Discrete-Time Robot Control: Design, Simulation & Comparison

A comprehensive Python project for designing, implementing, simulating, and comparing control strategies for three discrete-time robot models.

Models

1. **Integrator Model** - 2D double integrator with additive disturbances
2. **Unicycle Model** - Nonholonomic mobile robot with heading dynamics
3. **Two-Link Manipulator** - Planar robot arm with Euler discretization

Controllers Implemented

Linear Controllers

- Proportional (P) Control
- PID Control
- LQR (Linear Quadratic Regulator)
- State-feedback with Pole Placement

Nonlinear Controllers

- Feedback Linearization
- Backstepping Control
- Sliding Mode Control (SMC)
- Computed Torque (for manipulator)

Optimal/Constrained

- Model Predictive Control (MPC)

Data-Driven

- Policy Gradient RL baseline

Features

- **Lyapunov Stability Analysis**: Symbolic and numerical verification
- **Symbolic Control**: Reachability, avoidance, recurrence via grid abstraction and LTL/Buchi automata
- **Interactive Drawing Canvas**: Click to draw polygonal obstacles and goal regions
- **Animation Export**: Export trajectories as GIF or MP4
- **Automated Testing**: pytest suite with 26 tests
- **Comprehensive Reports**: Automatic markdown reports with numerical metrics and figures

Interactive Canvas

Launch the interactive drawing canvas with:

```
python interactive_canvas.py
```

Features:

- **Draw obstacles**: Click to add vertices, right-click to finish polygon
- **Draw goal region**: Switch to Goal mode, draw polygon
- **Set start position**: Switch to Start mode, click to place
- **Select controller**: Choose from LQR, Proportional, MPC, or Reach-Avoid
- **Run simulation**: Click Run to see trajectory
- **Export animation**: Save as GIF

Keyboard shortcuts: `o`=obstacle, `g`=goal, `s`=start, `c`=clear, `r`=run

Installation

```
pip install -r requirements.txt
```

Quick Start

```
# Run smoke test for integrator model
python run_smoke_test.py

# Run all tests
pytest tests/

# Run full comparison with all models
python run_all.py

# Generate comprehensive comparison report with numerical metrics
python generate_report.py

# Launch interactive drawing canvas
python interactive_canvas.py

# Run interactive demo with user input
python interactive_demo.py
```

Project Structure

```
├── models/               # Robot model definitions
│   ├── integrator.py     # 2D integrator
│   ├── unicycle.py       # Nonholonomic mobile robot
│   └── manipulator.py    # Two-link planar arm
├── controllers/          # Controller implementations
```

```
│   ├── proportional.py  # P control
│   ├── pid.py           # PID control
│   ├── lqr.py           # LQR control
│   ├── mpc.py           # Model Predictive Control
│   ├── rl_policy_gradient.py  # REINFORCE RL
│   ├── unicycle/        # Unicycle-specific controllers
│   └── manipulator/     # Manipulator-specific controllers
├── sim/                 # Simulation engine & plotting
│   ├── simulator.py     # Core simulation loop
│   ├── plotting.py      # Visualization utilities
│   └── animation.py     # GIF/MP4 export
├── symbolic/            # Symbolic control
│   ├── grid_abstraction.py  # Space discretization
│   ├── reach_avoid.py       # Reach-avoid planning
│   └── ltl_automata.py      # LTL/Buchi for patrolling
├── analysis/            # Stability analysis
│   └── lyapunov.py      # Lyapunov function computation
├── notebooks/           # Jupyter demos (3 notebooks)
├── tests/               # pytest test suite (26 tests)
├── report/              # Generated reports & figures
├── interactive_canvas.py    # Draw obstacles/goals interactively
├── interactive_demo.py      # CLI demo with user input
├── generate_report.py       # Generate comparison report
├── run_all.py              # Run full test suite
├── requirements.txt
├── USAGE_GUIDE.md          # Detailed usage documentation
└── README.md
```

Model Specifications

Based on discrete-time models from Symbolic_control_lecture-7.pdf.

Model 1: Integrator

- State: $x = [x1, x2]$
- Dynamics: $x(t+1) = x(t) + \tau(u(t) + w(t))$
- State constraints: $X = [-10,10] \times [-10,10]$
- Input constraints: $U = [-1,1] \times [-1,1]$
- Disturbance: $W = [-0.05,0.05] \times [-0.05,0.05]$

Model 2: Unicycle

- State: $x = [x1, x2, \theta]$
- Input constraints: $U = [0.25,1] \times [-1,1]$
- Disturbance: $W = [-0.05,0.05]^3$

Model 3: Two-Link Manipulator

- State: $x = [\theta 1, \theta 2, \dot{\theta} 1, \dot{\theta} 2]$
- Parameters: $m1=m2=1.0$ kg, $\ell 1=\ell 2=0.5$ m, $g=9.81$ m/s^2

License

MIT License