Table of Contents

Physics Informed Neural Networks

Problem

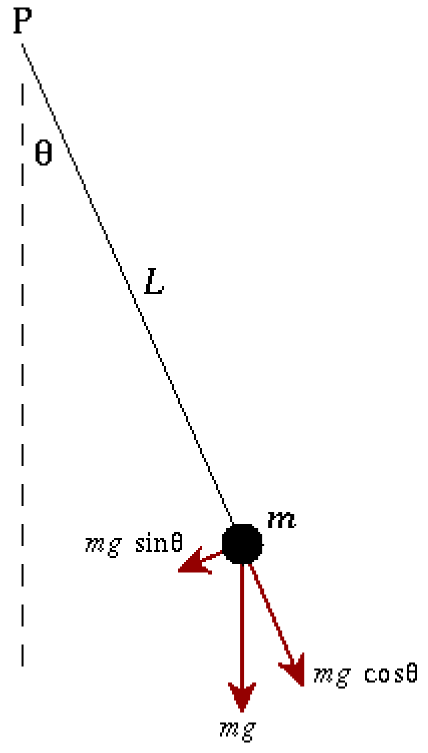
Let's say we have a swing and we want to model the acceleration at any position.

We measured the length of the rope, and **took 10 observations** at different angles from the top and noted the acceleration.

Physics

The classical physics tells us that an Ideal pendulum of length L satisfies the following diffential equation

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0$$



True Model

Let us assume the motion of the swing is goverened by a differential equation. Since this is not an ideal pendulum it is likely that it is not following above differential equation but maybe

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta - 0.1 * \theta = 0$$

where $0.1 * \theta$ is accounting for other factors. But we of corse are **not aware** of the equation.

> Can we predict the true acceleration just from those **10 observations**?

Simulating Observations

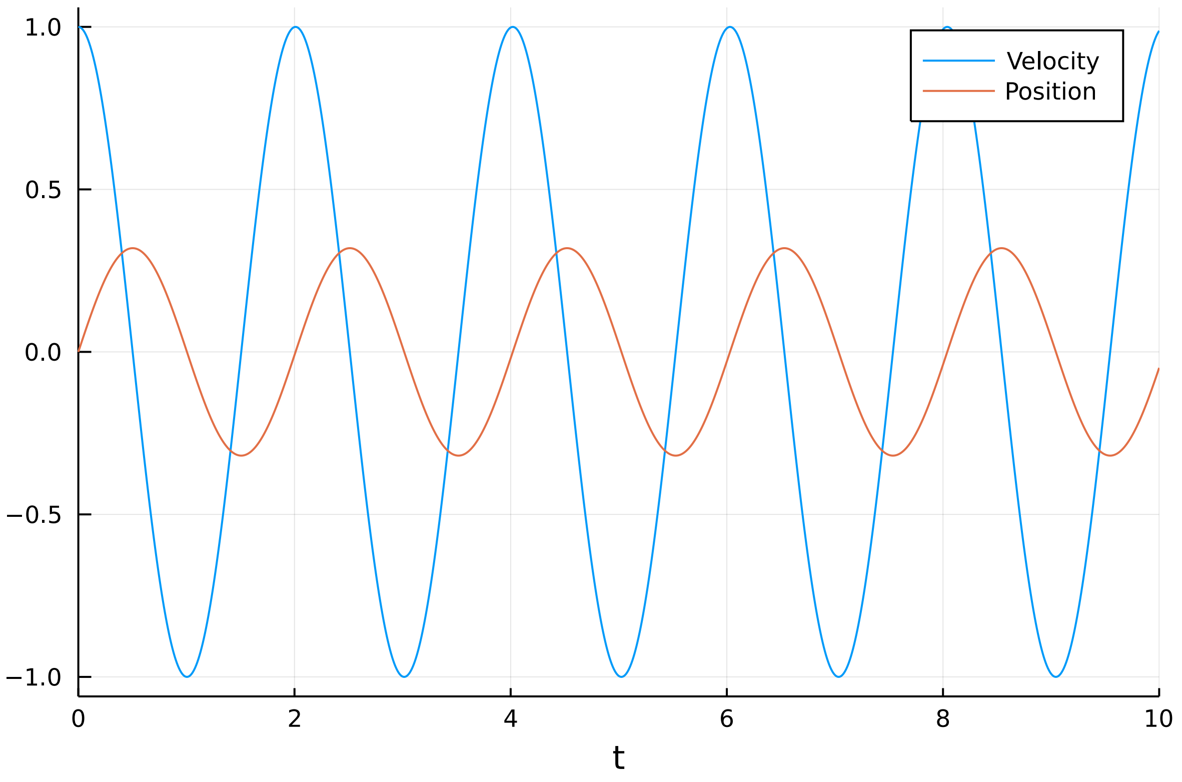
In order to simulate the observations, we will use the true model and solve it suing a numerical ODE solver and observe 10 points.

