inSILIntsFloats 外部数据输入接口实验

1. 文件目录

例程目录：[安装目录]\RflySimAPIs\4.RflySimModel\0.ApiExps\11.inSILAPI\1.InSILIntsFloats\

文件夹/文件名称	说明
1.InFaultAPITest\Readme.pdf 2.InSILIntsFloats_sim\Readme.pdf 3.inSILIntsFloats_py\Readme.pdf	基于 inSILIntsFloats 接口的电机故障注入实验步骤

2. 总体说明

2.1. 综合模型的需求

- 详细的综合模型协议参考**错误!未找到引用源。**

在原有动力学模型的基础上添加控制器，从而在 dll 模型中形成被控对象和控制器组成的完整控制闭环

2.2. 外部故障注入的需求

- 详细的故障注入模式参考**错误!未找到引用源。**

可以通过外部控制程序触发执行器故障，从而实现更灵活的故障注入模式。

2.3. 接口数据解析

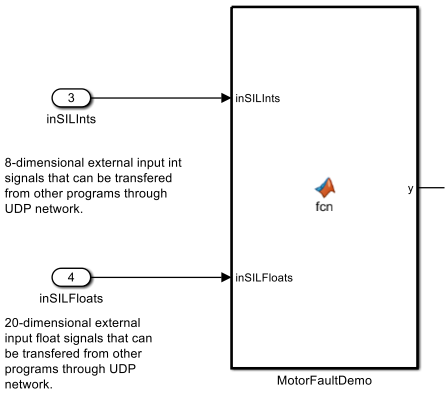
结构体定义如下：

```
struct PX4SILIntFloat{
    int checksum;//1234567897
    int CopterID;//飞机的 ID
    int inSILInts[8];//int 标志位
    float inSILFLoats[20];//float 参数位
};
```

3. 关键功能的实现

3.1. 电机故障模块的实现

- Simulink 模型见\1.InFaultAPITest\Exp2_MaxModelTemp.slx

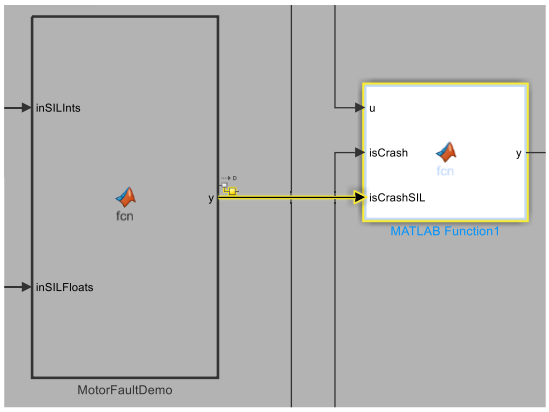



```
function y = fcn(inSILInts,inSILFloats)
y=0;
if inSILInts(1)>0.5 && inSILFloats(1)>0.5
    y=1;
end
```

首先，检查 inSILInts 的第一个元素和 inSILFloats 的第一个元素是否大于 0.5。

如果以上两个条件都满足，那么将 y 的值设置为 1。

如果任一条件不满足，y 的值保持为 0。




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

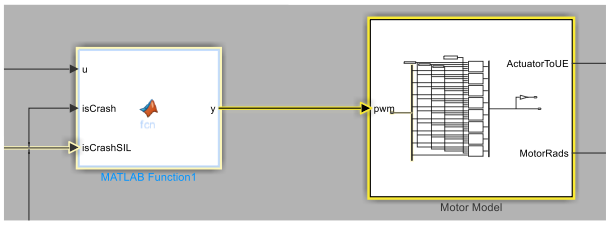
其中输入参数 u 来自 pwm 指令中的前 8 位，isCrash 是来自碰撞模块 CollisionDetection [4]的坠机标志位，isCrashSIL 来自电机故障模块 MotorFaultDemo 的输出 y

