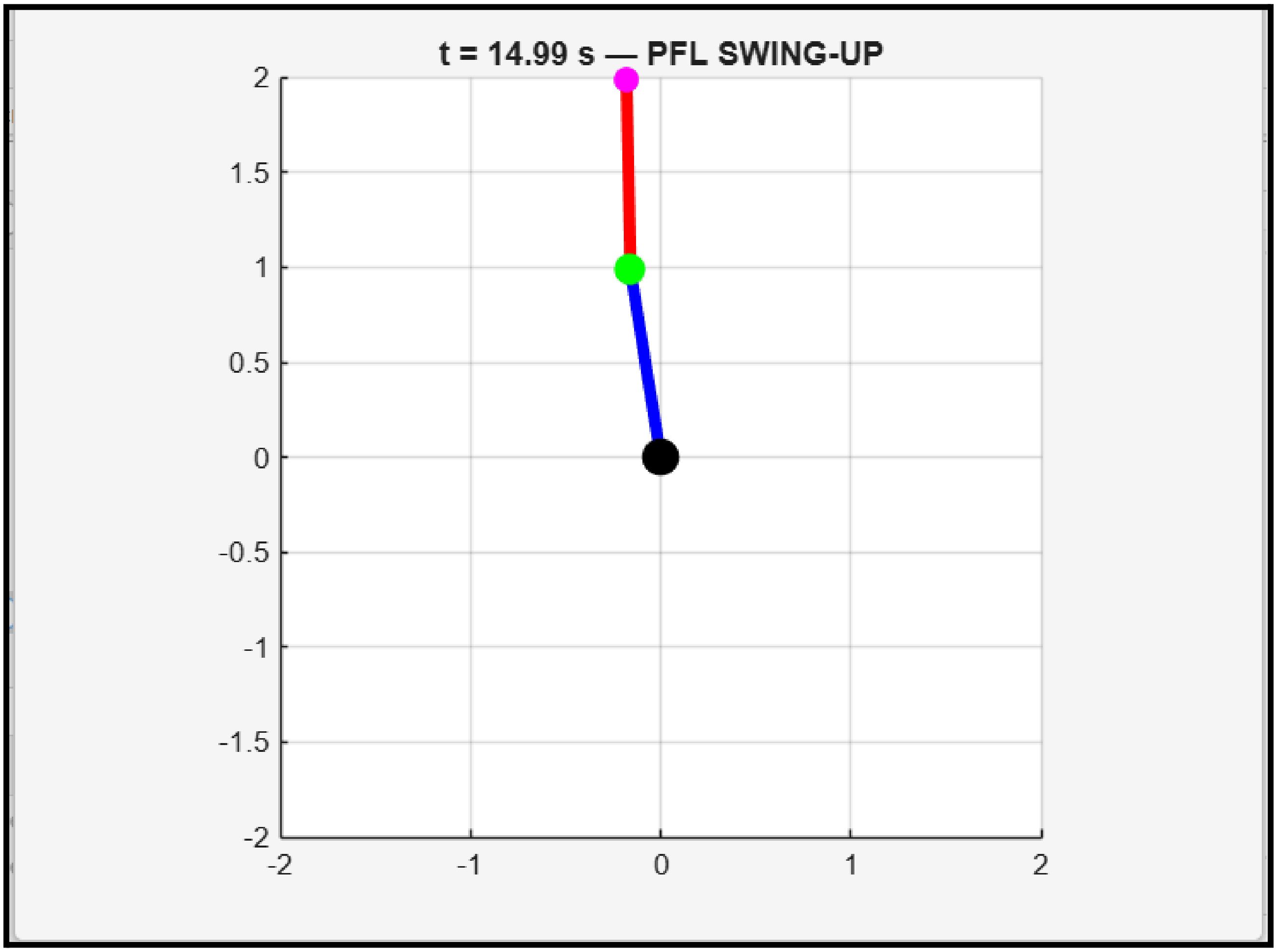
Partial Feedback Linearization

Acrobot



Jash Shah

1 Dec. 2025

ECE 238 - ADV CONTROL DES LAB

Link to Code: https://github.com/Jash-2000/Adv_Control_Systems/tree/main/Acrobot_PFL

INTRODUCTION

I implemented a two-stage controller for the 2-link Acrobot (unactuated shoulder, actuated elbow) following Spong (1995): an energy-shaping Partial Feedback Linearization (PFL) swing-up and a linear LQR stabilizer near the upright equilibrium. The code contains:

- symbolic Lagrangian derivation that produces `M(q), C(q,dq), G(q)` and saves them to a MAT file,
- a fast numerical `acrobotDynamics` using Lagrangian forms,
- `computeTotalEnergy` (T+V),
- `pflSwingUpController` for energy pumping,
- `linearizeAcrobot` to compute (A,B) about upright and `lqr` to compute K,
- closed-loop simulation with `ode45`, animation and diagnostics.

