2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Hybrid Control Strategy for Robust Global Swing-Up of the Pendubot

Rowland W. O'Flaherty, Ricardo G. Sanfelice and Andrew R. Teel

Abstract— **Combining local state-feedback laws and open-loop schedules, we design a hybrid control algorithm for robust global stabilization of the pendubot to the upright configuration (both links straight up with zero velocity). Our hybrid controller performs the swing-up task by executing a decision-making algorithm designed to work under the presence of perturbations. The hybrid control algorithm features logic variables, timers, and hysteresis. We explicitly design, implement, and validate this control strategy in a real pendubot system using Matlab/Simulink with Real-time Workshop. Experimental results show the main capabilities of our hybrid controller.**

I. INTRODUCTION

The pendubot is a two-link underactuated mechatronic device frequently used for research in nonlinear control and robotics. As shown in Figure 1, the pendubot consists of a two-link planar robot arm (two coupled pendulums) with only one torque actuator at link 1. The pendubot is equipped with two optical sensors that measure angles at the joints. Each joint can fully rotate 360 degrees without any constraints on their motion. The pendubot has a total of four equilibrium configurations defined by the angular position and velocity of its links: both links resting ($\mathcal{A}_r$), link 1 resting and link 2 upright ($\mathcal{A}_{ru}$), link 1 upright and link 2 resting ($\mathcal{A}_{ur}$), and both links upright ($\mathcal{A}_u$).

One control challenge for the pendubot system is robust global stabilization of the upright configuration $\mathcal{A}_u$. While, several control strategies such as energy pumping [2], [4], jerk control [1], trajectory tracking [6], and hybrid control [9] have been proposed in previous years to solve the pendubot swing-up problem, the first provable robust and global feedback stabilizer stems from the hybrid control strategy in [7]. This general control strategy for robust global stabilization of nonlinear systems applied to the pendubot system combines, as illustrated in [7], local and regional feedback stabilizers with open-loop schedules to steer the trajectories of the pendubot to $\mathcal{A}_u$, even from configurations like $\mathcal{A}_{ru}$ and $\mathcal{A}_{ur}$ where energy-based control strategies would fail. In this paper, we explicitly design, implement, and validate the hybrid control algorithm proposed in [7].

The paper is organized as follows. Section II presents a model for the pendubot system and the control strategy. In Section III, we design the pieces building the hybrid controller. Section IV describes the testbed for hardware implementation and presents experimental results.

R. W. O'Flaherty and A. R. Teel: ECE Department, University of California, Santa Barbara, CA 93106, *oflaherty@ece.ucsb.edu, teel@ece.ucsb.edu*.

R. G. Sanfelice: Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, MA 02139, *sricardo@mit.edu* (research performed while at the University of California, Santa Barbara).

Research supported in part by NSF grants CCR-0311084, ECS-0622253, and AFOSR grant FA9550-06-1-0134.

II. ROBUST GLOBAL STABILIZATION OF THE PENDUBOT

Through the use of Euler-Lagrange equations, we model the pendubot system in Figure 1 as a differential equation with states given by a vector of joint angles $\phi := [\phi_1\ \phi_2]^\top \in \mathbb{R}^2$ and angular velocities $\dot\phi \in \mathbb{R}^2$.

$$\ddot\phi = \mathbf{N}^{-1}\mathbf{R} - \mathbf{N}^{-1}\mathbf{O}\dot\phi - \mathbf{N}^{-1}\mathbf{P} - \mathbf{N}^{-1}\mathbf{Q}, \qquad (1)$$

where, omitting the arguments ϕ and $\dot\phi$ for simplicity,

$$\mathbf{N}(\phi) = \begin{bmatrix} \theta_1+\theta_2+2\theta_3\cos(\phi_2) & \theta_2+\theta_3\cos(\phi_2) \\ \theta_2+\theta_3\cos(\phi_2) & \theta_2 \end{bmatrix},$$

$$\mathbf{O}(\phi,\dot\phi) = \begin{bmatrix} -\theta_3\sin(\phi_2)\dot\phi_2 & -\theta_3\sin(\phi_2)\dot\phi_2 - \theta_3\sin(\phi_2)\dot\phi_1 \\ \theta_3\sin(\phi_2)\dot\phi_1 & 0 \end{bmatrix},$$

$$\mathbf{P}(\phi) = \begin{bmatrix} \theta_4\gamma\cos(\phi_1)+\theta_5\gamma\cos(\phi_1+\phi_2) \\ \theta_5\gamma\cos(\phi_1+\phi_2) \end{bmatrix},$$

$$\mathbf{Q}(\dot\phi) = \begin{bmatrix} \theta_6\dot\phi_1 \\ \theta_7\dot\phi_2 \end{bmatrix},\ \mathbf{R}(\phi,\dot\phi) = \begin{bmatrix} u \\ 0 \end{bmatrix},$$

The torque input is denoted as u while the seven parameters in $\mathbf{N}$, $\mathbf{O}$, $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ denoted as $\theta := \{\theta_1, \theta_2, \ldots, \theta_7\}$ contain the physical parameters of the pendubot system. These are given by: $\theta_1 = m_1a_1^2 + m_2l_1^2 + I_1$, $\theta_2 = m_1a_2^2 + I_2$, $\theta_3 = m_2l_1a_2$, $\theta_4 = m_1a_1 + m_2l_1$, $\theta_5 = m_2a_2$, $\theta_6 = \nu_1$, $\theta_7 = \nu_2$, where m_1, m_2; l_1, l_2; a_1, a_2; and I_1, I_2 denote the mass, length, center of mass, and moment of inertia for link 1 and link 2, respectively. The constant ν_1 and ν_2 define the viscous friction coefficients of the joint 1 and joint 2, respectively, and γ denotes gravity. The state equations can be written as

$$\dot{\mathbf{x}} = f(\mathbf{x}, u)\ , \qquad (2)$$

where $\mathbf{x} := [\phi_1\ \dot\phi_1\ \phi_2\ \dot\phi_2]^\top$ and f is obtained from (1). While we do not do it explicitly, we consider the following embedding for the angles: ϕ_1 and ϕ_2 are given by the angle