当 isCrashSIL 标志位为 1 时

y(1)随机数的范围在 0 到 0.5 之间触发 1 号电机故障。

y(2)随机数的范围在 0 到 0.2 之间触发 2 号电机故障。

y(3)随机数的范围在 0.9 到 1 之间触发 3 号电机故障。



3.2. Python 注入故障

关键接口函数 sendSILIntFloat

- 详细解析见**错误!未找到引用源。**中的 python 控制接口 DllSimCtrlAPI

接口使用示例解析

3.inSILIntsFloats_py\inSIL.py 关键代码如下：

导入必需的库

```
import numpy as np
from DllSimCtrlAPI import DllSimCtrlAPI
```

numpy：用于创建和操作数值数组，常用于科学计算。

DllSimCtrlAPI：从 DllSimCtrlAPI 模块导入的 DllSimCtrlAPI 类，这个类包含用于传输外部数据给 dll 模型的多个方法。

初始化 DLL 控制接口

```
dll1 = DllSimCtrlAPI()
```

创建 DllSimCtrlAPI 类的一个实例，命名为 dll1。

发送 inSILInts 整数数据

```
silInt = np.zeros(8).astype(int).tolist()
silInt[0] = 5
dll1.sendSILIntFloat(silInt)
```

使用 numpy 创建一个长度为 8 的整数数组 silInt，所有元素初始值为 0。

将 silInt 数组的第一个元素设置为 5

调用 dll1.sendSILIntFloat(silInt) 方法发送这个整数数组。由于未指定浮点数组，不起实际作用。

发送浮点数据

```
silFloat = np.zeros(20).astype(float).tolist()
silFloat[0] = 5
dll1.sendSILIntFloat(silInt,silFloat)
```

使用 numpy 创建一个长度为 20 的浮点数组 silFloat，所有元素初始值为 0。

将 silFloat 数组的第一个元素设置为 5。

调用 dll1.sendSILIntFloat(silInt, silFloat) 方法发送整数数组和浮点数组。这次调用与前一次的区别在于同时指定了浮点数据。

1.InFaultAPITest\InFaultAPITest.py 关键代码如下

导入必需的库

```
import time
```

```
import math
import numpy as np
import PX4MavCtrlV4 as PX4MavCtrl
from DllSimCtrlAPI import DllSimCtrlAPI
```

time：用于处理时间相关的功能，例如延时。

numpy：用于数值计算，创建和操作数值数组。

PX4MavCtrlV4 和 DllSimCtrlAPI：RflySim 通信 API，其中 PX4MavCtrlV4 是用来控制无人机，而 DllSimCtrlAPI 用于进行模拟数据的注入和其他控制任务。

初始化无人机控制和模拟接口

```
dll1 = DllSimCtrlAPI()
mav1 = PX4MavCtrl.PX4MavCtrler(1)
```

创建 DllSimCtrlAPI 实例 dll1 用于 DLL 模型外部数据通信。

创建 PX4MavCtrl.PX4MavCtrler 实例 mav1，指定为飞机 1。

初始化 MAVLink 通信循环

```
mav1.InitMavLoop()
```

通过调用 InitMavLoop 方法初始化 MAVLink 数据接收循环。

```
mav1.InitTrueDataLoop()
```

调用 InitTrueDataLoop 方法用于初始化仿真真值传输。

```
mav1.initOffboard()
```

调用 initOffboard 准备无人机进行外部控制。

定义时间间隔和条件控制变量

```
lastTime = time.time()
startTime = time.time()
timeInterval = 1/30.0
```

设置一个定时循环，频率为 30Hz，用于定期执行代码块。

```
flag = 0
flagTime=startTime
```

使用 flag 变量作为状态标记来控制无人机的不同行为。

定时任务执行

```
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
```

在无限循环内，通过计算实现精确的定时任务，确保任务按照 30Hz 的频率执行。

```
if time.time() - startTime > 5 and flag == 0:
        flag = 1
        flagTime=time.time()
        mav1.SendMavArm(True) # Arm the drone
       mav1.SendPosNED(0,0,-10)
```

在达到 5 秒时，执行起飞前的准备工作，解锁无人机并发送起飞指令。

```
if time.time() - startTime > 20 and flag==1:
        silInt=np.zeros(8).astype(int).tolist()
        silFloat=np.zeros(20).astype(float).tolist()
        silInt[0]=1
        silFloat[0]=1
        dll1.sendSILIntFloat(silInt,silFloat)
        flag=2
```

