$\sqrt{Gabe}$
GabeL@virginia.edu
https://gabriel-vzh2vs.github.io/SYS3062-Website/
UVA's Systems and Information Engineering

March 10th, 2026

SYS 3062: Lab 6

Hybrid Simulations: ODEs and Discrete Events

Before the Thermostat: SimPy Basics

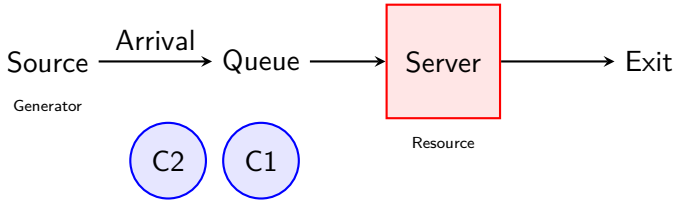
Lab 5 uses SimPy to schedule events. To understand the syntax, let's look at the simplest possible simulation: a line at a coffee shop (M/M/1 Queue).

The Three Core Concepts:

1. **Environment ('env'):** The clock. It manages simulation time.
2. **Process ('env.process'):** A Python generator (function) that defines behavior (e.g., a Customer arriving).
3. **Event ('yield'):** The magic word. It pauses the function until an event finishes (e.g., 'env.timeout(5)' means "sleep for 5 minutes").

GabeL@virginia.edu

Visualizing the M/M/1 Queue



GabeL@virginia.edu

The Code: Defining Behavior

In SimPy, we write a function that describes the *lifecycle* of one agent.

```
def customer(env, name, server):
    print(f"{name} arrives at {env.now}")

    # Request the Server (Resource)
    # 'with' automatically handles the Queue!
    with server.request() as req:
        yield req  # Wait until server is free

        print(f"{name} starts service at {env.now}")

        # Simulate Service Time (e.g., 5 minutes)
        # This 'pauses' this function for 5 ticks
        yield env.timeout(5)

        print(f"{name} leaves at {env.now}")
```

GabeL@virginia.edu

How this applies to the Thermostat

In Lab 6, we use the same mechanism, but different logic:

- **Queue Example:** `yield env.timeout(5)` → "Service takes 5 mins."
- **Lab 6 Thermostat:** `yield env.timeout(1/60)` → "Step physics forward 1 min."

Key Takeaway: `yield` is just a "Pause Button."

- In Queues, we pause for Activity (Service).
- In Lab 5, we pause for Time Steps (Integration).

GabeL@virginia.edu

Lab 6: Hybrid System Simulation

The "Hybrid" Challenge

Real-world systems often involve two distinct types of behavior interacting simultaneously:

1. **Continuous Dynamics (Physics):** Variables change smoothly over time (e.g., Temperature, Pressure, Velocity).
2. **Discrete Events (States):** The system "jumps" between states instantly (e.g., Heater turns ON, Valve Closes).

Today's Subject: Thermostatic Control We will simulate a room where:

- The Temperature evolves according to differential equations (ODEs).
- The Heater switches On/Off based on thresholds.

GabeL@virginia.edu

ODEs in Simulation: Discretizing Reality

The Continuous Reality

In the real world, temperature changes smoothly and continuously.

$$\frac{dT}{dt} = \beta(\dot{Q}_{gain} - \dot{Q}_{loss})$$

The Computational Problem: Computers cannot handle "continuous" time. They operate in steps. We must discretize the equation.

Euler's Method (Numerical Integration)

We approximate the curve by taking small linear steps (Δt):

$$T_{new} \approx T_{old} + \left(\frac{dT}{dt}\right)_{old} \times \Delta t$$

GabeL@virginia.edu

Implementing Euler's Method

In our code, we implement this integration explicitly inside the simulation loop.

The Code Mapping:

- **Rate Calculation (dT/dt):**

```
dtr = delta_tr_rate(house.Tr, house.Te, ...)
```

- **Euler Integration Step:**

```
# T_new      = T_old     + (Rate * TimeStep)
house.Tr += dtr * DT
```

Critical Note: If your Time Step (DT) is too large (e.g., 1 hour), the error accumulates, and the simulation becomes unstable. We use $DT = 1/60$ (1 minute) for accuracy.

