Robust Global Swing-Up of the Pendubot via Hybrid Control

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 robustly 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 the control strategy and implement it in a real pendubot system using Matlab/Simulink with Real-time Workshop. Experimental results show the main capabilities of our hybrid controller.**

I. Introduction

Following the hybrid control strategy in [5] for robust global stabilization of nonlinear systems, we design and implement in hardware a hybrid control strategy that swings the pendubot to the upright position in a robust, globally asymptotic manner, where both links have zero velocity. The control strategy combines the following local feedback stabilizers and open-loop control signal to steer the trajectories of the pendubot to the desired equilibrium point. By linearizing the system at the upright and resting equilibrium points ($\mathcal{A}_u$ and $\mathcal{A}_r$, respectively), we construct local stabilizers for points in these neighborhoods. More specifically, we construct an open-loop control signal to take the state to a neighborhood of $\mathcal{A}_u$ from a neighborhood of $\mathcal{A}_r$, and two different open-loop controllers to take the state to a neighborhood of $\mathcal{A}_r$ from a neighborhood of around the two other equilibrium ($\mathcal{A}_{ur}$, with the first link up and the second link down, and $\mathcal{A}_{ru}$, with the first link down and the second link up). These control laws are combined with a "bootstrap" feedback controller, which is designed to steer the state to a neighborhood of the union of $\mathcal{A}_u$, $\mathcal{A}_r$, $\mathcal{A}_{ur}$, and $\mathcal{A}_{ru}$, to globally swing up the pendubot to the point $\mathcal{A}_u$. Our hybrid controller includes decision-making and hysteresis features that confer a margin of robustness to exogenous signals.

The paper is organized as follows. In Section II, we present a model for the pendubot system and our control strategy. In Section III, we describe the hybrid controller design and implementation that is used to globally stabilize the pendubot to the upright equilibrium. In Section IV, we describe the tools and hardware used to conduct the experiment, and present the experimental results.

R. W. O'Flaherty and A. R. Teel: Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, *row_007@umail.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 at the Department of Electrical and Computer Engineering, University of California, Santa Barbara).

II. Robust Global Stabilization of the Pendubot

A. System Dynamics

The pendubot is an underactuated mechatronic device frequently used for research in nonlinear control and robotics. As shown in Figure 1, it consists of a two-link planar robot arm (two coupled pendulums) with only one torque actuator at the first link. The pendubot is equipped with two optical sensors that measure angles at the shoulder and elbow joints. Each joint can fully rotate 360 degrees without any constraints on their motion. The pendubot has a total of four equilibrium points defined by the position of the inner and outer links: fully resting, resting and upright, upright and resting, and fully upright.

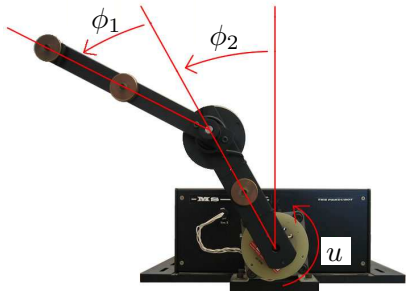

Fig. 1. *The pendubot system: a two-link pendulum with torque actuation u in the first link.*

Through use of Euler-Lagrange equations, we characterize the pendubot system as a vector of joint angles, ϕ, and a vector of their corresponding angular velocities, $\dot{\phi}$. We define $\phi := [\phi_1 \ \phi_2]^\mathsf{T} \in \mathbb{R}^2$ and $\dot{\phi} := [\dot{\phi}_1 \ \dot{\phi}_2]^\mathsf{T} \in \mathbb{R}^2$. As shown in [2] these state vectors also serve to characterize the matrices employed by the Euler-Lagrange equation: inertia, centrifugal Coriolis, derivative of potential energy, and viscous friction ($\mathbf{N}(\phi)$, $\mathbf{O}(\phi,\dot{\phi})$, $\mathbf{P}(\phi)$, and $\mathbf{Q}(\dot{\phi})$ respectively). Together, these functions define the equation of motion for torque,

$$\mathbf{R}(\phi,\dot{\phi}) = \mathbf{N}(\phi)\ddot{\phi} + \mathbf{O}(\phi,\dot{\phi})\dot{\phi} + \mathbf{P}(\phi) + \mathbf{Q}(\dot{\phi}), \tag{1}$$

which may be re-written as

$$\ddot{\phi} = \mathbf{N}^{-1}\mathbf{R} - \mathbf{N}^{-1}\mathbf{O}\dot{\phi} - \mathbf{N}^{-1}\mathbf{P} - \mathbf{N}^{-1}\mathbf{Q}, \tag{2}$$

with the function variables, ϕ and $\dot{\phi}$, omitted for simplicity.