Goal: get a robust swing up and stable catch at upright using LQR.

Hypothesis

- PFL can shape the passive joint dynamics so that a virtual input `v` (based on energy error) increases total mechanical energy to the upright energy.
- When the state is close to upright and slow enough, a locally linearized LQR stabilizer about the upright will catch and hold the robot.
- With correct Lagrangian EOMs, correct energy reference, and proper switching/hysteresis/saturation, the scheme will succeed

Variables and System Definitions

- `params`: physical parameters (`m1,m2,l1,l2,lc1,lc2,I1,I2,g`).
- States `x = [theta1; theta2; dtheta1; dtheta2]` in **standard convention**: `theta1=0` is upright, positive CCW, hanging-down at `pi`.
- `x0 = [pi;0;0;0]`: start hanging down.
- `x_eq = [0;0;0;0]`: target upright.
- `computeTotalEnergy(x,params)`: returns `E = T + V` with `T = 0.5 dq' M dq` and `V` computed with vertical positions.
- `E_desired`: `E_desired = E_up - E_down`

- `linearizeAcrobot(x_eq,params)`: returns (A,B) for LQR.
- `K`: LQR gain from `lqr(A,B,Q,R)`.
- Controller parameters: `control.k_energy`, `control.u_max`, `theta_threshold`, `omega_threshold`.

The system is defined precisely using the physics in MATLAB as follows:

```
function dx = LagrangianEOMS(x, u, p)
% Lagrangian-based acrobot dynamics
%% Unpack states
q1 = x(1); q2 = x(2);
dq1 = x(3); dq2 = x(4);
q = [q1; q2]; dq = [dq1; dq2];
%% Unpack parameters
m1 = p.m1; m2 = p.m2;
l1 = p.l1; l2 = p.l2;
lc1 = p.lc1; lc2 = p.lc2;
I1 = p.I1; I2 = p.I2;
g  = p.g;
%% Define symbolic variables
syms s1 s2 ds1 ds2 dd1 dd2 real
qs  = [s1; s2]; dqs = [ds1; ds2]; ddqs = [dd1; dd2];
%% Kinematics
x1 = lc1*sin(s1);   y1 = lc1*cos(s1);
x2 = l1*sin(s1) + lc2*sin(s1 + s2);
y2 = l1*cos(s1) + lc2*cos(s1 + s2);
J1 = jacobian([x1; y1], qs);
J2 = jacobian([x2; y2], qs);
v1 = J1*dqs;
v2 = J2*dqs;
%% Energies
T = 1/2*m1*(v1.'*v1) + 1/2*I1*ds1^2 + ...
   1/2*m2*(v2.'*v2) + 1/2*I2*(ds1 + ds2)^2;
V = m1*g*y1 + m2*g*y2;
L = T - V;
%% Lagrange equations
dLd_dq = jacobian(L, dqs).';
d_dt_dLd_dq = jacobian(dLd_dq, qs)*dqs + jacobian(dLd_dq, dqs)*ddqs;
dLd_q = jacobian(L, qs).';
Q = [0; u]; % only joint 2 actuated
EOM = simplify(d_dt_dLd_dq - dLd_q - Q);
%% Solve for accelerations
[Msym, rhs] = equationsToMatrix(EOM, ddqs);
dd = double(subs(Msym \ rhs, [s1 s2 ds1 ds2], [q1 q2 dq1 dq2]));
%% Output state derivative
dx = [dq1; dq2; dd(1); dd(2)];
end
```

Results and Discussion

During the first few seconds:

- PFL adds energy to the system
- theta_1 oscillates, gaining amplitude
- theta_2 oscillates to transfer energy through the unactuated joint
- Energy E(t) converges to Edesired. This confirms that the energy-shaping controller is functioning properly.

