题目 4：带有阻力的单摆运动（此题 100 分）

提示：此题难度较大，建议最后时间充足时，再进行尝试

一. 题目概述

提供一个单摆运动的数据集，存储在 pendulum.csv 的文件中，包含两个变量：

timestamp：时间，单位：秒(s)，单位为标准单位，在解题过程中不用考虑单位换算

theta：摆角，带有正负，表示方向，单位为弧度，即使用 sin 函数运算时不用考虑单位换算

请基于以上的数据集，使用 PyTorch 找到带有阻力的单摆运动的有关参数，补齐残缺数据并进行预测。

二. 提交要求

请基于数据集完成以下 5 道小题，按照要求把答案复制到答题卡的指定位置，并提交整个作答过程中的 Notebook 文档（会影响评分）。一共需要提交以下两个文件，请按照命名规则命名。

1. jupyter notebook 运行文件（包括运行结果的运行文件），命名规则为 NOAI4_学生编号.ipynb

（1）不提交 jupyter notebook 文件的，此题计 0 分。

（2）在作答过程中如使用了多个 jupyter notebook，请提供清晰的命名，例如用来做哪道题的。

2. NOAI4 答题卡的.docx 文档，命名规则为 NOAI4_学生编号.docx

三. 参考用库

1. 可以将下表用库直接复制到 notebook 中

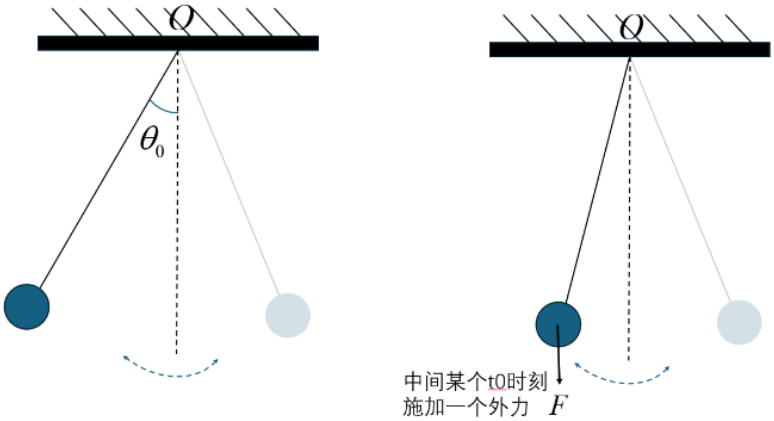
```
import numpy as np
import torch
from torch import nn
from scipy.integrate import odeint
import pandas as pd
import matplotlib.pyplot as plt
import math
```

2. 在整个答题过程中，为了避免卡在代码细节，你可以参考组委会提供的 pandas、numpy、matplotlib、PyTorch 参考代码，以及小爱同学求解带有阻力的直线运动的求解代码“附件 5_带阻力直线运动（数据预测）”，可以根据自己需要进行复制和修改。

四. 题目主体

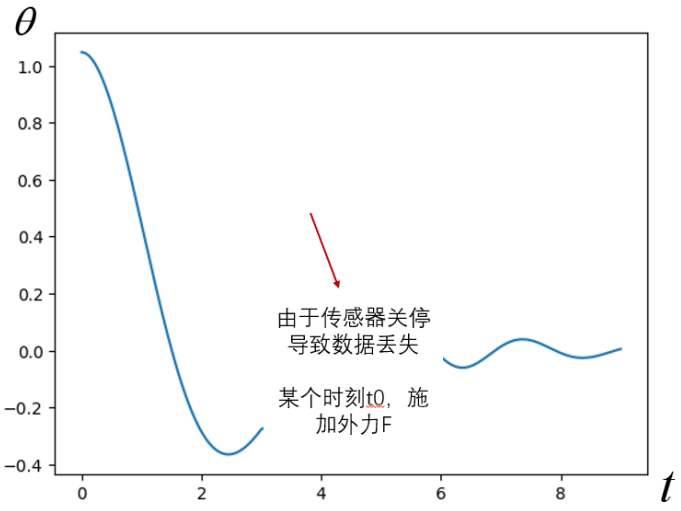
请阅读以下材料，求解 5 道小题，每道 20 分。

材料：在机器学习中我们有时会遇到需要借助少量数据完成对数据中的规律进行提取的场景。在这种场景下，如何充分利用先验知识（公式）处理数据、设计模型并成功求解未知参数是我们解决问题的关键所在。下面考虑一个单摆场景。



