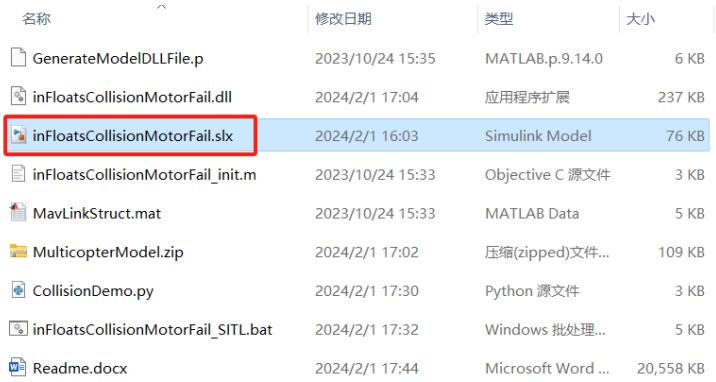
② 若使用 Pixhawk 6X 飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐 PX4 固件版本为：1.12.3。其他配套飞控及编译命令请见：https://rflysim.com/doc/zh/1/Hardware.html。

5. 实验步骤

5.1. 必做实验：DLL 模型生成

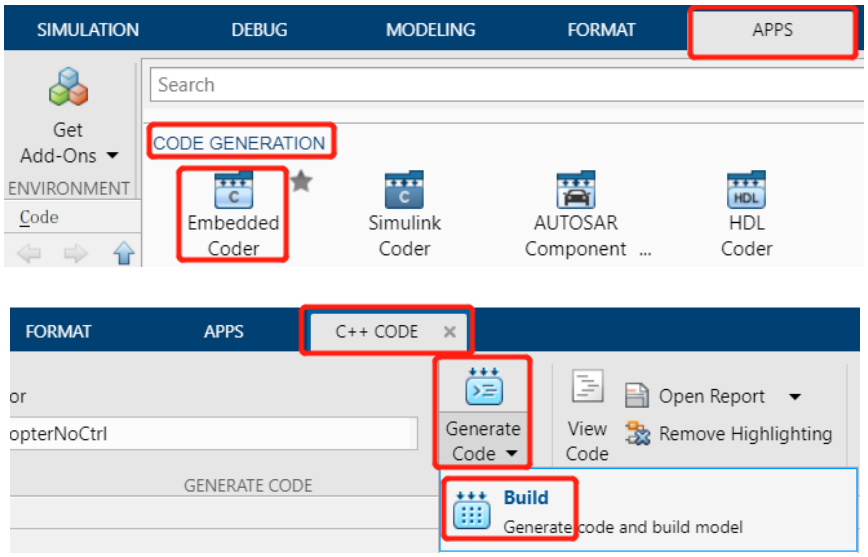
Step 1：编译模型

在 Matlab 中打开“inFloatsCollisionMotorFail.slx”文件，点击 Build Model 按钮生成代码。



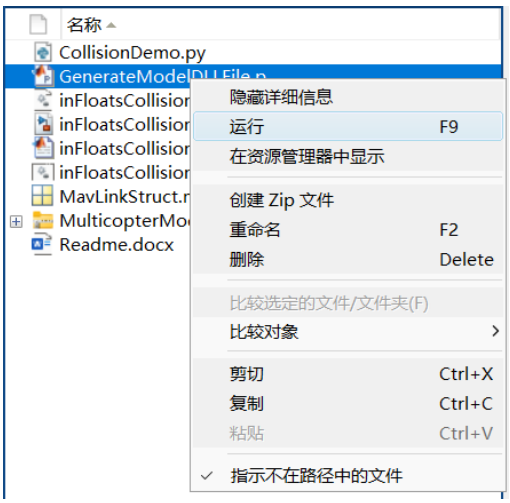
对于 MATLAB 2019a 及之前版本，工具栏样式见下图，直接点击它的编译按钮“Build”即可。

