inDoubCtrls外部数据输入接口实验

3.文件目录

例程目录：
[安装目录]\RflySimAPIs\4.RflySimModel\0.ApiExps\11.inSILAPI\2.inDoubCtrls

文件夹/文件名称	说明
1.sendSILIntDouble\Readme.pdf 2.sendInDoubCtrls\Readme.pdf	Python调用inDoubCtrls接口实验步骤

总体说明

大场景仿真综合模型的需求

在原有动力学模型(被控对象)的基础上实现控制器从而实现更明显的控制闭环。

接口数据解析

结构体定义如下：

struct DllInDoubCtrls{

int checksum;//校验码1234567897

int CopterID; // 飞机的ID

double inDoubCtrls[28];//28维的double型输入

};

关键功能的实现

sendSILIntDouble发送

发送接口函数sendSILIntDouble

- 详细解析见[4]中的python控制接口 DllSimCtrlAPI

接口使用示例解析

1.sendSILIntDouble\sendSILIntDouble.py 关键代码如下：

导入必需的库和仿真API

import numpy as np

import time

import decimal

from DllSimCtrlAPI import DllSimCtrlAPI

numpy：用于创建和操作数值数组，常用于科学计算。

time：用于处理时间相关的任务。

decimal：提供高精度的十进制数运算。

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模型的多个函数。

初始化DLL控制接口

dll1 = DllSimCtrlAPI()

创建 DllSimCtrlAPI 类的一个实例，命名为 dll1。

初始化发送到dll的数据

insilInt = np.zeros(8).astype(int).tolist()

insilInt = np.arange(0,8,1)

inSILDoubs = np.zeros(20)

inSILDoubs[0] = decimal.Decimal('3.14159265358979323846')

inSILDoubs[1] = decimal.Decimal('4.14159265358979323846')

inSILDoubs[2] = decimal.Decimal('5.14159265358979323846')

insilInt初始化为从0到7的整数列表，用于某种类型的标识。

inSILDoubs是一个包含20个零值的浮点数组，前三个值被设置为高精度的十进制值。

30Hz循环执行发送任务

while True:

lastTime = lastTime + timeInterval

sleepTime = lastTime - time.time()

if sleepTime > 0:

time.sleep(sleepTime) # sleep until the desired clock

else:

lastTime = time.time()

dll.sendSILIntDouble(insilInt, inSILDoubs)