如上图所示，现有一可以视为质点的质量为 1（$m=1$）的小球通过一个长度为l的轻绳悬挂于一个固定点O，设$\theta(t)$ 为t时刻绳与过O铅垂线的夹角，称作摆角，将绳绷直并以初始的摆角$\theta_0 < \pi/2$（弧度制，不懂弧度制的请查阅附录 1）在$t=0$时刻无初速释放小球，则小球可以在轻绳和铅垂线对应的平面进行运动。这里以小球的初始摆角θ_0为正，并在铅垂线左侧时，摆角$\theta(t)$的符号记为正，在铅垂线右侧时，摆角$\theta(t)$的符号记为负。设重力加速度为$g=9.8$（整个题目求解过程中可以不用考虑单位换算，直接对数值进行运算即可），考虑大小与速度成正比、方向与速度相反的阻力，其中阻力大小μ在运动中保持恒定。

现有一传感器能够实现对小球摆角的精确记录，但是在实验过程中传感器出现了问题，它在记录时某时刻突然中断了 3s 后才被重启，且在小球没有停止运动时就已经停止工作。**已知在中断的时间内的某一时刻t_0开始小球受到了竖直向下大小恒定为F的外力，之后一直持续到运动结束**。请尝试通过已经记录的数据对小球的情况进行推断和预测。数据在文件 pendulum.csv 中，文件包括两列，第一列记录了小球运动的时间戳t，用变量 timestamp 表示，第二列记录了小球的摆角信息θ，用变量 theta 表示。我们将数据中$\theta(t)$绘制如下：



为了求解这个问题，你需要有微分方程的先验知识，在整个运动过程中。角速度 $\omega(t)$ 指的是 $\theta(t)$ 的瞬时变化率（有符号），角加速度 $a(t)$ 指的是角速度 $\omega(t)$ 的瞬时变化率（有符号），即 $\omega(t)=\frac{\mathrm{d}\theta(t)}{\mathrm{d}t}$，$a(t)=\frac{\mathrm{d}\omega(t)}{\mathrm{d}t}$（不懂导数的可以查阅附录 2）。则上述单摆问题中，$a(t)$ 与 $\omega(t)$ 和 $\theta(t)$ 之间应该满足如下微分方程：

$a(t)=-\alpha\cdot\omega(t)-\beta\sin\left(\theta(t)\right)$，其中 α,β 为参数，根据牛顿第二定律，$\alpha=\frac{\mu}{m}$，

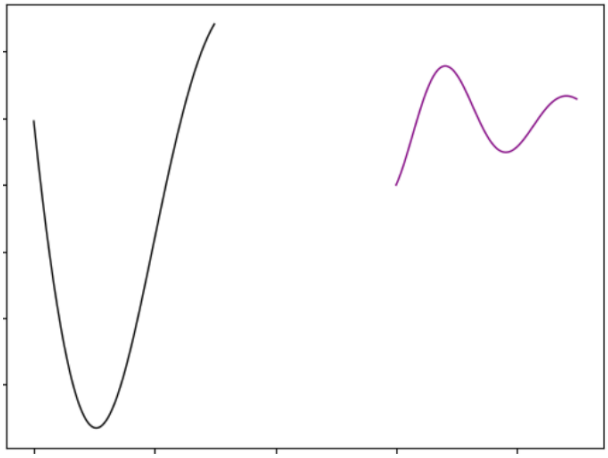
当 $0\le t<t_0$ 时，没有施加外力，此时，$\beta=\beta_1=\frac{g}{l}$；

当 $t\ge t_0$ 时，施加了外力 F，此时，$\beta=\beta_2=\frac{g}{l}+\frac{F}{ml}$.

利用微分方程，如果 α,β_1,β_2 已知，当知道 t 时刻的角速度 $\omega(t)$ 和摆角 $\theta(t)$，就可以根据微分方程求出来 $a(t)$。但是本题的问题是 α,β_1,β_2 未知，只有记录在每个时刻的摆角数据 $\theta(t)$，你需要根据提供的数据，**“回归”**出来 α,β_1,β_2，这是你在本题的核心任务，在得到 α,β_1,β_2 后，你就相当于求得了整个的微分方程，就可以利用微分方程对小球的摆角数据进行补全和预测。