对于 2019b 及之后版本，点击 APPS - CODE GENERATION –Embedded Coder 才能弹出代码生成工具栏，在其中如下图所示点击“C++CODE” -“Generate Code”-“Build”按钮就能编译生成代码。



Step 2：生成 DLL 文件

模型编译完成后，在 matlab 中右键“GenerateModelDLLFile.p”文件，点击运行，生成 DLL 文件。



5.2. 必做实验：软件在环仿真

Step 1：启动仿真

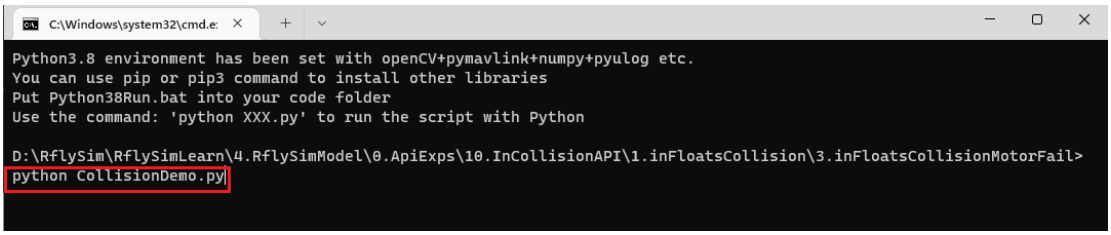
双击运行“inFloatsCollisionMotorFail_SITL.bat”批处理文件，自动启动两架飞机的软件在环仿真。

Step 2：等待初始化完成

等待 RflySim3D 中显示两架四旋翼初始化完毕。

Step 3：运行控制程序

在文件夹下，双击 Python38Run.bat，打开集成好的 python 环境，在该环境下运行 CollisionDemo.py 文件，输入 python CollisionDemo.py