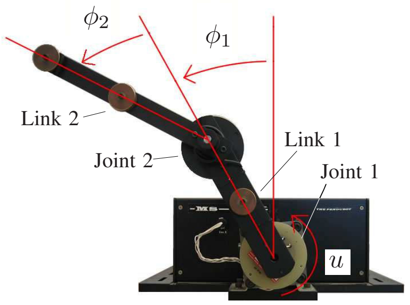


Fig. 1. *The pendubot system: a two-link pendulum with torque actuation in link 1. The pendulum angles are denoted by ϕ_1 and ϕ_2 for link 1 and link 2, respectively. The torque input is denoted by u.*

of vectors z_1 and z_2, respectively, such that z_1, z_2 belong to the unit circle S^1 in $\mathbb{R}^2$. Then, without loss of generality, we assume that $\phi_1 = \angle z_1 \mod 2\pi$ and $\phi_2 = \angle z_2 \mod 2\pi$, where $\angle : S^1 \to [-\pi, \pi)$ is such that $\angle z$ denotes the angle, positive in the counterclockwise direction, between z and the positive horizontal axis.

The four equilibrium points of the pendubot system are given by $\mathcal{A}_r := (\pi, 0, 0, 0)$, $\mathcal{A}_{ru} := (\pi, 0, \pi, 0)$, $\mathcal{A}_{ur} := (0, 0, \pi, 0)$, and $\mathcal{A}_u := (0, 0, 0, 0)$.

The *throw-and-catch* hybrid control algorithm for nonlinear systems proposed in [7] aims to robustly globally stabilize nonlinear systems to compact sets. For the pendubot system, this control strategy is as follows.

1. A local state-feedback stabilizing controller κ_r for the resting equilibrium point $\mathcal{A}_r$,
2. A local state-feedback stabilizing controller κ_u for the upright equilibrium point $\mathcal{A}_u$,
3. An open-loop controller $\alpha_{ur\to r}$ to transition from a neighborhood of $\mathcal{A}_{ur}$ to a neighborhood of $\mathcal{A}_r$,
4. An open-loop controller $\alpha_{ru\to r}$ to transition from a neighborhood of $\mathcal{A}_{ru}$ to a neighborhood of $\mathcal{A}_r$,
5. An open-loop controller $\alpha_{r\to u}$ to transition from a neighborhood of $\mathcal{A}_r$ to a neighborhood of $\mathcal{A}_u$, and
6. A bootstrap controller κ_0 to steer trajectories to a neighborhood of the union of the four equilibria from any other point,

to perform the following tasks.

A. Steer the trajectories to a neighborhood of $\mathcal{A}_r$ by applying $\alpha_{ur\to r}$ or $\alpha_{ru\to r}$ when the trajectory is near $\mathcal{A}_{ur}$ or $\mathcal{A}_{ru}$, respectively.
B. Stabilize the trajectories to the resting equilibrium by applying κ_r when the trajectory is near $\mathcal{A}_r$.
C. Steer the trajectories to a neighborhood of $\mathcal{A}_u$ by applying $\alpha_{r\to u}$ when the trajectory is near $\mathcal{A}_r$.
D. Stabilize the trajectories to $\mathcal{A}_u$ by applying κ_u when the trajectory is near $\mathcal{A}_u$.
E. Steer the trajectories toward the union of the four equilibria by applying κ_0 from any other point or when either $\alpha_{r\to u}, \alpha_{ur\to r}$, or $\alpha_{ru\to r}$ is not capable of performing the open-loop maneuver in the expected amount of time.

The executions in A and C are called *throw mode* while the local stabilization in B and D are called *catch mode*. The sets defining *throws* that start near $\mathcal{A}_{ur}$, $\mathcal{A}_{ru}$, $\mathcal{A}_r$ are denoted by $S_{ur\to r}$, $S_{ru\to r}$, $S_{r\to u}$, respectively. The sets of points at which the throws end when starting near $\mathcal{A}_{ur}$, $\mathcal{A}_{ru}$, $\mathcal{A}_r$ are denoted by $E_{r\to u}$, $E_{ur\to r}$, $E_{ru\to r}$, respectively. These sets are subsets of the basin of attraction of the local stabilizers κ_u and κ_r, denoted L_{v_u} and L_{v_r}, respectively. The execution in E corresponds to the *recovery mode*.

III. Control Design

To stabilize the pendubot trajectories to the upright configuration, a hybrid controller is implemented with logic variables and logic rules on a digital control board with sampling time $T_s = 0.0005$ sec.

A. Identification of Parameters

The model parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_7\}$ are determined using the least-squares method of system identification proposed in [5]. We proceed to apply this identification method to the pendubot as done in [3]. Let $\mathbf{T}(\mathbf{x})$ denote both motor torque and friction forces, $K(\mathbf{x})$ kinetic energy, and $U_r(\mathbf{x})$ potential energy (with respect to $\mathcal{A}_r$), that is, $\mathbf{T}(\mathbf{x}) = \mathbf{R}(\phi, \dot\phi) + \mathbf{Q}(\dot\phi)$, $K(\mathbf{x}) = \frac{1}{2}\dot\phi^\mathsf{T}\mathbf{N}(\phi)\dot\phi$, and $U_r(\mathbf{x}) = \theta_4\gamma(\cos(\phi_1) + 1) + \theta_5\gamma(\cos(\phi_1 + \phi_2) + 1)$. The energy theorem[1] leads to

$$\int_0^t \mathbf{T}(\mathbf{x})^\mathsf{T}\dot\phi dt = \sum_{i=1}^{7} \frac{\partial K(\mathbf{x})}{\partial \theta_i}\theta_i + \sum_{i=1}^{7} \frac{\partial U_r(\mathbf{x})}{\partial \theta_i}\theta_i, \quad (3)$$

where K and U_r are such that their partial derivatives with respect to the parameters θ are constant and independent of the parameters. Then, approximating the integral in (3) and using experimental data, a least-squares fit for the set of parameters θ is obtained. To implement this identification scheme, the pendubot was driven with an open-loop uniformly distributed random signal $u : [0, 15 \text{ sec}] \to [-1, 1]$. The system identification algorithm[2] was iterated 32 times to obtain a proper set of parameter values given by

$$\theta : \begin{cases} \theta_1 = 0.0266, \theta_2 = 0.0092, \theta_3 = 0.0074, \theta_4 = 0.1803, \\ \theta_5 = 0.0634, \theta_6 = 0.0130, \theta_7 = 0.0042, \end{cases}$$

which is consistent with the values obtained in [9].