Let us assume that $g = 10m/s^2$ and $L = 1m$. Hence the equation is

$$\frac{d^2\theta}{dt^2} + 10\sin\theta - 0.1 * \theta = 0$$

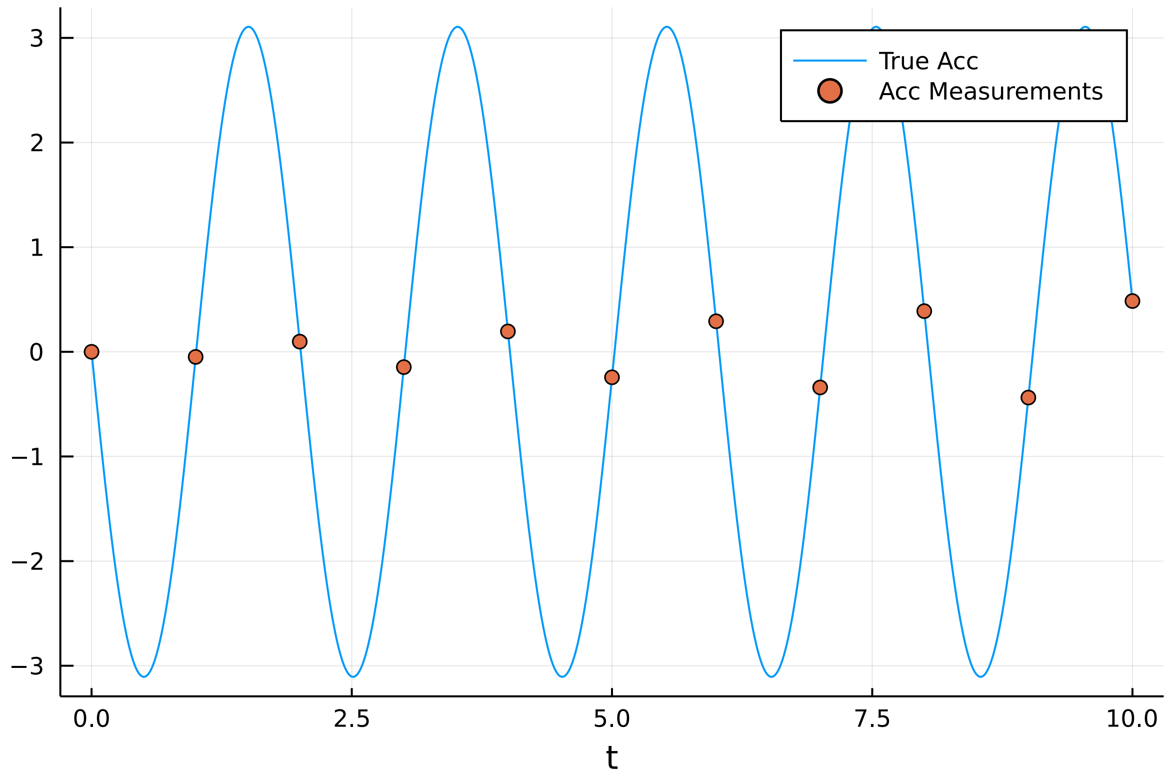
Let us assume that $\theta'(0) = 0$ and $\theta(0) = 0$

Let's use a ODE solver to find the solution.



```
begin
    using DifferentialEquations, Plots
    g = 10
    acc(dθ,θ,g,t) = -g*sin(θ) + 0.1*θ
    prob = SecondOrderODEProblem(acc,1.0,0.0,(0.0,10.0),g)
    sol = solve(prob)
    plot(sol,label=["Velocity" "Position"])
end
```