1. 绘制数据中角速度 $\omega(t)$ 的变化图（20 分）

请直接根据图中数据，绘制角速度 $\omega(t)$ 的变化图（缺失部分不用补全直接留空即可），在传感器工作的前半段，角速度 $\omega(t)$ 数据用蓝线绘制，后半段为红线绘制。一个绘制样例效果如下图所示（注意：样例图中线的位置和颜色不具有参考价值）



提示：在 Python 中一般不会直接求解导数，而是使用差分代替，例如，

$$\omega(t_1)=\left.\frac{\mathrm{d}\theta(t)}{\mathrm{d}t}\right|_{t=t_1}=\frac{\theta(t_2)-\theta(t_1)}{t_2-t_1},$$

其中 t_1,t_2 为数据中前后两个相邻时刻，$\theta(t_1),\theta(t_2)$ 为前后两个相邻时刻的摆角。

请将绘制好的角速度 $\omega(t)$ 的变化图，复制到答题卡的相应区域。

2. 若在t_0时刻没有施加外力F，此时，预测第 6s 时小球的摆角$\theta(6)$。（20 分）

要完成此问，你首先需要通过数据“回归”出来α,β_1，确定施加外力前的微分方程。但是你可能从来没有学过如何使用 PyTorch 进行参数“回归”。为此，组委会在**附录 3** 中提供了一个小爱同学使用 PyTorch 求解带有阻力的直线运动的微分方程的参数并进行预测的案例，请阅读后帮助求解此问题，其中代码也可以直接复制使用。

你在改编代码的过程中可能会在 Python 中用到三角函数的计算，计算方法如下：

（1）在 python 中可以使用 math 库计算三角函数值

math.sin(theta)计算 theta 的正弦值，其中 theta 是一个数，要求输入为弧度制；

math.cos(theta)计算 theta 的余弦值；其中 theta 是一个数，要求输入为弧度制；

math.tan(theta)计算 theta 的正切值；其中 theta 是一个数，要求输入为弧度制；

（2）如果在搭建神经网络时，由于 torch 的运算对象均为 torch.tensor，所以需要使用 torch 内自带的三角函数计算方法：

torch.sin(tensor_theta)计算 theta 的正弦值，其中 theta 是一个 torch.tensor，tensor 中的数均要求弧度制；

torch.cos(tensor_theta))计算 theta 的余弦值；其中 theta 是一个 torch.tensor，tensor 中的数均要求弧度制；

torch.tan(tensor_theta))计算 theta 的正切值；其中 theta 是一个 torch.tensor，tensor 中的数均要求弧度制；

注：本题需要使用传感器在前半段（传感器坏之前）的所有数据，建立模型求解微分方程的参数，仅使用 3 个点手动求解参数的计 0 分。

请将求解参数的模型 class 和你预测的第 6s 时的小球摆角$\theta(6)$填写到答题卡的相应位置，其中$\theta(6)$可以直接打印出来结果并复制到答题卡上。

3. 若t_0时刻施加了外力F，预测实验中小球在传感器记录结束后（数据中的最后一个时刻之后），下一次摆角为零（$\theta(t)\approx 0$）的时刻。（20 分）

注：本题需要使用传感器记录的所有数据，建立模型求解微分方程的参数，仅使用 3 个点手动求解参数的计 0 分。

请将求解参数的模型 class 和你预测的下一次摆角为零（$\theta(t)\approx 0$）的时刻填写到答题卡的相应位置。

如果第 2、3 题的 class 写在了一起，则可以复制一样的，不影响评分。

4. 计算绳子的长度l，阻力μ和施加的外力F的值。（20 分）

（1）绳子长度$l=$_____.

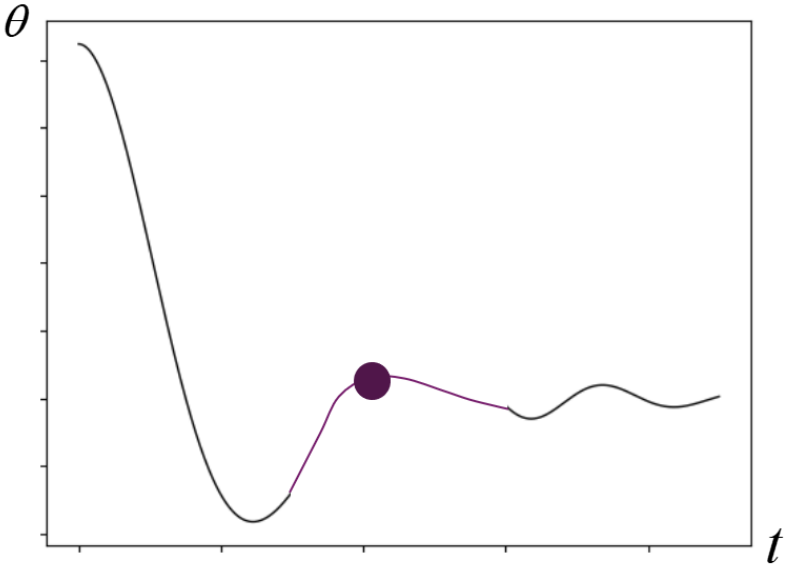
（2）阻力$\mu=$_____.