The respected matrices and vectors are given by:

$$\begin{aligned}
\mathbf{N}(\phi) &= \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos\phi_2 & \theta_2 + \theta_3 \cos\phi_2 \\ \theta_2 + \theta_3 \cos\phi_2 & \theta_2 \end{bmatrix} \\
\mathbf{O}(\phi, \dot{\phi}) &= \begin{bmatrix} -\theta_3 \sin\phi_2 \dot{\phi}_2 & -\theta_3 \sin\phi_2 \dot{\phi}_2 - \theta_3 \sin\phi_2 \dot{\phi}_1 \\ \theta_3 \sin\phi_2 \dot{\phi}_1 & 0 \end{bmatrix} \\
\mathbf{P}(\phi) &= \begin{bmatrix} \theta_4 g \cos\phi_1 + \theta_5 g \cos\phi_1 + \phi_2 \\ \theta_5 g \cos\phi_1 + \phi_2 \end{bmatrix} \\
\mathbf{Q}(\dot{\phi}) &= \begin{bmatrix} \theta_6 \dot{\phi}_1 \\ \theta_7 \dot{\phi}_2 \end{bmatrix} \\
\mathbf{R}(\phi, \dot{\phi}) &= \begin{bmatrix} u \\ 0 \end{bmatrix}.
\end{aligned}$$

u is defined as the input torque to the system. The seven parameters (denoted as $\theta_1 \cdots \theta_7$) that define $\mathbf{N}(\phi)$, $\mathbf{O}(\phi, \dot{\phi})$, $\mathbf{P}(\phi)$, $\mathbf{Q}(\dot{\phi})$, and $\mathbf{R}(\phi, \dot{\phi})$, relate to the physical properties of the pendubot where $m_{1,2}$, $l_{1,2}$, $a_{1,2}$, and $I_{1,2}$ identify the mass, length, center of mass, and moment of inertia of the inner (link 1) and outer (link 2) links. $F_{v_{1,2}}$ defines the viscous friction coefficients of the shoulder (joint 1) and elbow (joint 2) joints.

$$\begin{aligned}
\theta_1 &= m_1 a_1^2 + m_2 l_1^2 + I_1 \\
\theta_2 &= m_1 a_2^2 + I_2 \\
\theta_3 &= m_2 l_1 a_2 \\
\theta_4 &= m_1 a_1 + m_2 l_1 \\
\theta_5 &= m_2 a_2 \\
\theta_6 &= F_{v_1} \\
\theta_7 &= F_{v_2}.
\end{aligned}$$

The general form of the state equations is thereby derived as

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) \Rightarrow \begin{bmatrix} \dot{x_1} = \dot{\phi}_1 \\ \dot{x_2} = f_1(\mathbf{x}, u) \\ \dot{x_3} = \dot{\phi}_2 \\ \dot{x_4} = f_2(\mathbf{x}, u) \end{bmatrix}. \tag{3}$$

$\mathbf{x} := \begin{bmatrix} \phi_1 & \dot{\phi}_1 & \phi_2 & \dot{\phi}_2 \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$ and $f(\mathbf{x}, u) : \mathbb{R}^4 \times \mathbb{R} \to \mathbb{R}^4$ is a nonlinear, locally Lipschitz function that define the dynamics of the pendubot.

The equilibrium points of the pendubot system are given by

- Upright ($\mathcal{A}_u$): $\phi_1 = 0$, $\dot{\phi}_1 = 0$, $\phi_2 = 0$, $\dot{\phi}_2 = 0$;
- Resting ($\mathcal{A}_r$): $\phi_1 = -\pi$, $\dot{\phi}_1 = 0$, $\phi_2 = 0$, $\dot{\phi}_2 = 0$;
- Upright/Resting ($\mathcal{A}_{ur}$): $\phi_1 = 0$, $\dot{\phi}_1 = 0$, $\phi_2 = \pi$, $\dot{\phi}_2 = 0$;
- Resting/Upright ($\mathcal{A}_{ru}$): $\phi_1 = -\pi$, $\dot{\phi}_1 = 0$, $\phi_2 = \pi$, $\dot{\phi}_2 = 0$.

These are depicted in Figure 2.