B. Local State Feedback Control Laws

The local state feedback controllers (κ_u and κ_r) are designed to stabilize the trajectories of the pendubot to the upright and resting configurations through linearization and pole placement.

1) Design of κ_u: The linearized model of the plant in (2) around $\mathcal{A}_u$ is of the form $\dot{\mathbf{x}} = F_u\mathbf{x} + G_u u$ and is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 61.4 & -0.63 & -24.2 & 0.37 \\ 0 & 0 & 0 & 1 \\ -43.2 & 1.14 & 111 & -1.12 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 48 \\ 0 \\ -87 \end{bmatrix} u. \quad (4)$$

Let $\kappa_u(\mathbf{x}) := -\mathbf{K}_u^\mathsf{T}\mathbf{x}$. We solve for the control gain $\mathbf{K}_u$ using a Zero-Order Hold (ZOH) discretization of (4) with sampling time T_s. Discrete Linear Quadratic Regulator (LQR) with weighting functions $\mathbf{Q} = diag(13\ 5\ 5\ 3)$ and $R = 1$ is used to solve for $\mathbf{K}_u$ and a matrix $\mathbf{P}_u$ satisfying $\mathbf{P}_u = \mathbf{P}_u^\mathsf{T} > 0$ and $(F_u - G_u\mathbf{K}_u^\mathsf{T})^\mathsf{T}\mathbf{P}_u + \mathbf{P}_u(F_u - G_u\mathbf{K}_u^\mathsf{T}) < 0$. The resulting control gains are

$$\mathbf{K}_u = [-91.7, -15.2, -89.1, -10.6]^\mathsf{T},$$
$$\mathbf{P}_u = 10^6 \times \begin{bmatrix} 1.45 & 0.25 & 1.30 & 0.14 \\ 0.25 & 0.04 & 0.23 & 0.02 \\ 1.30 & 0.23 & 1.19 & 0.13 \\ 0.14 & 0.02 & 0.13 & 0.01 \end{bmatrix}.$$

[1]The energy theorem states that the work of forces applied to a system is equal to the change of total energy of the system.

[2]This identification algorithm uses MATLAB's `lsqnonneg()` function (a linear least squares function with nonnegativity constraints) and `trapz()` function for approximation of the integral in (3) to solve for a set of parameter values θ.

These control gains place all the poles of the closed-loop system inside the unit circle (i.e. controller $\kappa_u(\mathbf{x})$ locally asymptotically stabilizes $\mathcal{A}_u$). An estimate of the basin of attraction for this local stabilizer was computed experimentally by trial and error using the Lyapunov function $V_u(\mathbf{x}) := \mathbf{x}^\mathsf{T} P_u \mathbf{x}$. It was found that $L_{V_u}(\hat{c}_u) = \{\mathbf{x} \in \mathbb{R}^4 \mid V_u(\mathbf{x}) \leq \hat{c}_u\}$ with $\hat{c}_u = 15000$ is a sublevel set contained in the basin of attraction of κ_u for $\mathcal{A}_u$. For implementation purposes, let $c_u = 0.95\hat{c}_u$.

2) Design of κ_r: The linearized model of the plant around $\mathcal{A}_r$ is of the form $\dot{\mathbf{x}} = F_r \mathbf{x} + G_r u$ and is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -61.4 & -0.63 & 24.2 & 0.37 \\ 0 & 0 & 0 & 1 \\ 43.2 & 1.14 & -111 & -1.12 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 48 \\ 0 \\ -87 \end{bmatrix} u, \quad (5)$$

In a similar fashion to the design of κ_u, let $\kappa_r(\mathbf{x}) := -\mathbf{K}_r^\mathsf{T}\mathbf{x}$. A ZOH discretization of (5) around $\mathcal{A}_r$ is used to solve for the control gain. As we proceeded for κ_u, $\mathbf{K}_r$ and a matrix $\mathbf{P}_r$, such that $\mathbf{P}_r = \mathbf{P}_r^\mathsf{T} > 0$ and $(F_r - G_r\mathbf{K}_r^\mathsf{T})^\mathsf{T}\mathbf{P}_r + \mathbf{P}_r(F_r - G_r\mathbf{K}_r^\mathsf{T}) < 0$, are solved using a discrete LQR design with a weighting matrices $\mathbf{Q} = diag(13\ 5\ 5\ 3)$ and $R = 1$. The resulting control gains are

$$\mathbf{K}_r = [-6.68, 1.95, -9.3, -0.96]^\mathsf{T},$$
$$\mathbf{P}_r = 10^4 \times \begin{bmatrix} 10.1 & -0.08 & 8.19 & -0.03 \\ -0.08 & 0.34 & -0.31 & 0.18 \\ 8.19 & -0.31 & 9.07 & -0.15 \\ -0.03 & 0.18 & -0.15 & 0.10 \end{bmatrix}.$$

Again the control gains place all the poles of the closed-loop system inside the unit circle (i.e. controller $\kappa_r(\mathbf{x})$ locally asymptotically stabilizes $\mathcal{A}_r$). Analogously to $\kappa_u(\mathbf{x})$ an estimate of the basin of attraction for this local stabilizer was performed using the Lyapunov function $V_r(\mathbf{x}) := \mathbf{x}^\mathsf{T} P_r \mathbf{x}$. Experiments were performed to compute sublevel sets of V_r. It was found that $L_{V_r}(\hat{c}_r) = \{\mathbf{x} \in \mathbb{R}^4 \mid V_r(\mathbf{x}) \leq \hat{c}_r\}$, where $\hat{c}_r = 1000$, is a sublevel set that the pendubot is locally stabilized to $\mathcal{A}_r$. For implementation purposes, let $c_r = 0.95\hat{c}_r$.