The room temperature (T_r) is governed by the balance of Heat Loss and Heat Gain.

Governing Equation

$$\frac{dT_r}{dt} = \beta(\dot{Q}_{\text{gain}} - \dot{Q}_{\text{loss}})$$

1. Heat Loss (Always Active):

$$\dot{Q}_{\text{loss}} = \frac{T_r - T_{env}}{\eta R}$$

Heat escapes to the environment (T_{env}) through walls (R).

2. Heat Gain (Conditional):

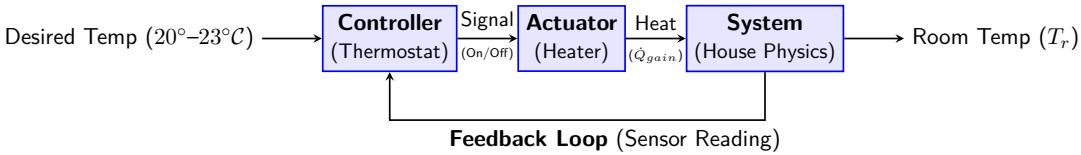
$$\dot{Q}_{\text{gain}} = \alpha(T_h - T_r)$$

Heat enters from the heater (T_h), ONLY if the Heater is ON.

GabeL@virginia.edu

Control Theory: The Feedback Loop

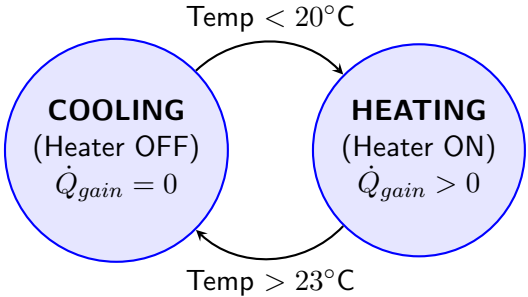
This lab models a classic **Closed-Loop Control System**.



Key Concept: The Controller observes the System's output (T_r) to decide the next action, correcting errors dynamically.

GabeL@virginia.edu

The system switches states based on a Hysteresis loop to prevent rapid flickering.



GabeL@virginia.edu

Why do we have two thresholds (20° and 23°) instead of just one target (e.g., 21.5°)?

The Problem: Chattering

If we had a single threshold, the heater would switch ON/OFF thousands of times per second as the temperature hovered exactly at the target. This destroys equipment.

The Solution: Hysteresis (Dead Band)

We create a gap where no state change occurs.

- **State A (Heating):** Stay ON until we hit the Top (23°).
- **State B (Cooling):** Stay OFF until we hit the Bottom (20°).

This memory in the system prevents rapid oscillation and models real-world hardware behavior.

GabeL@virginia.edu

Algorithm Part 1: Logic & Setup

Algorithm 1 Hybrid Loop (Part I: Discrete Control)

$T_r \leftarrow 20$ ▷ Initial Room Temp
$State \leftarrow$ OFF ▷ Initial Heater State
$dt \leftarrow 0.01$
for $t = 0$ **to** 24 **step** dt **do**
    Update $T_{env}(t)$ ▷ Stochastic/Sinusoidal
    // Discrete Event Check (Hysteresis)
    if $T_r < 20$ **then**
        $State \leftarrow$ ON
    else if $T_r > 23$ **then**
        $State \leftarrow$ OFF
    end if
    ... Continue to Physics Update (Next Slide)
end for

GabeL@virginia.edu

Algorithm Part 2: Physics Update

Algorithm 2 Hybrid Loop (Part II: Continuous Physics)

for $t = 0$ **to** 24 **step** dt **do**
    // Continuous Dynamics (ODEs)
    $\dot{Q}_{loss} \leftarrow (T_r - T_{env})/(\eta \times R)$
    if $State ==$ ON **then**
        $\dot{Q}_{gain} \leftarrow \alpha \times (T_h - T_r)$
    else
        $\dot{Q}_{gain} \leftarrow 0$
    end if
    // Euler Integration Step
    $dT \leftarrow \beta \times (\dot{Q}_{gain} - \dot{Q}_{loss})$
    $T_r \leftarrow T_r + (dT \times dt)$
end for

GabeL@virginia.edu

Task 2: Adding Human Behavior

The Scenario: Humans are unpredictable. Sometimes they open a window, ruining the thermal isolation.