根据timeInterval调整lastTime。

计算sleepTime，即距离下一个周期开始的剩余时间。

如果sleepTime为正，暂停执行直到下一个周期。

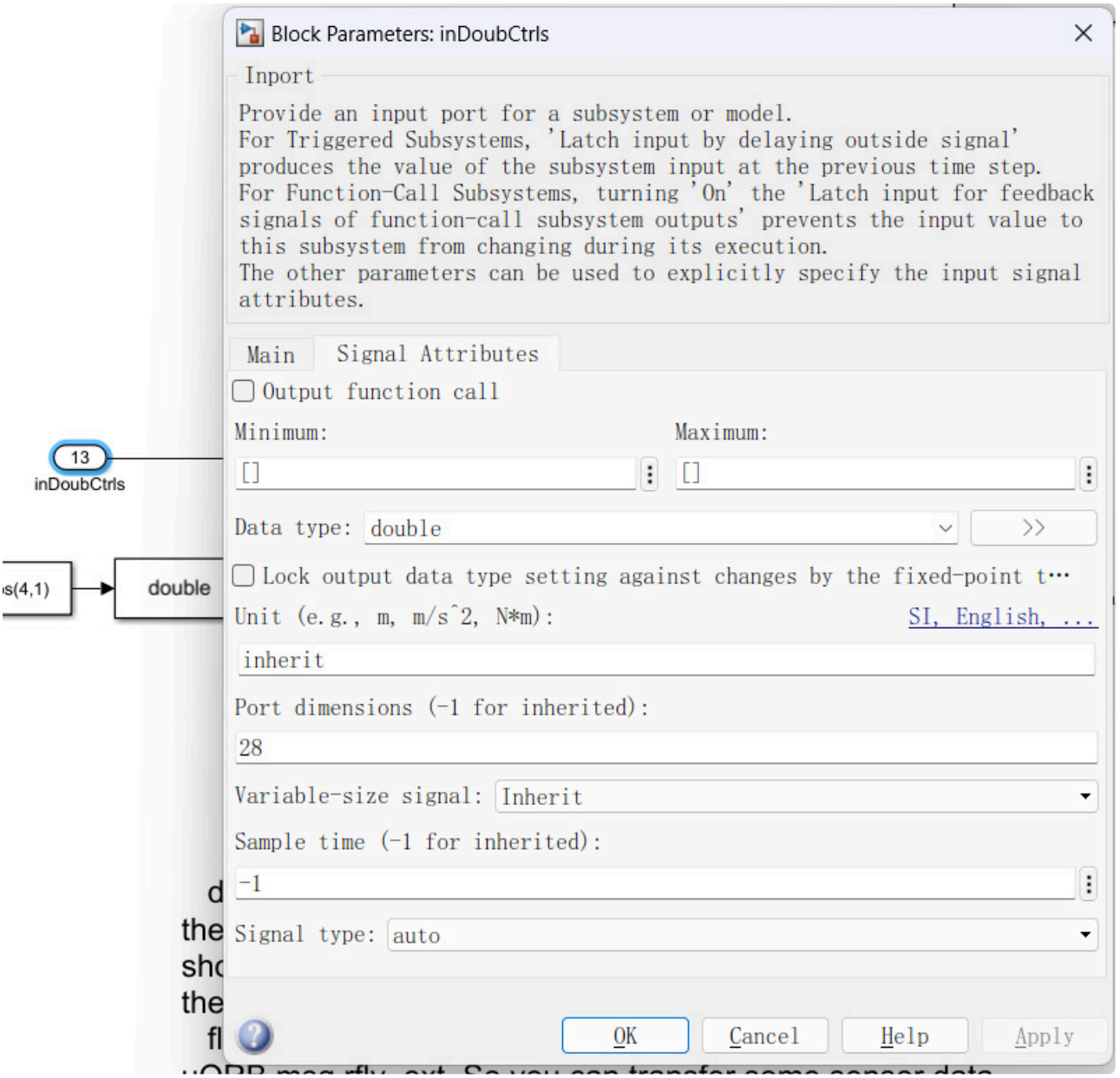
如果sleepTime不为正，说明执行时间已超出预定周期，更新lastTime为当前时间。

每个周期（30Hz，即每秒30次）调用dll.sendSILIntDouble函数发送整数列表和浮点数组到仿真系统。

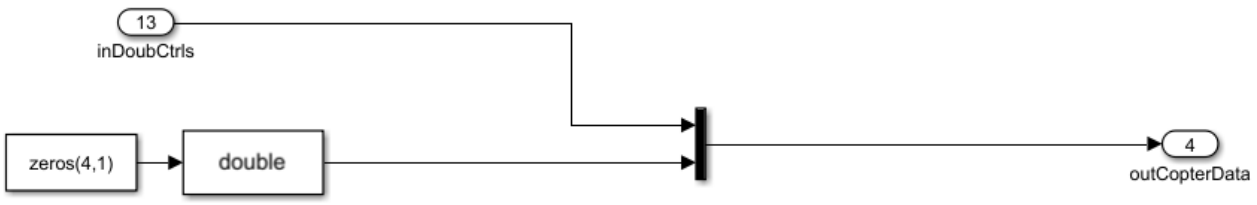
Simulink监听UDP30101端口

- 详细Simulink代码生成配置参考[2]
- **详细outCopterData接口介绍参考[4]**
- **动力学模型的外部UDP通信端口介绍参考[3]**