B. Hybrid Control Strategy

Several control strategies such as energy pumping [3], trajectory tracking [4], and jerk control [1] have been use to stabilize the pendubot system to the upright position with zero velocity. These strategies, however, do not guarantee swing up to $\mathcal{A}_u$ for every initial condition $\mathbf{x} \in \mathbb{R}^4$. In particular, closed-loop solutions resulting from these strategies starting from either of the unstable equilibrium points $\mathcal{A}_{ur}, \mathcal{A}_{ru}$ stay there for all time.

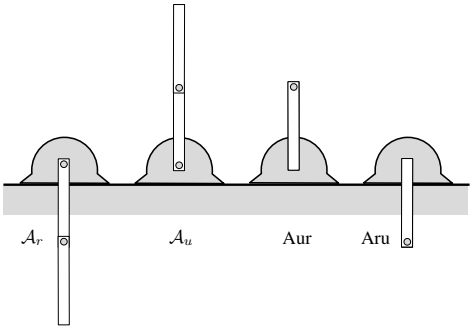

Fig. 2. *Equilibrium configurations of the pendubot.*

The hybrid control algorithm for nonlinear systems in [5], referred to as *throw-and-catch control*, has been proposed to robustly globally stabilize nonlinear systems to compact sets. As outlined in [5, Section II.A], this strategy solves the problem of swinging up the pendubot to $\mathcal{A}_u$ globally and robustly. We review this control strategy below.

Given

- a local state-feedback stabilizing controller κ_r for the resting equilibrium point $\mathcal{A}_r$,
- a local state-feedback stabilizing controller κ_u for the upright equilibrium point $\mathcal{A}_u$,
- an open-loop controller $\alpha_{ru \to r}$ to transition from $\mathcal{A}_{ru}$ to $\mathcal{A}_r$,
- an open-loop controller $\alpha_{ur \to r}$ to transition from $\mathcal{A}_{ur}$ to $\mathcal{A}_r$,
- an open-loop controller $\alpha_{r \to u}$ to transition from $\mathcal{A}_r$ to $\mathcal{A}_u$,
- a bootstrap controller κ_r to remove energy quickly from the system, and
- a reset controller κ_0 to recalibrate joint angles,

execute the following algorithm:

1) When the pendubot state is near $\mathcal{A}_{ur}$ or $\mathcal{A}_{ru}$ apply $\alpha_{ur \to r}$ or $\alpha_{ru \to r}$, respectively, to steer the trajectories toward $\mathcal{A}_r$.
2) When the pendubot state is near $\mathcal{A}_r$ apply κ_r to steer the trajectories to $\mathcal{A}_r$.
3) When the pendubot state is near $\mathcal{A}_r$ apply $\alpha_{r \to u}$ to steer the trajectories towards $\mathcal{A}_u$.
4) When the pendubot state is near $\mathcal{A}_u$ apply κ_u to stabilize the pendubot in the upright position.
5) For any other point in $\mathbb{R}^4$, apply κ_e to steer the trajectories toward $\mathcal{A}_r$.
6) When the pendubot does not reach $\mathcal{A}_r$ for more then τ_{reset} seconds, apply κ_0 for τ_{settle} seconds to recalibrate the link angles.

The transitions between equilibrium points in 1) and 2) is called *throw* mode, and the local stabilization in 3) and 4) is called *catch* mode. The sets where throws nearby $\mathcal{A}_{ur}$, $\mathcal{A}_{ru}$, or $\mathcal{A}_r$ are started are denoted by $S_{ur \to r}$, $S_{ru \to r}$, or

$S_{r\to u}$, respectively. The sets where throws nearby $\mathcal{A}_r$ and $\mathcal{A}_u$ are finished are denoted by L_{W_r} and L_{W_u}, respectively. The decision made in 5) and 6) correspond to *recovery* mode. Figure 3 shows the combination of these tasks to accomplish global stabilization to the point $\mathcal{A}_u$ of the pendubot.

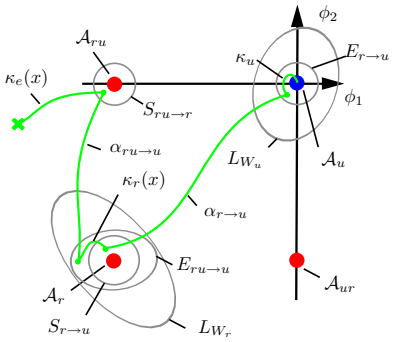

Fig. 3. *Control strategy for robust global stabilization of the pendubot to the point $\mathcal{A}_u$. A sample trajectory in the ϕ_1, ϕ_2 plane resulting from our control strategy is depicted. From the initial point $\times$, the trajectory is steered to the neighborhood $S_{ru\to r}$ of $\mathcal{A}_{ru}$ with $\kappa_r(x)$, from which it is "thrown" to the neighborhood $E_{ru\to r}$ of $\mathcal{A}_r$ with the control law $\alpha_{ru\to r}$. The local stabilizer κ_r "catches" the state to a point in $S_{r\to u}$ from where the open-loop law $\alpha_{r\to u}$ is applied. Finally, after the "throw", the state reaches a point in $E_{r\to u}$ and the last "catch" by the local stabilizer κ_u steers the trajectory to $\mathcal{A}_u$.*