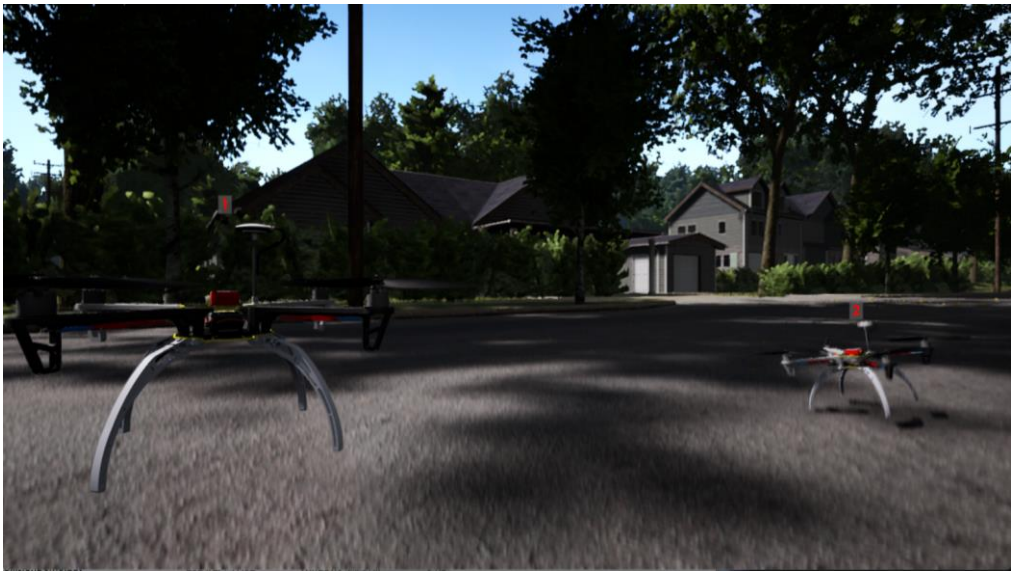


Step 4：观察结果

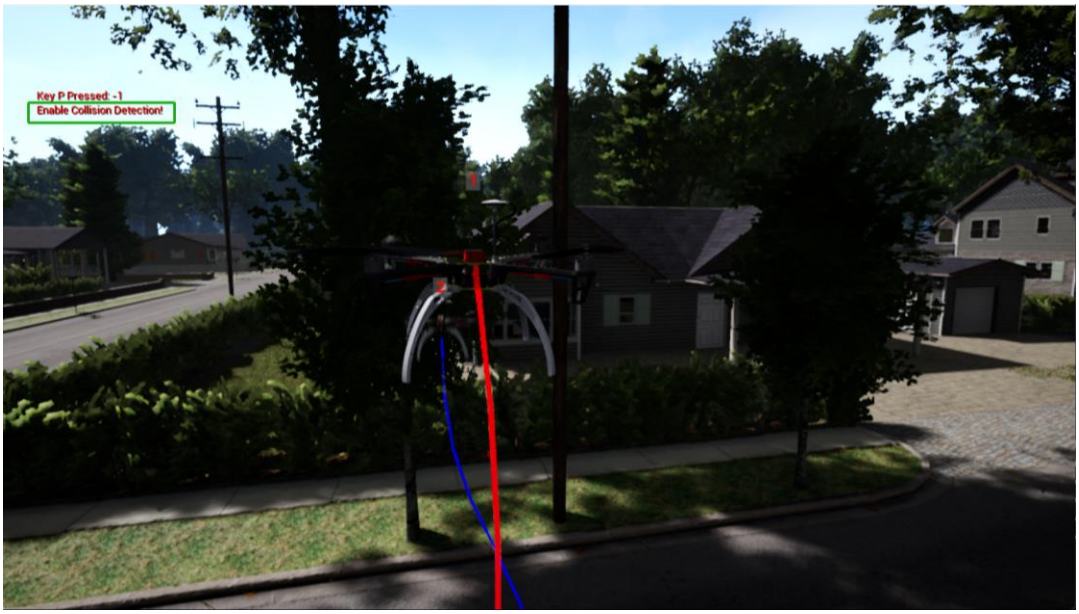
在 RflySim3D 中观察仿真效果。

首先，会看到两架四旋翼开启了标号和轨迹。

然后，可以看到四旋翼分别从初始点开始起飞。



起飞到空中后，开启碰撞使能。



飞机相撞后，在 RflySim3D 左上角能看到碰撞信息，并且能看到碰撞发生后，四旋翼触发了电机故障坠地。

5.3. 选做实验（VS Code 调试运行）

准备工作：

- 先确保已经按 RflySimAPIs\1.RflySimIntro\2.AdvExps\e3_PythonConfig\Readme.pdf 步骤，正确配置 VS Code 环境。或者配置了自己的 Pycharm 等自定义 Python 环境。
- 其他步骤与上文相同，运行 CollisionDemo.py 时，可使用 VS Code（或 Pycharm 等工具）来打开 CollisionDemo.py 文件，并阅读代码，修改代码，调试执行等。

扩展实验：

- 请自行使用 VS Code 阅读 CollisionDemo.py 源码，通过程序跳转，了解每条代码的执行原理；再通过调试工具，验证每条指令的执行效果。

```
import PX4MavCtrlV4 as PX4MavCtrl
import UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()

#Create a new MAVLink communication instance, UDP sending port (CopterSi
mav = PX4MavCtrl.PX4MavCtrler(1)
mav2 = PX4MavCtrl.PX4MavCtrler(2)

ue.sendUE4Cmd('RflyChangeViewKeyCmd S',0) #开启标号显示
time.sleep(0.5)
ue.sendUE4Cmd('RflyChangeViewKeyCmd T',0) #开启标号显示
time.sleep(0.5)


#Turn on MAVLink to monitor CopterSim data and update it in real time.
mav.InitMavLoop()
mav2.InitMavLoop()
time.sleep(0.5)


#Display Position information received from CopterSim
print(mav.uavPosNED)


#Turn on Offboard mode
mav.initOffboard()
mav2.initOffboard()
```