Modeling the Event

This is a **Discrete Event** causing parameter changes.

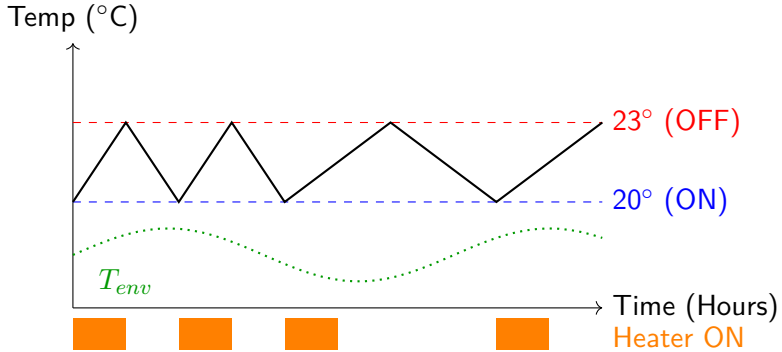
- **Normal State:** Insulation $\eta = 1.0$.
- **Event (Window Open):** Insulation η jumps to 5.0 (Efficiency degrades).

Expected Result: When the window opens, $\dot{Q}_{loss}$ spikes. The temperature drops rapidly, forcing the heater to turn ON and stay ON longer to compensate.

GabeL@virginia.edu

Expected Output: The Sawtooth Pattern

You should generate a plot showing the Room Temperature oscillating between the bounds, while the Environment changes in the background.



GabeL@virginia.edu

Now, it's your turn to implement a hybrid simulation in Python or using Excel! Good Luck, particularly if you are using Excel.

GabeL@virginia.edu

SimPy Strategy: The Physics Loop

```
def physics_and_thermostat_process(env, house):
    t1, t2 = 20, 23 # Thresholds

    while True:
        # Calculate dTe (External) and dTr (Room)
        dtr = delta_tr_rate(house.Tr, house.Te,
        house.heating, house.eta)
        house.Tr += dtr * DT

        if house.Tr >= t2 and house.heating:
            house.heating = False
        elif house.Tr <= t1 and not house.heating:
            house.heating = True

        yield env.timeout(DT) # DT = 1/60 (1 Minute)
```

GabeL@virginia.edu

SimPy Strategy: Discrete Events (People)

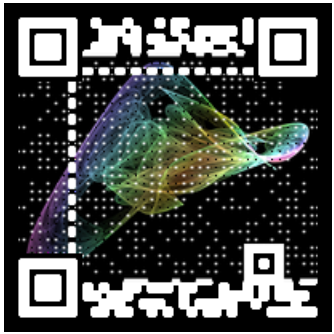
```python
def people_process(env, house, person_id):
    # Sleep until morning (Random wait)
    wake_delay = 6 + random.normalvariate(0, 0.5)
    yield env.timeout(wake_delay)

    while env.now < 22: # Until bedtime
        # EVENT: Open Window (Degrade Insulation)
        house.eta = random.random() # Efficiency drops (< 1.0)

        open_duration = 0.1 * abs(random.normalvariate(1, 0.3))
        yield env.timeout(open_duration)

        house.eta = 1.0 # Efficiency restored
        yield env.timeout(random.random())
```


GabeL@virginia.edu

Feedback Form

Throughout the course, I may ask for your feedback on:

- **Course Materials**
- **Lecture & Lab**
- **Assignments**



GabeL@virginia.edu

Action Items

First, I will thank you all again for coming to lab at 5 pm on a Tuesday!

And now, you need to do the following tasks:

- **Submit** your Python file to Gradescope by next week's lab!
- **Read** Prelab 7 for next week!

Next week we are doing output analysis!

GabeL@virginia.edu