（3）施加的外力$F=$_____.

请将以上结果填写到答题卡的相应位置。

5. 使用上一问得到的微分方程补全观测数据并进行可视化，并用实心点图中标记 F 施加的时刻。（20 分）

要求原始数据用蓝线绘制，中间补上的数据用红线绘制，F 施加时刻的实心点用红点标注，实心点的大小不小于 20。一个绘制样例效果如下图所示（注意：样图与题目要求的颜色、线型和位置都不具有参考价值）



提示：可以直接用 ax.scatter([x], [y], s=20, color='red')标注一个红色实心点，其中 x 和 y 分别为点的横纵坐标。

请将你绘制好的图片复制到答题卡的相应区域。

附录 1：什么是弧度制（高一数学内容）

弧度制是一种测量角度的单位制度，它定义了一个完整圆周为 2π 弧度。在弧度制中，角度的度量是基于圆的半径，而不是像度数制那样基于圆周的分割。弧度制在数学和物理学中非常重要，因为它与圆的几何属性和三角函数的自然定义紧密相关。

以下是弧度制和度数制之间的一些基本转换关系：

1. 一个完整的圆周等于 360 度，也等于 2π 弧度；
2. 1 度等于 $\frac{\pi}{180}$ 弧度；
3. 1 弧度等于 $\frac{180}{\pi}$ 度。

因此，如果你有一个角度的度数，你可以使用以下公式将其转换为弧度：

$$\text{弧度} = \text{角度} \cdot \frac{\pi}{180};$$

例如，将 90 度转换为弧度：

$$90^\circ \cdot \frac{\pi}{180} = \frac{\pi}{2} \text{弧度};$$

相反，如果你有一个角度的弧度，你可以使用以下公式将其转换为度数：

$$\text{角度} = \text{弧度} \cdot \frac{180}{\pi};$$

例如，将 $\frac{\pi}{2}$ 转换为角度：

$$\frac{\pi}{2} \times \frac{180}{\pi} = 90^\circ.$$

附录 2：什么是导数（高二数学内容）

导数是微积分学中的一个基本概念，它描述了一个函数在某一点上的变化率。更具体地说，导数衡量的是当输入值（通常表示为 x ）发生微小变化时，函数输出值（通常表示为 $f(x)$ ）的变化量。

形式上，如果函数 $f(x)$ 在点 x 处可导，那么 $f(x)$ 在 x 处的导数记为 $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$，$\frac{\mathrm{d}f(x)}{\mathrm{d}x}$ 的值是函数在该点的切线斜率。如果用极限的语言来描述，导数可以定义为：

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

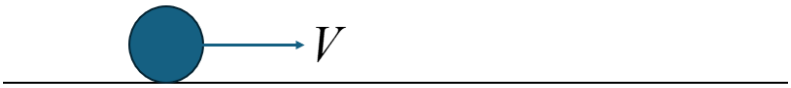
在本题中：

$$\omega(t) = \frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = \lim_{h \to 0} \frac{\theta(t+h) - \theta(t)}{h},$$

$$a(t) = \frac{\mathrm{d}\omega(t)}{\mathrm{d}t} = \lim_{h \to 0} \frac{\omega(t+h) - \omega(t)}{h}.$$

附录 3 小爱同学使用 PyTorch 求解带有阻力的微分方程的案例

现在有一个可以看作质点的小球进行带有阻力的直线运动。



其中位移 $s(t)$，速度为 $v(t)=\frac{\mathrm{d}s(t)}{\mathrm{d}t}$，加速度为 $a(t)=\frac{\mathrm{d}v(t)}{\mathrm{d}t}$。小球受到两个阻力，一个阻力和当前速度成正比，另一个阻力和当前的位移成正比。即小球的运动满足如下微分方程：

$$a(t)=-\alpha v(t)-\beta s(t),$$

其中 α 是与速度成正比的阻力参数，其中 β 是与位移成正比的阻力参数，

小爱同学将该运动过程的数据记录在了 sv_data.csv 中，包括两列数据：
t：时间，单位：s，即为标准单位，不用考虑单位换算
s：位移，单位，m，即为标准单位，不用考虑单位换算

