EECS 367 & ROB 320 Lab
Pendularm support

Administrative

- Assignment #1: Path Planning
  - Due Friday, February 4
- Quiz #2: Next Monday, January 24th
  - Through gradescope, available 12:00am-11:59pm
  - Time limit of 30 minutes
  - Covers material from assignment #1
  - Don’t discuss quiz with other students; honor code

Administrative

- Pendularm Setpoint Competition!
  - Goal: Who's controller can reach the most set points in 60 seconds?
  - Next Monday: Code release to visualize and test set point control
  - Next Wednesday: During interactive session (only), students can collaborate on PID tuning to optimize their solutions

Lab Takeaways

1. Stencil review
2. Equation translation
3. Implementation advice
4. Big picture

Pendularm Overview

	Assignment 2: Pendularm
4	Euler integrator
4	Velocity Verlet integrator
4	PID control

Stencil Review

update_pendulum_state.js

```
1  function update_pendulum_state(numerical_integrator, pendulum, dt, gravity) {
2      // integrate pendulum state forward in time by dt
3      // please use names 'pendulum.angle', 'pendulum.angle_previous', etc. in else codeblock between line 28-30
4
5      if (typeof numerical_integrator === "undefined")
6          numerical_integrator = "none";
7
8      if (numerical_integrator === "euler") {
9
10     // STENCIL: a correct Euler integrator is REQUIRED for assignment
11
12     }
13     else if (numerical_integrator === "verlet") {
14
15     // STENCIL: basic Verlet integration
16
17     }
18     else if (numerical_integrator === "velocity verlet") {
19
20     // STENCIL: a correct velocity Verlet integrator is REQUIRED for assignment
21
22     }
23     else if (numerical_integrator === "runge-kutta") {
24
25     // STENCIL: Runge-Kutta 4 integrator
26     }
27     else {
28         pendulum.angle_previous = pendulum.angle;
29         pendulum.angle = (pendulum.angle+Math.PI/180)%(2*Math.PI);
30         pendulum.angle_dot = (pendulum.angle-pendulum.angle_previous)/dt;
31         numerical_integrator = "none";
32     }
33
34     return pendulum;
35 }
```

Feature stencils

Default rotation

Acceleration Helper Function

update_pendulum_state.js

```
37  function pendulum_acceleration(pendulum, gravity) {
38      // STENCIL: return acceleration(s) system equation(s) of motion
39
40  }
41
```

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta) + \frac{\tau}{ml^2}$$

Euler Equations

update_pendulum_state.js

```
8       if (numerical_integrator === "euler") {
9
10      // STENCIL: a                                          for assignment
11
12      }
13      else if (numerical_integrator === "verlet") {
14
15      // STENCIL: basic Verlet integration
16
17      }
18      else if (numerical_integrator === "velocity verlet") {
19
20      // STENCIL: a correct velocity Verlet integrator is REQUIRED for assignment
21
22      }
23      else if (numerical_integrator === "runge-kutta") {
24
```

$$\theta(t+dt) = \theta(t) + \dot{\theta}(t)dt$$
$$\dot{\theta}(t+dt) = \dot{\theta}(t) + \ddot{\theta}(t)dt$$

pendulum_acceleration

Verlet Equations

update_pendulum_state.js

```
8       if (numerical_integrator === "euler") {
9
10      // STENCIL: a                                          for assignment
11
12      }
13      else if (numerical_integrator === "verlet") {
14
15      // STE
16
17      }
18      else if (numerical_integrator === "velocity verlet") {
19
20      // STENCIL: a correct velocity Verlet integrator is REQUIRED for assignment
21
22      }
23      else if (numerical_integrator === "runge-kutta") {
24
```

$$\theta(t+dt) = \theta(t) + \dot{\theta}(t)dt$$
$$\dot{\theta}(t+dt) = \dot{\theta}(t) + \ddot{\theta}(t)dt$$

$$\theta(t+dt) = 2\theta(t) - \theta(t-dt) + \ddot{\theta}(t)dt^2$$

Don’t forget to initialize in `init_verlet_integrator`!

`pendulum_acceleration`

Velocity Verlet Equations

update_pendulum_state.js

```
8       if (numerical_integrator === "euler") {
9
10      // STENCIL: a                                                    for assignment
11
12      }
13      else if (numerical_integrator === "verlet") {
14
15      // STE
16
17      }
18      else if (numerical_integrator === "velocity verlet") {
19
20      // STEN                                                          or assignment
21
22      }
23      else if
24
```

$$\theta(t+dt) = \theta(t) + \dot{\theta}(t)dt$$
$$\dot{\theta}(t+dt) = \dot{\theta}(t) + \ddot{\theta}(t)dt$$

$$\theta(t+dt) = 2\theta(t) - \theta(t-dt) + \ddot{\theta}(t)dt^2$$

$$\theta(t+dt) = \theta(t) + \dot{\theta}(t)dt + \frac{1}{2}\ddot{\theta}(t)dt^2$$

$$\dot{\theta}(t+dt) = \dot{\theta}(t) + \frac{\ddot{\theta}(t) + \ddot{\theta}(t+dt)}{2}dt$$

PID Control

update_pendulum_state.js

```
49  function set_PID_parameters(pendulum) {
50      // STENCIL: change pid parameters
51      pendulum.servo = {kp:0, kd:0, ki:0}; // no control
52      return pendulum;
53  }
54
55  function PID(pendulum, accumulated_error, dt) {
56      //
58
59      re
60  }
```

Tune PID parameters here

pendulum.servo.error

pendulum.control

_error

accumulated_error

$$e(t) = \theta_{desired}(t) - \theta(t)$$

$$\tau(t) = K_p e(t) + K_i \int_0^t e(\alpha)d\alpha + K_d \frac{d}{dt} e(t)$$

$$\int_0^t e(\alpha)d\alpha = \sum_{\alpha=0}^{t} e(\alpha) = e(t) + \sum_{\alpha=0}^{t-1} e(\alpha)$$

Lab Takeaways

1. Stencil review
2. Equation translation
3. Implementation advice
4. Big picture

Implementation Advice

Pay attention to time index within equations!

- Previous time step: $t - dt$
- Current time step: t
- Future time step: $t + dt$

Parameterized helper functions can help reduce code duplication

- We've done this for you with `pendulum_acceleration(pendulum, gravity)`

Unnecessary global variables can be difficult to debug!

Motivation of Assignment

Understand how *error* can be used as feedback in a closed-loop fashion to control a system

- **Practice:** Implement a PID servo controller for a pendulum functioning under well defined rules (classical/Newtonian mechanics)

Motivation of Assignment

Why are we working with Newton's equations of motion and numerical integrators?

The real world operates under Newtonian mechanics, why not implement the pendulum and PID controller there?

- Too expensive and too slow!

Instead, implement pendulum in simulation

- Use Newton's equations of motion to model the change of pendulum's position over time (model of the state dynamics)
- With numerical integration techniques and an initial position, we can approximate the pendulum's position at any specific time

Motivation of Assignment

Understand how *error* can be used as feedback in a closed-loop fashion to control a system

- **Practice:** Implement a PID servo controller for a pendulum functioning under well defined rules (classical/Newtonian mechanics)

Understand why simulations are relevant in robotics and how they can be implemented using differential equations

- **Practice:** Build a digital simulation of the pendulum operating under the laws of classical mechanics