C. Open-Loop Control

The control signals $\alpha_{r\to u}$, $\alpha_{ur\to r}$, and $\alpha_{ru\to r}$ are designed through partial feedback linearization as in [8] (see also [3]). To this end, note that (1) can be written as

$$N_{11}\ddot{\phi}_1 + N_{12}\ddot{\phi}_2 + O_{11}\dot{\phi}_1 + O_{12}\dot{\phi}_2 + P_1 + Q_1 = u \quad (6)$$
$$N_{21}\ddot{\phi}_1 + N_{22}\ddot{\phi}_2 + O_{21}\dot{\phi}_1 + O_{22}\dot{\phi}_2 + P_2 + Q_2 = 0 \quad (7)$$

where N_{ij}, O_{ij}, P_i, and Q_i, $i = 1, 2$ and $j = 1, 2$, correspond to the entries in the matrices N, O, P, and Q, respectively. Solving for $\ddot{\phi}_2$ in (7), substituting it into (6), and let

$$u = \overline{N_{11}} v + \overline{O_{11}}\dot{\phi}_1 + \overline{O_{12}}\dot{\phi}_2 + \overline{P_1} + \overline{Q_1} \quad (8)$$

where

$$\overline{N_{11}} = N_{11} - \frac{N_{12}N_{21}}{N_{22}},\ \overline{O_{11}} = O_{11} - \frac{N_{12}O_{21}}{N_{22}},$$

$$\overline{O_{12}} = O_{12},\ \overline{P_1} = P_1 - \frac{N_{12}P_1}{N_{22}},\ \overline{Q_1} = Q_1 - \frac{N_{12}Q_1}{N_{22}},$$

we get $\ddot{\phi}_1 = v$. Then, we can design an outer-loop controller to make $\phi_1, \dot{\phi}_1$ track a given reference trajectory $r^* := [\phi^*, \dot{\phi}^*]^\top$, where ϕ^* is the reference position and $\dot{\phi}^*$ is the reference velocity for link 1. We control v through a PD controller $v = J^D(\dot{\phi}^* - \dot{\phi}_1) + J^P(\phi^* - \phi_1)$, where J^D is the derivative gain and J^P is the position gain.

Through trial and error, we obtained that a reference signal $r_u^* := [\phi_u^*, \dot{\phi}_u^*]^\top$, where $\dot{\phi}^*{}_u = 0$, $\phi_u^* = 0$, and the gain values $J_u^D = 20$ and $J_u^P = 192$ consistently steered the trajectories from a neighborhood of $\mathcal{A}_r$ to a neighborhood of $\mathcal{A}_u$. To steer the trajectories to a neighborhood of $\mathcal{A}_r$ from nearby $\mathcal{A}_{ur}$ or $\mathcal{A}_{ru}$, a reference signal $r_r^* := [\phi_r^*, \dot{\phi}_r^*]^\top$, where $\dot{\phi}_r^* = 0$, $\phi_r^* = -\pi$, and gain values $J_r^D = 20$ and $J_r^P = 80$ were used.

1) Design of $\alpha_{r\to u}$: Given the reference signal r_u^* and gains J_u^D and J_u^P, the control input u resulting from (8) is recorded in memory using sampling time T_s during the throw from a point nearby $\mathcal{A}_r$ to a point nearby $\mathcal{A}_u$. The open-loop control $\alpha_{r\to u}$ is given by this recorded input. Experimentally, we estimate a sublevel set of $W_r(\mathbf{x}) = K(\mathbf{x}) + U_r(\mathbf{x})$, denoted $S_{r\to u}$, defined by a constant $c_{r\to u}$ from where such throws are successful. This constant was set to $c_{r\to u} = 0.0001$, which defines $S_{r\to u} := \{\mathbf{x} \in \mathbb{R}^4 \mid W_r(\mathbf{x}) \leq 0.0001\}$.

2) Design of $\alpha_{ur\to r}$ and $\alpha_{ru\to r}$: Given the reference signal r_r^* and gains J_r^D and J_r^P, the control input u is recorded with sampling time T_s for the pendubot during the throws from $\mathcal{A}_{ur}$ and $\mathcal{A}_{ru}$ to $\mathcal{A}_r$. Then, as in Section III-C.1, the open-loop control $\alpha_{ur\to r}$ and $\alpha_{ru\to r}$ is set equal to the respected recorded input u. Experimentally, we estimate a sublevel set of $W_{ur}(\mathbf{x})$ and $W_{ru}(\mathbf{x})$, denoted $S_{ur\to r}$ and $S_{ru\to r}$, and defined by the constants $c_{ur\to r}$ and $c_{ru\to r}$, respectively, from where such throws are successful. $W_{ur}(\mathbf{x})$ and $W_{ru}(\mathbf{x})$ are defined as $W_{ur}(\mathbf{x}) = K(\mathbf{x}) + U_{ur}(\mathbf{x})$ and $W_{ru}(\mathbf{x}) = K(\mathbf{x}) + U_{ru}(\mathbf{x})$, where $U_{ur}(\mathbf{x})$ and $U_{ru}(\mathbf{x})$ are the potential energy with respect to $\mathcal{A}_{ur}$ and $\mathcal{A}_{ru}$, respectively.Then, these sets are given by $S_{ur\to r} := \{\mathbf{x} \in \mathbb{R}^4 \mid W_{ur}(\mathbf{x}) \leq c_{ur\to r}\}$ and $S_{ru\to r} := \{\mathbf{x} \in \mathbb{R}^4 \mid W_{ru}(\mathbf{x}) \leq c_{ur\to r}\}$ with $c_{ru\to r} = c_{ru\to r} = 0.1$.