III. Hybrid Control Design and Implementation

To control the pendubot's position to the upright equilibrium, 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 $\theta_1 \cdots \theta_7$ were determined using a least squares method of system identification. 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 $\mathcal{A}_r$). Using the energy theorem[1] for a standard over-determined matrix equation of the form $\mathbf{Ax} = \mathbf{b}$, we obtain

$$\int_0^t \mathbf{T}(\mathbf{x})^\mathsf{T} \dot{\phi} dt = \sum_{i=1}^{7} \frac{\partial K(\mathbf{x})}{\partial \theta_i}\theta_i + \sum_{i=1}^{7} \frac{\partial U_r(\mathbf{x})}{\partial \theta_i}\theta_i, \quad (4)$$

where

$$\mathbf{T}(\mathbf{x}) = \mathbf{R}(\phi, \dot{\phi}) + \mathbf{Q}(\dot{\phi}), \quad (5)$$

$$K(\mathbf{x}) = \frac{1}{2}\dot{\phi}^\mathsf{T}\mathbf{N}(\phi)\dot{\phi}, \quad (6)$$

$$U_r(\mathbf{x}) = \theta_4 g(\cos(\phi_1) + 1) + \theta_5 g(\cos(\phi_1 + \phi_2) + 1). \quad (7)$$

To implement this identification scheme, the pendubot was driven with an open-loop random signal $u_{Id} : [0, 15 \text{ sec}] \to [-1, 1]$. The input, u_{Id}, and outputs, ϕ_1 and ϕ_2, of the pendubot were recorded and loaded into a Matlab M-file where the identification algorithm was executed. This identification algorithm used equation (4)[2] and MATLAB's `lsqnonneg()` function (a linear least squares function with nonnegativity constraints) to solve for a set of parameter values θ. The system identification algorithm was iterated 32 times to obtain a proper set of parameter values. The average values resulting from the runs are:

$$\theta : \begin{cases} \theta_1 = 0.0271, \theta_2 = 0.0091, \theta_3 = 0.0072, \theta_4 = 0.1817, \\ \theta_5 = 0.0640, \theta_6 = 0.0034, \theta_7 = 0.0014. \end{cases}$$

The derived parameter set is consistent with a similar published pendubot model [2], and proves to be an appropriate representation of the system based on experimental tests.

B. Local State Feedback Control

The local state feedback controllers (κ_r and κ_u) are designed to steer the system from points in a defined basin of attraction to the the upright and resting equilibrium ($\mathcal{A}_u$ and $\mathcal{A}_r$, respectively) through linearization and pole placement.

1) Design of κ_u: The linearized model of the plant around $\mathcal{A}_u$ is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 60 & -0.16 & -23 & 0.14 \\ 0 & 0 & 0 & 1 \\ -39 & 0.29 & 111 & -0.42 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 47 \\ 0 \\ -84 \end{bmatrix} u. \quad (8)$$

Given $\kappa_u(\mathbf{x}) := -\mathbf{K}_u\mathbf{x}$, we solve for the control gain using a ZOH discretization of (3) around $\mathcal{A}_u$ with sampling time T_s. A discrete LQR function is used to solve for $\mathbf{K}_r$ with weighting functions $\mathbf{Q} = diag(13\ 5\ 5\ 3)$ and $R = 1$. The resulting control gains are $\mathbf{K}_r = [-7.41, 1.90, -10.02, -1.01]$. These control gains place all the poles of the closed-loop system inside the unit circle at locations $\{0.9139, 0.9992, 0.9965 \pm i0.00051\}$. The 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 and it was found that initial $W_u(\mathbf{x})$ lower than 0.55 permits the pendubot to locally stabilize $\mathcal{A}_u$. A much more conservative value is used for the sublevel set $L_{W_u}(c_u) = \{W_u(\mathbf{x}) \leq c_u\}$, $c_u = 0.08$, where $W_u(\mathbf{x}) = K(\mathbf{x}) + U_u(\mathbf{x})$ and $U_u(\mathbf{x})$ is the potential energy with respect to $\mathcal{A}_u$. Figure 4 characterized the evolution of the energy from different initial energy levels $W_u(\mathbf{x})$.