6. 参考资料

[1]. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中 DLL/SO 模型与通信接口的重要参数

部分。

[2]. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中的环境配置

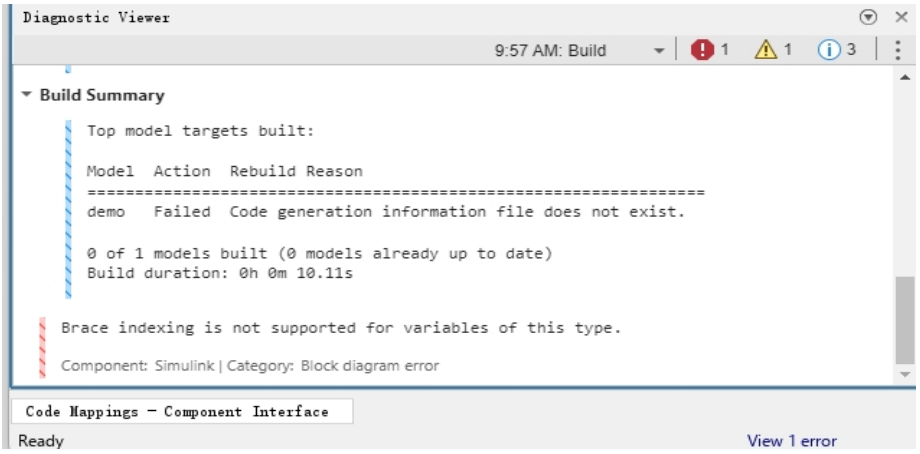
[3]. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中的 Simulink 建模模板介绍

[4]. PX4PSP\RflySimAPIs\4.RflySimModel\API.pdf 中 DLL/SO 模型与通信接口的数据协议部分

[5].

7. 常见问题

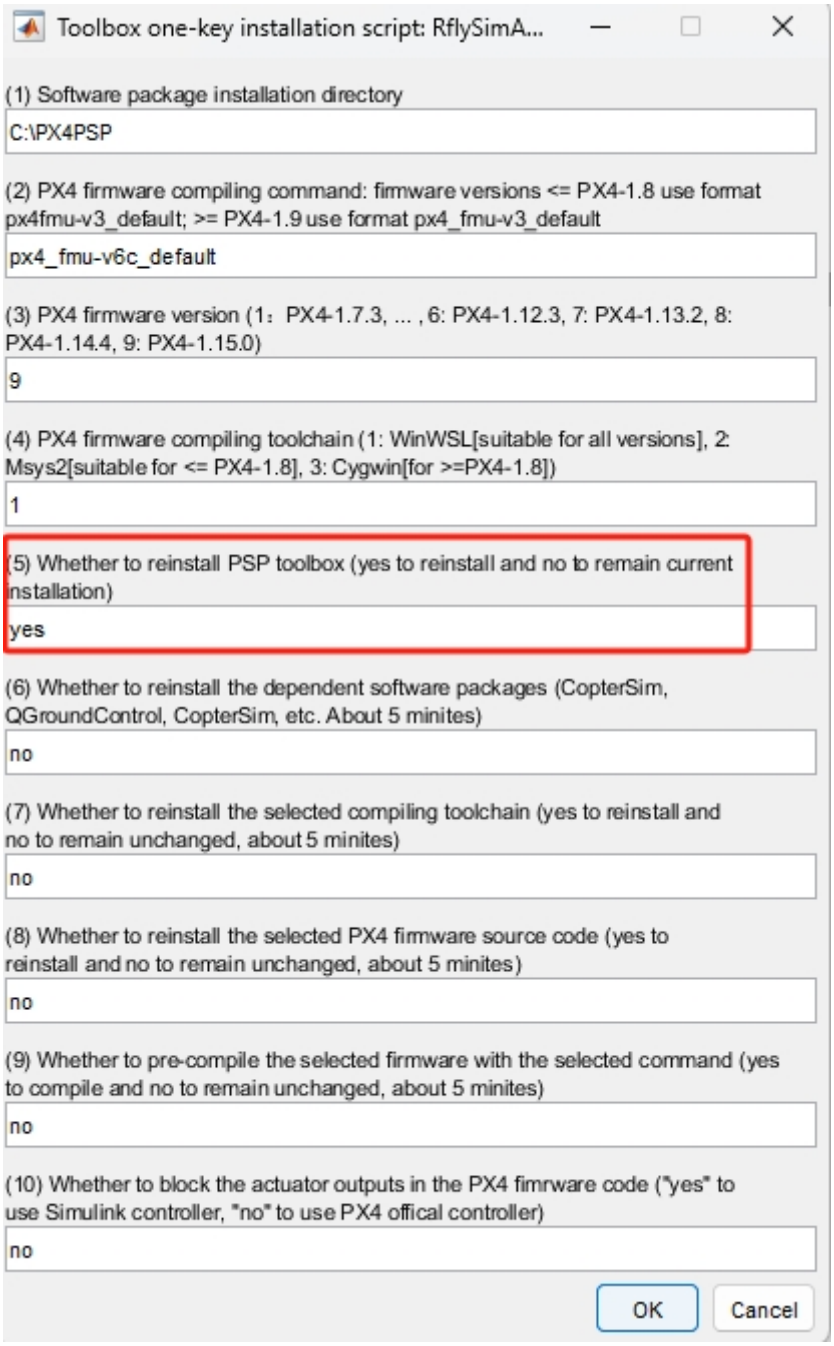
Q1：未正确安装 visual studio c++编译环境并配置 mex，导致 Simulink 文件编译失败



