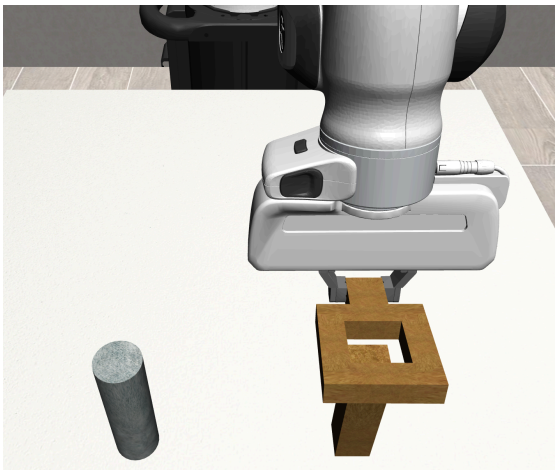
COM SCI 188: Introduction to Robotics (spring 2025)

Coding Assignment 3: Imitation with Dynamic Movement Primitives

Deadline: 5/23 11:59pm

Task: NutAssemblySquare



You are given `demonstration_data.npz` containing one demonstration of this task. The goal is to reproduce the motion of this trajectory with dynamic movement primitives.

You will implement two Python modules—first, the core DMP class for trajectory learning and generation; and second, a high-level policy that stitches DMP rollouts together with simple PID control.

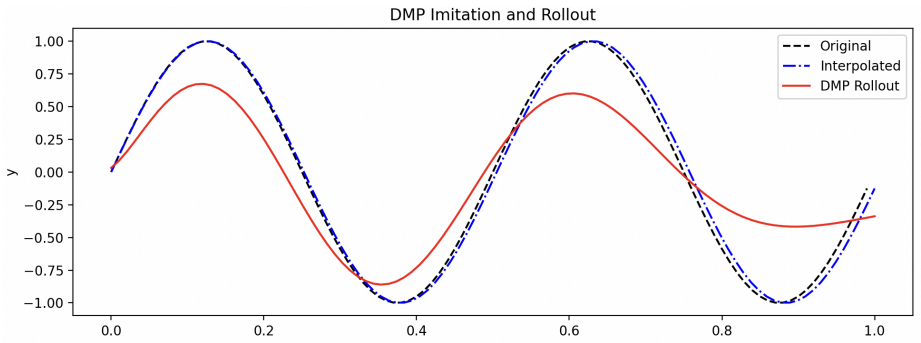
Follow the steps below, and use the provided tests to verify your work.

1. Implement the DMP core (`dmp.py`)

- **File:** `dmp.py`
- **Class to fill in:** `CanonicalSystem` and `DMP`
- **Required functionality:**
  - **CanonicalSystem**
    Initialize with `dt` and `ax`.
    Implement `reset()`, `step(tau, error_coupling)` and `rollout(tau, ec)` so that the phase variable *x* decays from 1 → 0.

  - **DMP**
    Constructor sets up `n_dmps`, `n_bfs`, `dt`, `y0`, `goal`, gains `ay`/`by`, weight matrix `w`, plus its `CanonicalSystem`.
    `reset_state()` to zero out `y, dy, ddy` and reset the canonical system.
    `_psi(x)` to compute Gaussian basis activations.
    `imitate(y_des)` to interpolate a demonstration, compute target forcing (`f_target`), and solve for weights `w`.
    `rollout(tau, error, new_goal)` to generate a new trajectory (with optional goal override).

- **Quick test:**
  Run `dmp.py`, you should see:



2. Implement the segmented DMP+PID policy (`dmp_policy.py`)

- **File:** `dmp_policy.py`
- **Class to fill in:** `DMPPolicyWithPID` (you may also add helper functions)
- **Required functionality:**
    1. **Trajectory segmentation**
        - Detect “grasp on/off” segments (e.g. when the grasp flag toggles between –1 and 1).

    2. **DMP fitting per segment**
        - For each segment, fit a 3-D position DMP on the end-effector trajectory.
        - Re-target the **first** segment’s endpoint to a new object pose, while replaying subsequent segments exactly.

    3. **PID tracking**
        - Associate a simple PID controller with each segment that tracks the DMP position rollout.
          The policy’s `get_action(obs)` should return a 7-D action:
          `[Δx, Δy, Δz, 0, 0, 0, grasp_flag]`
        - where the first three entries come from PID, rotation is fixed at zero, and the last entry is the segment’s grasp signal.

- **Test:** Run **`test_ca3.py`** (do not modify this file).

Submission Checklist

- `dmp.py` with **fully working** `CanonicalSystem` and `DMP` classes, passing the built-in main-block plot test.
- `dmp_policy.py` with **fully working** `DMPPolicyWithPID`, passing `test_ca3.py` with a success rate >0.8