Observations



```
begin
    plot_t = 0:0.01:10
    data_plot = sol(plot_t)
    positions_plot = [state[2] for state in data_plot]
    acc_plot = [acc(state[1],state[2],g,plot_t) for state in data_plot]

    t = 0:1:10
    dataset = sol(t)
    position_data = [state[2] for state in sol(t)]
    acc_data = [acc(state[1],state[2],g,t) for state in sol(t)]

    plot(plot_t,acc_plot,xlabel="t",label="True Acc")
    scatter!(t,acc_data,label="Acc Measurements")
end
```

```
Float64[0.0, -0.0487458, 0.0974773, -0.146183, 0.194848, -0.243461, 0.292009, -0.340
```

```
acc_data
```

Neural Network

Let us train a neural network just on the basis of the observations.

```
Chain(#1, Dense(1, 32, tanh), Dense(32, 1), first)
```

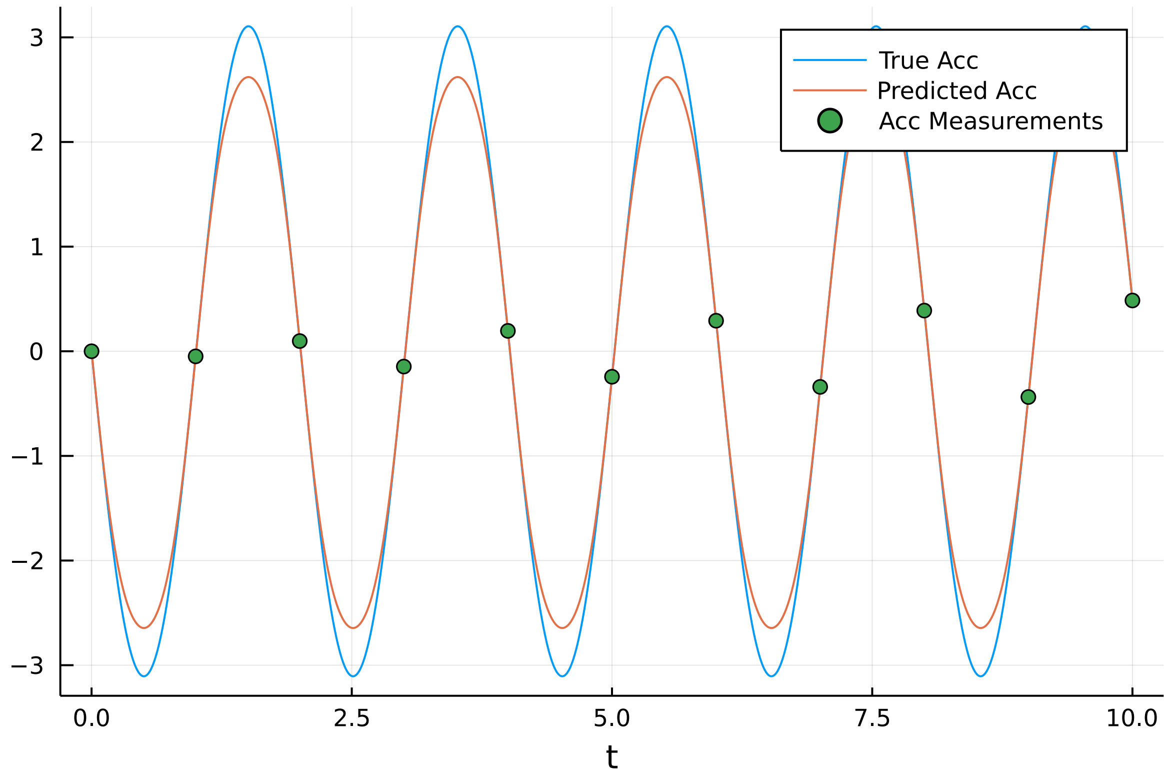
```
begin
    using Flux
    NN = Chain(x -> [x],
               Dense(1,32,tanh),
               Dense(32,1),
               first)
```

```
        end
```

```
0.8441568145498834
```

```
begin
    loss() = sum(abs2,NN(position_data[i]) - acc_data[i] for i in
1:length(position_data))
    loss()
end
```

```
begin
    display(loss())
    Flux.train!(loss, Flux.params(NN),Iterators.repeated((), 1000), ADAM(0.1))
end
```



```
begin
    learned_acc_plot = NN.(positions_plot)
    plot(plot_t,acc_plot,xlabel="t",label="True Acc")
    plot!(plot_t,learned_acc_plot,label="Predicted Acc")
    scatter!(t,acc_data,label="Acc Measurements")
end
```

We can see that our neural network approximated the supposed acceleration but we are not quite using all he information we have about the system.