A1：首先将低于当前 MATLAB 版本的 Visual Studio C++编译环境安装到 VS 默认安装目录，然后在 MATLAB 的命令行窗口中输入指令“mex -setup”，一般来说会自动识别并安装上支持的编译器，命令行显示“MEX 配置使用‘Microsoft Visual C++ 2017’以进行编译”的字样说明安装正确。详细环境配置参考”[RflySim 平台安装目录]\RflySimAPIs\4.RflySimModel\API.pdf“中的环境配置

Q2：编译报错，无法加载库文件

A2：这可能是由于安装平台时 PX4PSP 工具箱未更新到最新版，更新 RflySim 安装包后按照如下配置重新安装平台即可


Toolbox one-key installation script: RflySimA...
(1) Software package installation directory
C:\PX4PSP
(2) PX4 firmware compiling command: firmware versions <= PX4-1.8 use format px4fmu-v3_default; >= PX4-1.9 use format px4_fmu-v3_default
px4_fmu-v6c_default
(3) PX4 firmware version (1：PX4-1.7.3, ... , 6: PX4-1.12.3, 7: PX4-1.13.2, 8: PX4-1.14.4, 9: PX4-1.15.0)
9
(4) PX4 firmware compiling toolchain (1: WinWSL[suitable for all versions], 2: Msys2[suitable for <= PX4-1.8], 3: Cygwin[for >=PX4-1.8])
1
(5) Whether to reinstall PSP toolbox (yes to reinstall and no to remain current installation)
yes
(6) Whether to reinstall the dependent software packages (CopterSim, QGroundControl, CopterSim, etc. About 5 minites)
no
(7) Whether to reinstall the selected compiling toolchain (yes to reinstall and no to remain unchanged, about 5 minites)
no
(8) Whether to reinstall the selected PX4 firmware source code (yes to reinstall and no to remain unchanged, about 5 minites)
no
(9) Whether to pre-compile the selected firmware with the selected command (yes to compile and no to remain unchanged, about 5 minites)
no
(10) Whether to block the actuator outputs in the PX4 fimrware code ("yes" to use Simulink controller, "no" to use PX4 offical controller)
no
OK
Cancel