Estimates of the sets where open-loop schedules $\alpha_{r\to r}$, $\alpha_{ur\to r}$, and $\alpha_{ru\to r}$ take the trajectories of the pendubot were also obtained experimentally. It was found that $\alpha_{r\to u}$ consistently steered the trajectories within the sublevel set $E_{r\to u} = \{\mathbf{x} \in \mathbb{R}^4 \mid V_u(\mathbf{x}) \leq c'_{r\to u}\}$ with $c'_{r\to u} = 13000$. The open-loop controllers $\alpha_{ur\to r}$ and $\alpha_{ru\to r}$ consistently steered the trajectories within the sublevel sets $E_{ur\to r} = \{\mathbf{x} \in \mathbb{R}^4 \mid V_r(\mathbf{x}) \leq c'_{ur\to r}\}$ and $E_{ru\to r} = \{\mathbf{x} \in \mathbb{R}^4 \mid V_r(\mathbf{x}) \leq c'_{ru\to r}\}$ with $c'_{ur\to r} = c'_{ru\to r} = 600$. Finally, constants $\tau_{r\to u}$, $\tau_{ur\to r}$, and $\tau_{ru\to r}$ defining the upper bounds on the time required to reach $\mathcal{A}_u$ and $\mathcal{A}_r$ during throws were found to be $\tau_{r\to u} = 1.5$ and $\tau_{ur\to r} = \tau_{ru\to r} = 2$.

D. Bootstrap Control

To steer trajectories from points where the previous control laws are not applicable to a neighborhood of the union of the four equilibria, we design a bootstrap controller κ_0. Since the system is naturally damped, one would assume that $\kappa_0 \equiv 0$ would be sufficient. However, since friction in the pendubot system is small, a more sophisticated controller is developed.

From (3), the energy of the pendubot (with respect to $\mathcal{A}_r$) is given by $W_r(\mathbf{x}) = K(\mathbf{x}) + U_r(\mathbf{x})$ and satisfies

$$\langle \nabla W_r(\mathbf{x}), f(\mathbf{x}, 0)\rangle < 0 \qquad \forall \mathbf{x} \notin \mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru},$$

by virtue of the dissipative properties of the pendubot system. Writing $f(\mathbf{x}, u)$ as $f_e(\mathbf{x}) + g_e(\mathbf{x})u$, note that

$$\begin{aligned}\langle \nabla W_r(\mathbf{x}), f(\mathbf{x}, u)\rangle &= \langle \nabla W_r(\mathbf{x}), f_e(\mathbf{x}) + g_e(\mathbf{x})u\rangle \\ &= \langle \nabla W_r(\mathbf{x}), f(\mathbf{x}, 0)\rangle + \langle \nabla W_r(\mathbf{x}), g_e(\mathbf{x})\rangle u \ .\end{aligned}$$

Then, taking $u = \kappa_0(\mathbf{x}) := -\lambda\langle \nabla W_r(\mathbf{x}), g_e(\mathbf{x})\rangle$ with $\lambda > 0$ improves convergence speed. Through trial and error, we determined that $\lambda = 0.2$ gives good performance.

E. Angular velocity estimation

The vector of joint angular velocities $\dot\phi$ are estimated via a Kalman state estimator using the nonlinear model of the plant in (2), that is, the estimated states are given by

$$\dot{\hat{\mathbf{x}}} = f(\hat{\mathbf{x}}, u) + \mathbf{L}(\mathbf{y} - \hat{\mathbf{y}}) \quad \text{with } \mathbf{L} = \begin{bmatrix} 15.2 & -3.90 \\ 123 & -64.7 \\ -3.90 & 21.2 \\ -77.7 & 233 \end{bmatrix}.$$

where $\mathbf{L}$ is the estimator gain matrix, $\mathbf{y}$ is the measured output of the pendubot system given by ϕ, and $\hat{\mathbf{y}}$ is the estimated output $\mathbf{y}$. The matrix $\mathbf{L}$ is determined experimentally and uses the solution to the algebraic Riccati equation associated to the Kalman estimator for the estimation of the state of the linearized model of (2) at each equilibrium configuration $\mathcal{A}_r$, $\mathcal{A}_{ru}$, $\mathcal{A}_{ur}$, and $\mathcal{A}_u$.

F. Hybrid Controller

Two logic variables and a timer are used to implement the decision-making strategy described in tasks A-E in Section II. As discussed in [7], the decisions made by this control strategy can be visualized in a directed tree or graph with nodes given by the equilibrium points. The directed tree consists of two paths: path 1 defined by $\mathcal{A}_{ur} \to \mathcal{A}_r \to \mathcal{A}_u$ and path 2 defined by $\mathcal{A}_{ru} \to \mathcal{A}_r \to \mathcal{A}_u$. We number the nodes that define each path via pairs $(i, j) \in R := \{1, 2, 3\} \times \{1, 2\}$, where i indicates node number and j path number. Using this approach, we introduce the following convenient notation:

- For each $(i, j) \in R$, $\mathcal{A}_{i,j}$ represents a node in the directed tree. Thus, $\mathcal{A}_{1,1} = \mathcal{A}_{ur}$, $\mathcal{A}_{1,2} = \mathcal{A}_{ru}$, $\mathcal{A}_{2,1} = \mathcal{A}_{2,2} = \mathcal{A}_r$, and $\mathcal{A}_{3,1} = \mathcal{A}_{3,2} = \mathcal{A}_u$.
- For each $i \in \{2, 3\}$, κ_i represents the stabilizing controller for the node $\mathcal{A}_{i,j}$ with a estimate of its basin of attraction for $\dot{\mathbf{x}} = f(\mathbf{x}, \kappa_i(\mathbf{x}))$ given by D^c_i and with a recovery set given by $\overline{D^r_{-i}}$, where $\kappa_2 = \kappa_r$, $D^c_2 := L_{V_r}(c_r)$, $D^r_2 := \overline{\mathbb{R}^4 \setminus L_{V_r}(\hat c_r)}$, $\kappa_3 = \kappa_u$, $D^c_3 := L_{V_u}(c_u)$, and $D^r_3 := \overline{\mathbb{R}^4 \setminus L_{V_r}(\hat c_u)}$; see Section III-B regarding the construction of the sets L_{V_r} and L_{V_u}.
- For each $(i, j) \in \cup_{k\in\{1,2\}}((1, k)\cup(2, k))$, $\alpha_{(i,j)\to(i+1,j)}$ represents an open-loop controller that is capable of steering the trajectories defined by (2) from the set $S_{i,j}$ to an open set $E_{i,j}$ within $\tau_{i,j} \geq 0$ seconds.Thus, $\alpha_{(1,1)\to(2,1)} = \alpha_{ur\to r}$, $\alpha_{(1,2)\to(2,2)} = \alpha_{ru\to r}$, and $\alpha_{(2,1)\to(3,1)} = \alpha_{(2,2)\to(3,2)} = \alpha_{r\to u}$; $\tau_{1,1} = \tau_{ur\to r}$, $\tau_{1,2} = \tau_{ru\to r}$, and $\tau_{2,1} = \tau_{2,2} = \tau_{r\to r}$. In addition, for $X = S$ and $X = E$ define $X_{1,1} = X_{ur\to r}$, $X_{1,2} = X_{ru\to r}$, and $X_{2,1} = X_{2,2} = X_{r\to u}$, which implies, e.g., $S_{1,1} = S_{ur\to r}$ and $E_{1,1} = E_{ur\to r}$. See Section III-C regarding the construction of these sets.

Let $P := \{1, 2\}$, $Q^c := \{-3, -2\}$, $Q^t := \{1, 2\}$, $Q := Q^c \cup Q^t \cup \{0\}$, and $\mathcal{X} := \mathbb{R}^4 \times Q \times P \times \mathbb{R}$. We denote the logic variables by q and p, which take value in $q \in Q$ and $p \in P$, respectively, and define the mode of operation as follows: *throw mode* occurs at the q^{th} node of the p^{th} path when $q \in Q^t, p \in P$; *catch mode* occurs at the $|q|^{th}$ node when $q \in Q^c$; and *recovery mode* occurs when $q = 0$. We also denote by $\tau \in \mathbb{R}$ the timer state and define the closed-loop state to be $\xi := [\mathbf{x}^\mathsf{T}\, q\, p\, \tau]^\mathsf{T}$. The main ingredients of a control logic implementing tasks A-E are the following.

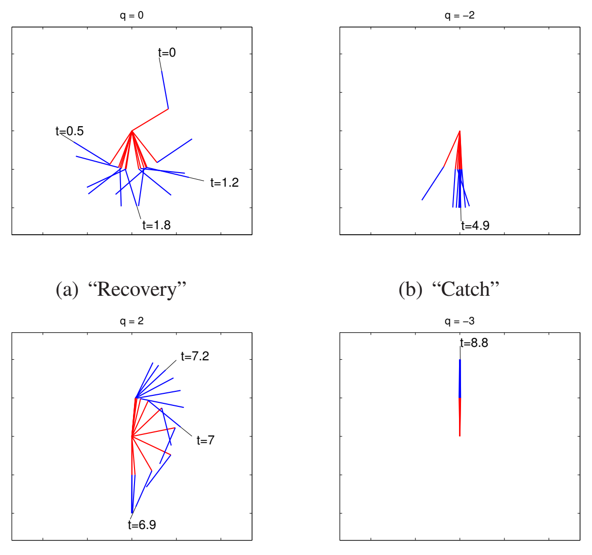
• Tasks A and C: When at a point from where the open-loop controls are applicable, that is, when $\mathbf{x} \in S_{|q|,p}$ for some q and p, then switch to *throw mode* and apply the corresponding open-loop control. Such a condition is true when ξ is in $D_t := \{\xi \in \mathcal{X} \mid \mathbf{x} \in S_{|q|,p}\}$. At such event, the controller updates (q, p) so that $\mathbf{x} \in S_{q^+,p^+}$, τ to zero, and sets the system's input u to $\alpha_{(q^+,p^+)\to(q^++1,p^+)}$.

• Tasks B and D: When at a point from where the local stabilizers are applicable, that is, when $\mathbf{x} \in D^c_{|q|}$ for some q, then switch to *catch mode* and apply the corresponding local stabilizer. Such is the case when ξ is in $D_c := \{\xi \in \mathcal{X} \mid \mathbf{x} \in D^c_{|q|}\}$. At such event, the controller updates (q, p) so that $\mathbf{x} \in D_{q^+}$, τ to zero, and sets u to κ_q.

• Task E: When the time spent in *throw mode* is larger or equal than the expected amount of time or when while in *catch mode* the state leaves the estimate of the basin of attraction of the local stabilizer currently applied, then switch to *recovery mode* and apply the bootstrap controller. Such a condition is true when ξ is in the set $D_r := \{\xi \in \mathcal{X} \mid q \in Q^t, \tau \geq \tau_{q,p}\} \cup \{\xi \in \mathcal{X} \mid q \in Q^c,\ \mathbf{x} \in D^r_{|q|}\}$. At such event, the controller updates (q, p) to $(0, p)$, τ to zero, and sets the system's input u to κ_0.