2) Design of κ_r: The linearized model of the plant around $\mathcal{A}_r$ is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -60 & -0.16 & 23 & 0.14 \\ 0 & 0 & 0 & 1 \\ 39 & 0.29 & -111 & -0.42 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 47 \\ 0 \\ -84 \end{bmatrix} u, \quad (9)$$

Given $\kappa_r(\mathbf{x}) := -\mathbf{K}_r\mathbf{x}$, we solve for the control gain using a ZOH discretization of (3) around $\mathcal{A}_r$ with a sampling time T_s. A discrete LQR function is

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

[2]MATLAB's `trapz()` function was used to approximate the integral in (4).

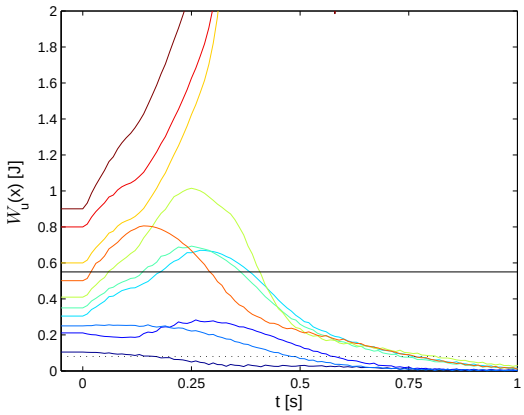

Fig. 4. The catch mode energy evolution for varying initial pendubot positions with near zero energy (with respect to $\mathcal{A}_u$) using control law κ_u. All initial $W_u(\mathbf{x})$ lower than 0.55 (solid black line) can be stabilized by the controller κ_u. The black dotted represents the c_u value that is used in the sublevel set L_{W_u}.

used to solve for $\mathbf{K}_r$ with weighting functions $\mathbf{Q} = diag(13\ 5\ 5\ 3)$ and $R = 1$. The resulting control gains are $\mathbf{K}_r = [-88.06, -14.44, -85.46, -10.27]$. These control gains place all the poles of the closed-loop system inside the unit circle at locations $\{0.9145, 0.9992, 0.9995 \pm i0.00035\}$. The controller $\kappa_r(\mathbf{x})$ locally asymptotically stabilizes $\mathcal{A}_r$. We computed an estimate of its basin of attraction experimentally by trial and error as the sublevel set $L_{W_r} = \{W_r(\mathbf{x}) \leq c_r\}$, $c_r = 0.2$, where $W_r(\mathbf{x}) = K(\mathbf{x}) + U_r(\mathbf{x})$ and $U_r(\mathbf{x})$ is the potential energy with respect to $\mathcal{A}_r$. Similar experiments to the ones in Figure 4 were also performed to compute c_r.

3) Determining Additional Control Parameters: The catch mode energy evolution figure (Figure 4) can also be used to determine $\overline{c}_r$ and $\overline{c}_u$, which define maximum sublevel sets in which κ_r and κ_u are to be applied, respectively. To incorporate hysteresis switching, we design $\overline{c}_r$ and $\overline{c}_u$ to be larger than c_r and c_u, respectively. Then, to leave catch mode, the energy of the system needs to change at least $\overline{c}_r - c_r$ or $\overline{c}_u - c_u$, respectively. $\overline{c}_u$ is extracted from Figure 4, where we infer that when the energy W_u is greater than 2, the pendubot will not stabilize the $\mathcal{A}_u$ equilibrium, thereby requiring to jump to the recovery mode. $\overline{c}_r$ was obtained similarly; $\overline{c}_r = 2$.

C. Open-Loop Control

The necessary control inputs to swing the links either up or down are are found through partial feedback linearization as described in [2]. To perform partial feedback linearization on the pendubot system (1) is broken up into

$$N_{11}\ddot{\phi}_1 + N_{12}\ddot{\phi}_2 + O_{11}\dot{\phi}_1 + O_{12}\dot{\phi}_2 + P_1 + Q_1 = u \quad (10)$$

and

$$N_{21}\ddot{\phi}_1 + N_{22}\ddot{\phi}_2 + O_{21}\dot{\phi}_1 + O_{22}\dot{\phi}_2 + P_2 + Q_2 = 0 \quad (11)$$

where N_{ij}, O_{ij}, P_i, and Q_i correspond to the entries in the matrices N, O, P, and Q, respectively. $\ddot{\phi}_2$ is solved for in (11) and plugged back into (10) to get

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

where $v = \ddot{\phi}_1$ and

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

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

This allows us to design an outer loop controller for the input v that will track a given trajectory of ϕ_1. We control v through a PD controller with feed-forward acceleration,

$$v = J^D(\dot{\phi^R} - \dot{\phi}_1) + J^P(\phi^R - \phi_1), \quad (13)$$

where J^D is the derivative gain, J^P is the position gain, ϕ^R is the pendubot's inner link reference position, and ϕ^R is its respective reference velocity.