在 20 秒后，执行故障注入，通过 sendSILIntFloat 发送特定的故障数据。

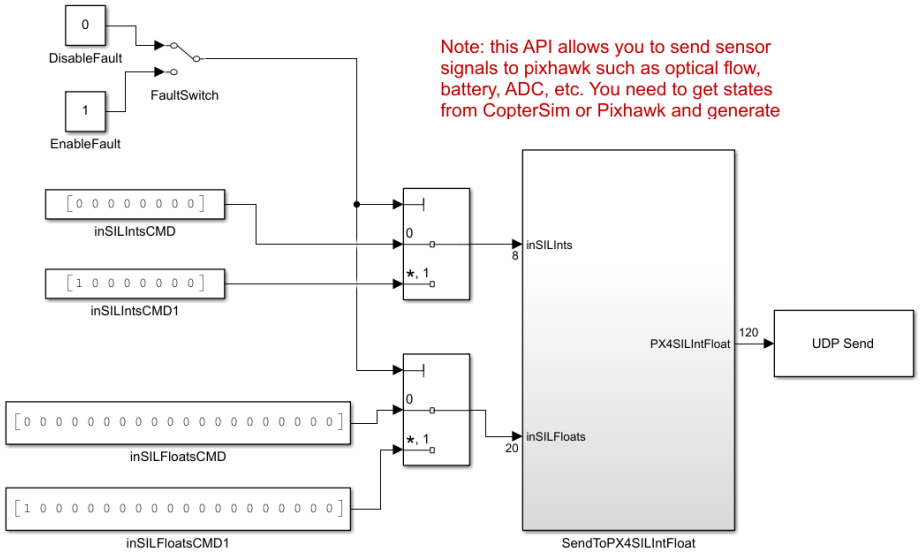
故障记录

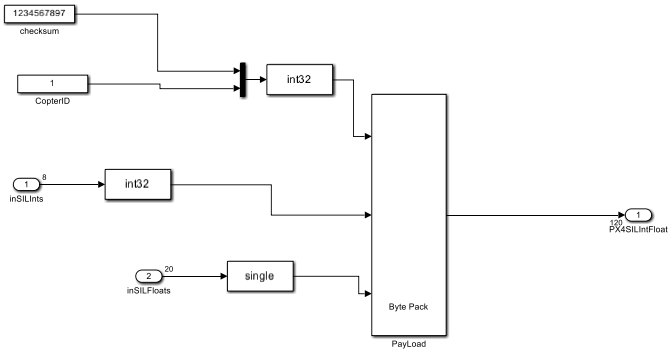
```
if flag==2:
        print(mav1.uavPosNED,mav1.truePosNED)
       if time.time() - startTime > 30:
           break
```

当 flag == 2（即故障已经被注入后）的条件下，代码持续打印无人机当前的位置和真实位置数据（uavPosNED 和 truePosNED）。当从开始计时到当前时间超过 30 秒时，跳出循环。这表明整个模拟的主要动作部分在 30 秒内完成。

3.3. Simulink 注入故障

与之前 inFaultAPITest 中通过故障注入脚本 InFaultAPITest.py 脚本注入电机故障的原理相同。这里通过 PX4ExtMsgSender.slx 中的 SendToPX4SILIntFloat 模块发送故障信息





这里的值会通过 UDP 30100 转发到 CopterSim 下的 DLL 模型的 inSILFloats 和 inSILInts 接口，打开 Exp2_MaxModelTemp.slx 可见，模型会通过以下两个函数作出判断：当 InsilInt 的第一位和 InsilFloat 的第一位大于 0.5 时触发电机故障

4. 相关文献

[1]. ..\..\..\6.RflySimExtCtrl\API.pdf

[2]. ..\..\..\7.RflySimPHM\API.pdf

[3]. ..\..\..\3.RflySim3DUE\API.pdf

[4]. 平台多旋翼模型控制仿真实验原理..\..\2.AdvExps\e2_MultiModelCtrl\Intro.pdf

[5]. ..\..\API.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