From the logic above, the output of the hybrid controller, which is connected to the system's input u and is denoted by the function κ_c, is given by

$$\kappa_c(\xi) := \begin{cases} \kappa_{|q|}(\mathbf{x}) & \text{if } q \in Q^c \\ \alpha_{(q,p)\to(q+1,p)}(\tau) & \text{if } q \in Q^t,\ p \in P \\ \kappa_0(\mathbf{x}) & \text{if } q = 0\ . \end{cases} \tag{9}$$



(a) "Recovery" (b) "Catch"

(c) "Throw" (d) "Catch"

Fig. 2. *Snap shots of link 1 (red line) and link 2 (blue line) from a frontal view while the different controllers are applied. (a) κ_0 is active while $q = 0$. (b) κ_r is active while $q = -2$. (c) $\alpha_{u\to r}$ is active while $q = 2$. (d) κ_u is active while $q = -3$.*

The sets D_t, D_c, and D_r build the jump sets of a hybrid controller, that is, the sets capturing the conditions at which the control variables are updated. Note that the construction of D_c and D_r are such that, while in *catch mode* with $q = -3$, a switch to another control mode would only occur if the trajectory enters the set D_r. Such a hysteresis mechanism confers robustness to the closed-loop system. A demonstration of the pendubot's motion while the controllers κ_0, κ_r, $\alpha_{u\to r}$, κ_u are being applied is depicted in Figure 2.

Each of the update laws described above are captured by a function of the state ξ, which defines the *jump map* of the hybrid controller. The implementation discussed in the next section only employs the conditions in the sets D_t, D_c, and D_r as well as their associated update laws. For this reason and due to space constraints, we do not reproduce the details of the hybrid controller itself here, but refer the reader to [7] for a detailed presentation and construction of the pieces building it. In fact, the hybrid controller for the pendubot system defined here follows the construction in [7] and, as stated in [7, Theorem 3.4], accomplishes the robust global stabilization task.

IV. Experimental Setup and Results

The experiments described in this paper are executed on the Mechatronic Systems Inc. pendubot model P-1. The implementation of the hybrid controller was carried out on a PC running The MathWorks, Inc. MATLAB, SIMULINK, and Real-Time Workshop. The interface between the PC and the pendubot consists of a Quanser MultiQ 3 I/O board running at a sampling frequency of 2 kHz.[3] The implementation setup can be seen in Figure 3.

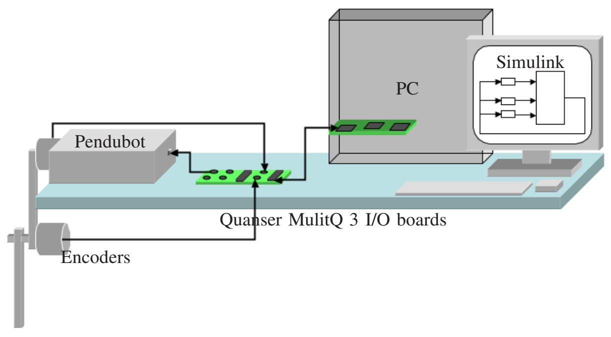


Fig. 3. *Illustration of the pendubot's interface with I/O boards and PC. The pendubot has two digital encoders, which are the inputs to the hybrid controller running on the PC in SIMULINK. The PC outputs a digital signal from the hybrid controller which is converted to analog signal for the Pendubot's motor by the Quanser MultiQ 3.*

Three different experiments of the pendubot illustrate the capabilities of our hybrid control strategy. In all the experiments (Figure 4(a), 4(b), 4(c)) the first plot (red curve) represents ϕ_1, the second plot (blue curve) represents ϕ_2, and third plot (black curve) represents q. [4]

A. Experiment 1

Figure 4(a) depicts an experiment that demonstrates the pendubot's robustness to small and large disturbances. In this experiment, the pendubot starts from an arbitrarily chosen point not close to $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. From this initial condition, the hybrid controller applies κ_0 bringing the trajectory towards a neighborhood of the union of the four equilibria. In this case, the trajectory reaches a neighborhood of $\mathcal{A}_r$. The trajectory enters the jump set D_c where a "catch" is performed (at around 2 sec.) to bring it to a neighborhood of $\mathcal{A}_r$. The trajectory enters the jump set D_t where a "throw" is performed to it from the resting configuration $\mathcal{A}_r$ to a neighborhood of the upright configuration $\mathcal{A}_u$. Then, the trajectories re-enters the jump set D_c, where it is caught by κ_u (at around 8 sec.), and consequently, locally stabilized to $\mathcal{A}_u$. The motion of the links up to this point is similar to the one depicted in Figure 2. Then, while the trajectory is in the neighborhood of $\mathcal{A}_u$, three small disturbances are applied and rejected by κ_u (between $t = 11$ and $t = 17$ sec.). At around 18 sec. a large disturbance is applied, which the local stabilizer κ_u is not able to reject. The trajectory enters the jump set D_r and the system jumps to recovery mode. From this condition, the normal catch-throw-catch sequence takes the system first to a neighborhood of $\mathcal{A}_r$ (at around 25 sec.) and then to $\mathcal{A}_u$ (at around 27 sec.). The plots in Figure 4(a) show that under the presence of small and large disturbances the system can recover and stabilize back to $\mathcal{A}_u$.

B. Experiment 2

Figure 4(b) depicts an experiment that demonstrates the pendubot's robustness to a large disturbance during a

[3]MATLAB's legacy_code() function was used to create specialized SIMULINK blocks for the hybrid control algorithm, which were written in C code and embedded into a SIMULINK model.

[4]A video of the pendubot using hybrid control strategy can accessed at http://www.scivee.tv/node/2721.

"throw". In this experiment, the pendubot starts at a random initial condition, a point not close to $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. As in Experiment 1, κ_0 is applied followed by κ_r and $\alpha_{r\to u}$. But while $\alpha_{r\to u}$ is being applied, a large disturbance obstructs the pendubot from reaching a neighborhood of the upright configuration $\mathcal{A}_u$ (at around 5.5 sec.). However, the hybrid controller is capable of detecting that the throw has failed by using the timer state. After bringing the links to a neighborhood of $\mathcal{A}_r$, attempts the throw again (at around 11 sec.). This "throw" is successful, and is followed by the application of κ_u (at around 12 sec.) to steer the trajectory toward $\mathcal{A}_u$. Since throws are an open-loop maneuver, the recovery feature of our hybrid control algorithm under perturbations is needed to have a robust closed-loop system.