Through trial and error, we found that a reference signal, r_u^*, defined as $\dot{\phi}_u^* = 0$ and $\phi_u^* = 0$ and gain values of $J_u^D = 20$ and $J_u^P = 191$ consistently swung the links up to a neighborhood of $\mathcal{A}_u$ from a neighborhood of $\mathcal{A}_r$. To bring the links to a neighborhood of $\mathcal{A}_r$ from $\mathcal{A}_{ur}$ or $\mathcal{A}_{ru}$ a reference signal, r_r^*, defined as $\dot{\phi}_r^* = 0$ and $\phi_r^* = -\pi$ and gain values of $J_r^D = 20$ and $J_r^P = 80$ were used.

1) Design of $\alpha_{r\to u}$: Given a reference signal r_u^* and gains J_u^D and J_u^P, the control input u resulting from (12) is recorded on memory 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, denoted $S_{r\to u}$, defined by the constant $c_{r\to u}$ from where such throws are successful.

2) Design of $\alpha_{ur\to r}$ and $\alpha_{ru\to r}$: Given a reference signal r_r^* and gains J_r^D and J_r^P the control input u is recorded 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} and W_{ru}, denoted $S_{ur\to r}$ and $S_{ru\to r}$, and defined by the constants $c_{ur\to r}$ and $c_{ru\to r}$ from where such throws are successful, respectively, where $W_{ur}(\mathbf{x}) = K(\mathbf{x}) + U_{ur}(\mathbf{x})$ and $W_{ru}(\mathbf{x}) = K(\mathbf{x}) + U_{ru}(\mathbf{x})$, $U_{ur}(\mathbf{x})$ and $U_{ru}(\mathbf{x})$ are the potential energy with respect to $\mathcal{A}_{ur}$ and $\mathcal{A}_{ru}$, respectively.

3) Determining Constant Values: Experiments were conducted to determine appropriate values to use for the constants $c_{r\to u}$, $c_{ur\to r}$ and $c_{ru\to r}$. We studied the throw mode energy evolution for varying initial pendubot positions with near zero energy (with respect to $\mathcal{A}_r$) using control law $\alpha_{r\to u}$ to determine $c_{r\to u}$. Based on the results described in Figure 4 that all initial pendubot positions with $W_r(\mathbf{x})$ lower than 10^{-3} can be caught by the controller κ_u. This method was also used to determine $c_{ur\to r}$ and $c_{ru\to r}$, both are set to be equal to 0.1.

The time series data in Figure 5 can also be used to determine $\tau_{ur,ru\to r}$ and $\tau_{r\to u}$, which define the greatest time

interval required to reach $\mathcal{A}_r$ and $\mathcal{A}_u$ during a throw, respectively. We determined $\tau_{r\to u} = 1.5$ seconds and $\tau_{ur,ru\to r} = 2$ seconds.

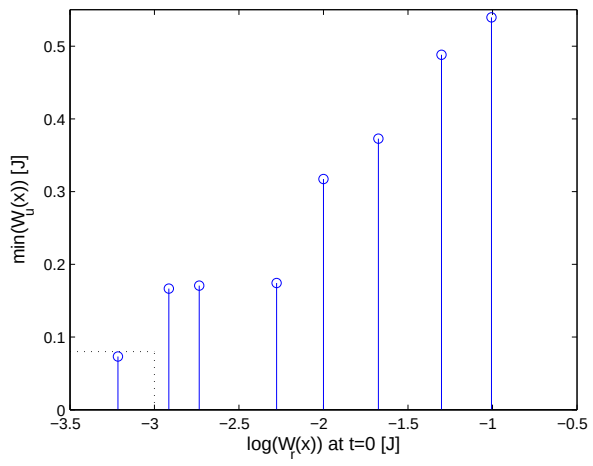

Fig. 5. Minimum energy (with respect to $\mathcal{A}_r$) resulting from using $\alpha_{r\to u}$. The y-axis describes the minimum energy (with respect to $\mathcal{A}_u$) resulting from the throw. Using $c_u = 0.8$, we confirmed that all initial pendubot positions with $W_r(\mathbf{x}) < 10^{-3}$ can be caught by the controller κ_u.