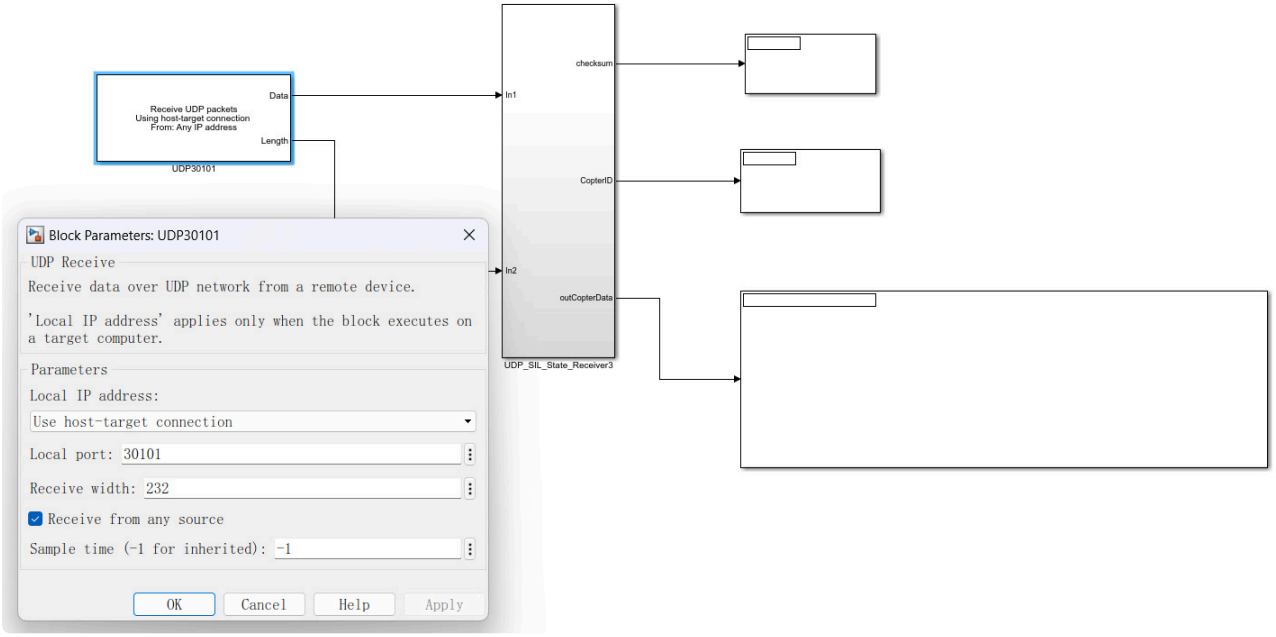
如果想使用DLL模型inDoubCtrls接口，那么在Simulink模型文件中新建1个输入端口inport，将名称修改为inDoubCtrls，数据类型设置为double，数据维度设置为28。保存编译并重新生成DLL文件后即可使用该接口。



在Simulink模型中将输入到inDoubCtrls的数据转发到outCopterData接口。
outCopterData接口会将数据通过UDP
30101系列端口转发出去。



通过监听UDP30100端口，按协议解析即可获取到输入到DLL模型的inDoubCtrls数据。



sendInDoubCtrls发送

发送接口函数sendInDoubCtrls

- 详细解析见[4]中的python控制接口 DllSimCtrlAPI

接口使用示例解析

2.sendInDoubCtrls\sendInDoubCtrls.py 关键代码如下：

导入必需的库和仿真API

import numpy as np

import time

import decimal

from DllSimCtrlAPI import DllSimCtrlAPI

numpy：用于创建和操作数值数组，常用于科学计算。

time：用于处理时间相关的任务。

decimal：提供高精度的十进制数运算。

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模型的多个函数。

初始化DLL控制接口

dll1 = DllSimCtrlAPI()

创建 DllSimCtrlAPI 类的一个实例，命名为 dll1。

初始化发送到dll的数据

inSILDoubs=np.zeros(28)

inSILDoubs[0]=decimal.Decimal('3.14159265358979323846')

inSILDoubs[1]=decimal.Decimal('4.14159265358979323846')

inSILDoubs[2]=decimal.Decimal('5.14159265358979323846')

inSILDoubs是一个包含28个零值的浮点数组，前三个值被设置为高精度的十进制值。

30Hz循环执行发送任务

while True:

lastTime = lastTime + timeInterval

sleepTime = lastTime - time.time()

if sleepTime > 0:

time.sleep(sleepTime)

else:

lastTime = time.time()

dll.sendInDoubCtrls(inSILDoubs)

根据timeInterval调整lastTime。

计算sleepTime，即距离下一个周期开始的剩余时间。

如果sleepTime为正，暂停执行直到下一个周期。

如果sleepTime不为正，说明执行时间已超出预定周期，更新lastTime为当前时间。

每个周期（30Hz，即每秒30次）调用dll.
sendInDoubCtrls函数发送双精度浮点数组到仿真系统。

Simulink监听UDP30101端口

- 详细Simulink代码生成配置参考[2]
- **详细outCopterData接口介绍参考[4]**
- **动力学模型的外部UDP通信端口介绍参考[3]**

同上3.1.3

相关文献

1. 第四讲API ..\..\API.pdf
2. ..\2.UserDefinedC++\Readme.pdf
3. ..\3.ExtCtrlAPI\Readme.pdf
4. ..\8.OutCopterData\Readme.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