Once the threshold is reached:

- the controller switches to LQR
- the state converges to the upright equilibrium
- the torque magnitude decreases
- oscillations damp out gracefully

I received the following results from my implementation:

```
Desired energy (upright):  -39.2400 J

A =
        0         0    1.0000         0
        0         0         0    1.0000
 -12.6129   12.6129         0         0
  16.8171  -46.2471         0         0

B =
        0
        0
  -4.2857
  13.7143

LQR gain K =
 -16.1556    3.5088    2.5147    5.6026
```

The following plot shows the controller dynamics after running through ODE45

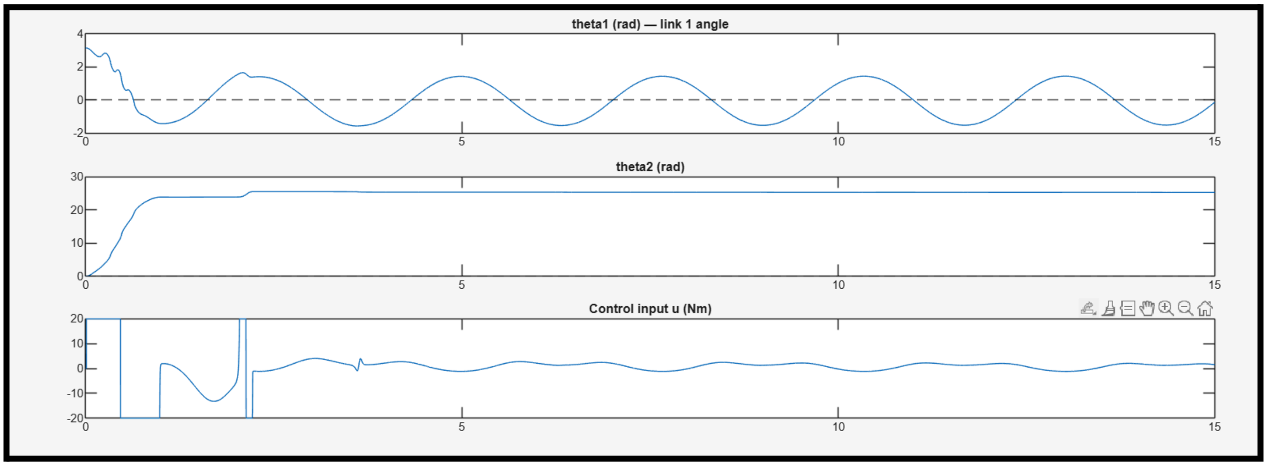


Fig1: State and Input Control Variables with respect to time

My implementation used some numerical tricks to make the simulation run faster:

1. **Pre-derived dynamics** → avoids runtime symbolic evaluation
2. **Angle wrapping** → prevents discontinuities
3. **Bounded control input** → prevents numerical integrator blow-up
4. **Small ODE timestep limit** (MaxStep=0.03) → prevents integration errors
5. **Switching hysteresis via two conditions**
6. **Correct gravity potential reference** → ensures the energy controller works
7. **Correct "Spong" coordinate system** → needed for LQR linearization stability

It was also noticed that when switching coordinates, LQR produced infinite gains.This happens when:

- the new coordinate transformation makes the upright equilibrium **non-differentiable**, or
- trigonometric expansions cause singularities near the equilibrium, or
- the linearization does not yield a controllable pair (A,B)

Thus, sticking to the Spong convention (θ=0 upright) is not just preference—it is mathematically required for a well-conditioned linearization.

Conclusion

This implementation successfully performs a full acrobot swing-up followed by upright stabilization using a hybrid controller composed of:

1. **Energy-based partial feedback linearization**
2. **Linearization-based LQR stabilization**

The key findings are:

- Choosing the correct coordinate system is essential for LQR solvability.
- Computing the desired energy requires careful definition of gravitational potential.
- Symbolic derivation combined with numerical caching yields both accuracy and speed.
- Switching logic must balance region size (too small → never switches, too large → LQR failure).
- The implemented controller matches the classic Spong (1995) approach and works robustly.

Overall, the simulation demonstrates a complete and well-designed control scheme for an underactuated system, with proper attention to modeling details, numerical stability, and hybrid control strategy.

.

REFERENCES

1. Spong, Mark W.. "Partial feedback linearization of underactuated mechanical systems." Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94) 1 (1994): 314-321 vol.1.