Adding prior physical knowledge

We might not know the actual differential equation governing the swing but we still know that this system must follow the pendulum equation pattern. So we can encode that **knowledge** in the loss function.

```
PINN = Chain(#9, Dense(1, 32, tanh), Dense(32, 1), first)
```

```
PINN = Chain(x -> [x],
             Dense(1,32,tanh),
             Dense(32,1),
             first)

```

```
2540.084468712278
```

```
begin
    random_positions = [2rand()-1 for i in 1:100] # random values in [-1,1]
    loss_ode() = sum(abs2,PINN(x) - (-g*sin(x)) for x in random_positions)
    loss_ode()
end
```

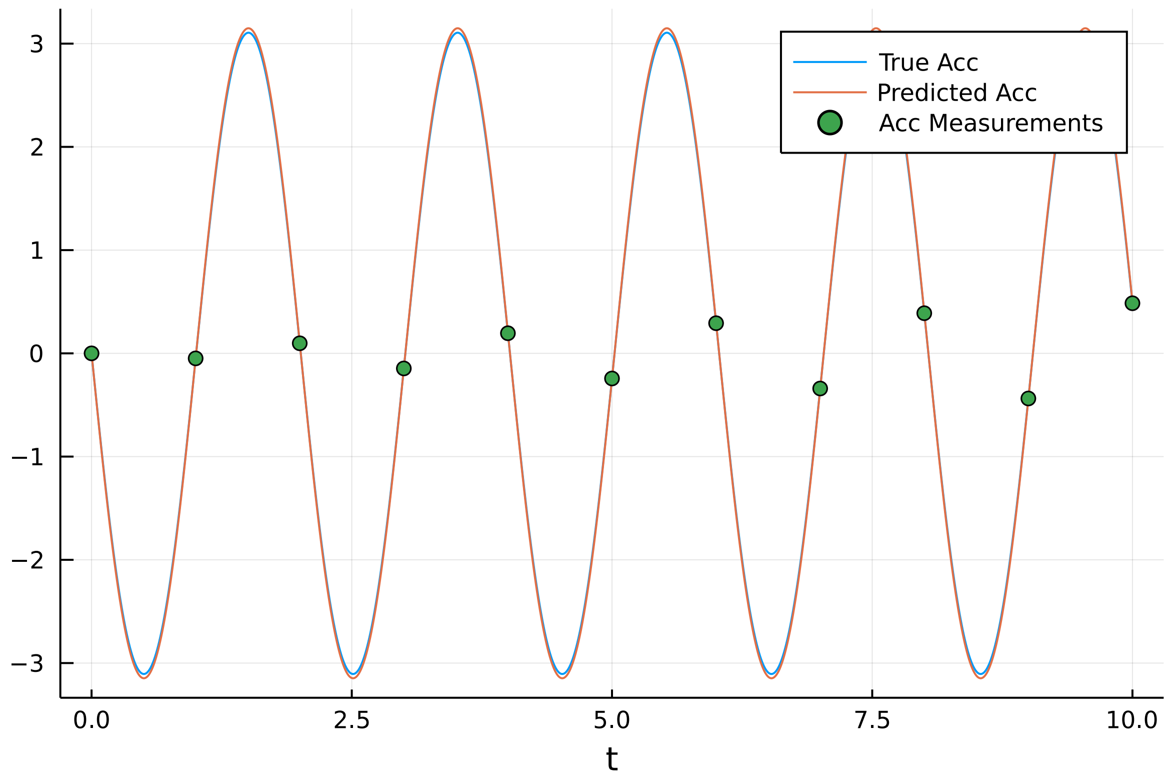
```
lossP (generic function with 1 method)
```

```
    lossP() = sum(abs2,NN(position_data[i]) - acc_data[i] for i in
1:length(position_data))
```

```
composed_loss (generic function with 1 method)
```

```
begin
    λ = 0.1
    composed_loss() = lossP() + λ*loss_ode()
end
```

```
begin
    Flux.train!(composed_loss, Flux.params(PINN),Iterators.repeated((), 1000),
ADAM(0.1))
end
```



```
begin
    Physics_learned_acc_plot = PINN.(positions_plot)

    plot(plot_t,acc_plot,xlabel="t",label="True Acc")
    plot!(plot_t,Physics_learned_acc_plot,label="Predicted Acc")
    scatter!(t,acc_data,label="Acc Measurements")
end
```

We can see that this is a quite accurate approximation of the supposed equation.