D. Bootstrap Control

In order to steer trajectories from points not in a neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$ to a small neighborhood around $\mathcal{A}_r$, energy is removed from the system via the controller κ_e. Since the system is naturally damped, one would assume that $\kappa_e \equiv 0$ would be sufficient to steer the trajectories of the pendubot to $\mathcal{A}_r$ in time. The friction in the pendubot is so small, however, that a more sophisticated controller is developed.

From (6) and (7), the energy of pendubot (with respect to $\mathcal{A}_r$) is given by $W_r(\mathbf{x}) = K(\mathbf{x}) + U_r(\mathbf{x})$ and satisfies

$$\langle \nabla W_r(\mathbf{x}), f(\mathbf{x}, u) \rangle < 0, \ \forall \mathbf{x} \neq 0. \tag{14}$$

Reformulating $f(\mathbf{x}, u)$ to $f_e(\mathbf{x}) + g_e(\mathbf{x})u$,

$$\begin{aligned} \langle \nabla E(\mathbf{x}), f(\mathbf{x}, u) \rangle &= \langle \nabla E(\mathbf{x}), f_e(\mathbf{x}) + g_e(\mathbf{x})u \rangle \\ &= \langle \nabla E(\mathbf{x}), f_e(\mathbf{x}) \rangle + \langle \nabla E(\mathbf{x}), g_e(\mathbf{x}) \rangle u. \end{aligned} \tag{15}$$

$\langle \nabla E(\mathbf{x}), f_e(\mathbf{x}) \rangle < 0, \ \forall \mathbf{x} \neq 0$ by virtue of the dissipative properties of the system. To guarantee that (15) is negative definite, we define the input as $u := -\lambda \langle \nabla E(\mathbf{x}), g_e(\mathbf{x}) \rangle$, where $\lambda \geq 0$. Through trial and error, we determined that $\lambda = 0.2$ provides the best results for the pendubot. We thereby design the bootstrap controller to be given by $\kappa_e(\mathbf{x}) = -0.2 \langle \nabla E(\mathbf{x}), g_e(\mathbf{x}) \rangle$.[3]

E. Reset Control

Due to the the numerical errors that arise when the pendubot links complete a number of revolutions, a reset control strategy is necessary to recalibrate the measured joint angles. Calibration is performed by setting $\kappa_0 \equiv 0$ in order to allow the pendubot to naturally return to its resting position, after which the states are reset to those defined by $\mathcal{A}_r$. When there are no numerical errors in the measured joint angles, κ_r brings the trajectories of the pendubot within $S_{r\to u}$ in some finite amount of time ($\tau_{reset} = 10$ seconds). Therefore, calibration occurs when the controller κ_r is applied to the system for more than some finite amount of time ($\tau_{reset} = 8$ seconds). κ_0 is applied to the system for $\tau_{settle} = 7$ seconds, before resetting the states to the resting equilibrium.

F. Angular velocity estimation

Joint velocities, $\dot{\phi}$, are estimated via a finite difference estimator. Joint velocities are calculated by taking the difference between the current sampled joint angle and the previous sampled joint angle then dividing that difference by the sampling period (T_s): $\dot{\phi} \approx (\phi(kT_s) - \phi((k-1)T_s))/T_s$. This method is susceptible to the noise introduced via the finite resolution of the optical encoders. We use first order low pass filters at the outputs of the joint positions to minimize the effect of noise. The poles of these filters were adjusted experimentally until reasonable results were reached. These poles were placed at 100 rad/s. The estimated angular velocity, $\hat{\dot{\phi}}$, implemented for the pendubot is defined as,

$$\hat{\dot{\phi}}(k) := 97.6(\phi(k-1) - \phi(k-2)) + 0.95\hat{\dot{\phi}}(k-1). \tag{16}$$

Experimentally, we found that this simple estimation algorithm works for the purposes of our problem. More advanced estimation techniques can be as well applied, e.g. Kalman filtering.

G. Hybrid Controller

The control laws defined in the previous sections are implemented in a hybrid controller as in [5]. Logic variables and a timer are used to implement the decision-making strategy described in Section II-B. Complete details about its implementation in a hybrid controller can be found in [5]. Its implementation is summarized below.