C. Experiment 3

Figure 4(c) depicts an experiment that demonstrates the pendubot's ability to start near the equilibrium point of $\mathcal{A}_{ur}$ and be stabilized to $\mathcal{A}_u$. Starting from a small neighborhood of $\mathcal{A}_{ur}$, the controller jumps to throw mode bringing the trajectory to a neighborhood of $\mathcal{A}_r$ by applying $\alpha_{ur\to r}$. Then, the normal catch-throw-catch sequence takes the system first to a neighborhood of $\mathcal{A}_r$ (at around 1.5 sec.) and then to $\mathcal{A}_u$ (at around 5.5 sec.).

V. Conclusion

Using a novel control strategy for robust global stabilization of nonlinear systems, we design and validate experimentally a hybrid controller to globally swing up the pendubot with robustness to exogenous disturbances. We introduced our control strategy and provided a step-by-step design procedure. Experimental results prove the efficacy of the algorithm, even under the presence of large disturbances that are practically impossible to reject by any local stabilizer for the swing-up configuration.

References

[1] P. Absil and R. Sepulchre. A hybrid control scheme for swing-up acrobatics. In *Proc. 5th European Control Conference*, pages 2860–2864, September 2001.

[2] K. J. Astrom and K. Furuta. Swinging up a pendulum by energy control. *International Federation of Automatic Control*, E:37–42, 1996.

[3] D. J. Block. Mechanical design and control of the pendubot. Master's thesis, University of Illinois, 1991.

[4] I. Fantoni, R. Lozano, and M. W. Spong. Energy based control of the pendubot. *IEEE Trans. Automatic Control*, 45:725–729, April 2000.

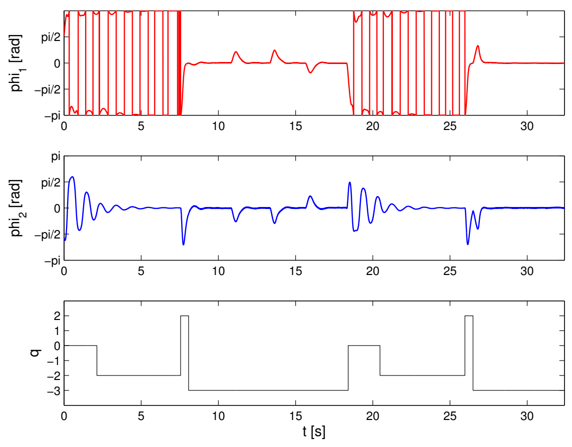
[5] M. Gautier and W. Khalil. On the identification of the inertial parameters of robots. In *Proc. IEEE Conf. Decision and Control*, pages 2264–2269, 1988.

[6] J. Rubi, A. Rubio, and A. Avello. Swing-up control problem for a self-erecting double inverted pendulum. *IEE Proceedings - Control Theory and Applications*, 149:169–175, March 2002.

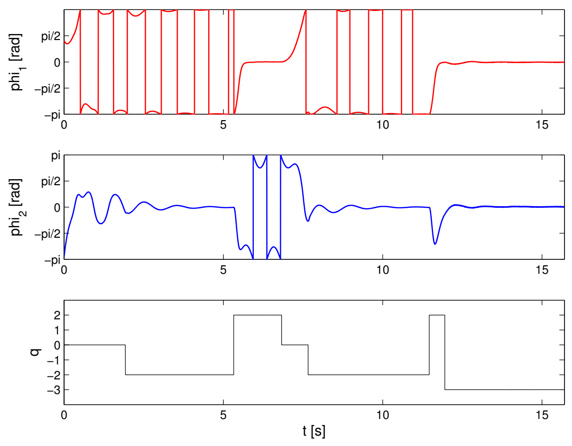
[7] R. G. Sanfelice and A. R. Teel. A "throw-and-catch" hybrid control strategy for robust global stabilization of nonlinear systems. *Proc. 26th American Control Conference*, pages 3470–3475, 2007.

[8] M. W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Proc. International Conf. Intelligent Robots and Systems*, pages 314–321, 1994.

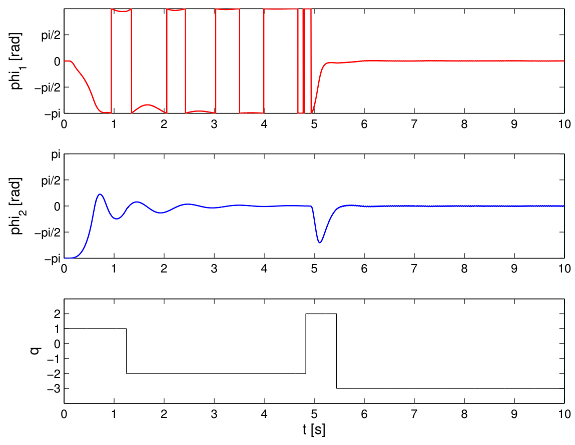
[9] M. Zhang and T. Tarn. Hybrid control of the pendubot. *ASME Transactions On Mechatronics*, 7:79–86, 2002.



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Fig. 4. *(a) Experiment 1: results of using the hybrid controller to stabilize the pendubot from a random initial condition in spite of small and large disturbances. (b) Experiment 2: results of using the hybrid controller to stabilize the pendubot from a random initial condition and in spite of a large disturbance during a "throw" from $\mathcal{A}_r$ to $\mathcal{A}_u$. (c) Experiment 3: results of using the hybrid controller to stabilize the pendubot from initial condition near the $\mathcal{A}_{ur}$ equilibrium.*