小爱同学使用 PyTorch 撰写了代码进行“回归”，求解了其中的 α,β，确定了微分方程，并预测了再过 5 秒之后的小球的位移和速度。他的求解过程的代码如下，你也可以直接打开附件 5 查看整个代码和运行结果。如果对 odeint 求解微分方程在某一时刻的位置的使用方法感兴趣，可以参考附件 6。

```
import numpy as np
import torch
from torch import nn
from scipy.integrate import odeint
import pandas as pd
import matplotlib.pyplot as plt
import math

###定义求导/差分运算###
def derivative(t, s):  #使用差分代替导数，可以用记录的位移值代替计算速度
    dsdt = (s[1:] - s[:-1]) / (t[1:] - t[:-1])  #这就意味着每次使用差分求一次导，数组的大小就要-1，s 记录 n 个数据，则 v 只能记录 n-1 个数据
    return dsdt

def load_data(csv_path):  #读取位移和时间数据
    data = pd.read_csv(csv_path)
    t = np.array(data["t"])  #读取时间
    s = np.array(data["s"])  #读取位移
    return t, s

###读取数据###
csv_path = "sv_data.csv"
t, s = load_data(csv_path)  #读取数据
v = derivative(t, s) #计算各个时刻的速度
#print(v)
```

```
###定义求解（回归出）微分方程参数的网络###
class StraightLine(nn.Module):  #使用 Pytorch 搭建微分方程模型
    def __init__(self):
        super().__init__()
        self.alpha  =  nn.Parameter(torch.randn((1,),  requires_grad=True))   #需要记录下来参数，
requires_grad = True 可以一直不用改，为默认参数
        self.beta = nn.Parameter(torch.randn((1,), requires_grad=True))

    def forward(self, theta, omega, add_force=False):
        a = - self.alpha * omega - self.beta * theta  #微分方程
        return a

###位移 s，速度 v，加速度 a 的数组大小统一，s 需要删掉最后两个值，v 需要删掉最后一个值，a 不需要删###
#为了能够将数据导入到 PyTorch 中，需要将 np.array 和 list 转为 torch.tensor，代码如下#
a = torch.from_numpy(derivative(t[:-1], v))  #每次使用差分求一次导，数组的大小就要-1
#print(a)
_v = torch.from_numpy(v[:-1]) #每次使用差分求一次导，数组的大小就要-1,s 记录 n 个数据，则 v 只能记录
n-1 个数据,a 只能记录 n-2 个数据
_s = torch.from_numpy(s[:-2]) #每次使用差分求一次导，数组的大小就要-1,s 记录 n 个数据，则 v 只能记录
n-1 个数据,a 只能记录 n-2 个数据

###开始进行训练###
num_epoch = 20000  #训练 20000 轮就差不多了
model = StraightLine()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

###提取模型求解出来的参数并打印####
alpha = (model.alpha).detach().item() #打印模型参数，与 v 相关的阻力系数
beta = (model.beta).detach().item() #打印模型参数,与 s 相关的阻力系数
print("alpha:{}".format(alpha))
print("beta:{}".format(beta))

####使用 odient 库求解微分方程的方法
def v_func(sv, t, alpha, beta):  #定义一个微分方程，输入分别为 sv（一个数对，分别为位移 s 和速度 v；
t 为时间，直接这么写就行，alpha，beta 分别是与 v 有关的阻力系数和与 s 有关的阻力系数）
    s,v = sv
    a = - alpha * v - beta * s #微分方程
    return np.array([v,a])
sv_expect = odeint(v_func, [s[-1],v[-1]], [0,5], args=(alpha,beta))  #打印五秒后的位移和速度，其
中 s[-1]和 v[-1]提取最后一个数，表示提取当前位置的数，0 表示当前时刻，5 表示 5 秒后
print("再过 5 秒后的位移 s 为",sv_expect[1,0]) #第一列为位移，第一行为当前时刻的位移
print("再过 5 秒后的速度 v 为",sv_expect[1,1]) #第二列为速度，第二行为当前时刻的速度
```