Two logic states, q and p, taking value in $Q := \{-3, -2, 0, 1, 2, 4\}$ and $P := \{1, 2\}$), and one timer, τ, define the state of the hybrid controller, denoted by $\mathcal{H}_c$. Let $\xi := [\mathbf{x}^\mathsf{T}, q, p, \tau]^\mathsf{T}$ be the control state. The value of logic variables q and p identify the appropriate control scheme:

q	p	Controller
-3	1 or 2	κ_u
-2	1 or 2	κ_r
0	1 or 2	κ_e
1	1	$\alpha_{ur\to r}$
1	2	$\alpha_{ru\to r}$
2	1 or 2	$\alpha_{r\to u}$
4	1 or 2	κ_0

In words, p denotes the "path" in the tree in Figure 3 taken by the trajectories, which can be either

$$\mathcal{A}_{ur} \to \mathcal{A}_r \to \mathcal{A}_u \text{ or } \mathcal{A}_{ru} \to \mathcal{A}_r \to \mathcal{A}_u \ ,$$

[3] $g_e(\mathbf{x})$ may be found using the following relationship: $g_e(\mathbf{x}) = \frac{\partial f(\mathbf{x},u)}{\partial u}$.

while q denotes the node in the current path.

Jumps in the logic state q and p occur as follows:

- When $q \neq 1$ and $W_{ur}(\mathbf{x}) \leq c_{ur\to r}$ or $W_{ru}(\mathbf{x}) \leq c_{ru\to r}$) then update q to 1, i.e., $q^+ = 1$.
- When $q = 0$ or $q = 1$ and $W_r(\mathbf{x}) \leq c_r$ then update q to -2, i.e., $q^+ = -2$.
- When $q = -2$ and $W_r(\mathbf{x}) \leq c_{r\to u}$ then update q to 2, i.e., $q^+ = 2$.
- When $q \neq -3$ and $W_u(\mathbf{x}) \leq c_u$ then update q to -3, i.e., $q^+ = -3$.
- When $q = 1$ and $\tau \geq \tau_{ur,ru\to r}$ then update q to 0, i.e., $q^+ = 0$.
- When $q = -2$ and $W_r(\mathbf{x}) \geq L_{v_r}$ then update q to 0, i.e., $q^+ = 0$.
- When $q = 2$ and $\tau \geq \tau_{r\to u}$ then update q to 0, i.e., $q^+ = 0$.
- When $q = -3$ and $W_u(\mathbf{x}) \geq L_{v_u}$ then update q to 0, i.e., $q^+ = 0$.
- When $q = 4$ and $\tau \geq \tau_{settle}$ then update q to 0, i.e., $q^+ = 0$.
- When $q = -2$ and $\tau \geq \tau_{reset}$ then update q to 4, i.e., $q^+ = 4$.
- When $W_{ur}(\mathbf{x}) \leq c_{ur\to r}$ then update p to 1, i.e., $p^+ = 1$.
- When $W_{ru}(\mathbf{x}) \leq c_{ur\to r}$ then update p to 2, i.e., $p^+ = 2$.

At every jump, the timer τ is reset to zero. Other variables remain constant and are omitted in the description above.

IV. Experimental Results

The experiments described in this paper are executed on the Mechatronic Systems Inc. pendubot model P-1. The hybrid controller makes use of a PC running The MathWorks, Inc. MATLAB, SIMULINK, and Real-Time Workshop. The PC communicated with the pendubot through the Quanser MultiQ3 I/O board at a sampling frequency of 2 kHz. MATLAB's legacy_code() function was used to create specialized SIMULINK blocks to allow the jump map and set to be written in C code and embedded into a SIMULINK model.

In this section, four different experiments of the pendubot illustrate the capabilities of our hybrid control strategy. Experiment 1 shows a nominal case with the pendubot starting with an arbitrary initial condition. This experiment is represented in three different plots shown in Figure 6 through Figure 7(b). Experiment 2 shows a nominal case with the pendubot starting at the equilibrium point $\mathcal{A}_{ur}$ and the results are in Figure 8. Experiment 3 shows how the closed-loop system can recover from small and large disturbances while the pendubot is being stabilized in the upright position. Results are shown in Figure 9. Experiment 4 shows closed-loop recovery from a large disturbance while the pendubot is in throw mode going from $\mathcal{A}_r$ to $\mathcal{A}_u$. Figure 10 shows the experimental data for Experiment 4. Additionally, a video of the pendubot using hybrid control to stabilize to the upright equilibrium can accessed at http://www.scivee.tv/node/2721.

A. Experiment 1

Figure 6, Figure 7(a), and Figure 7(b) are three different representations of Experiment 1. The first plot (blue curve) in

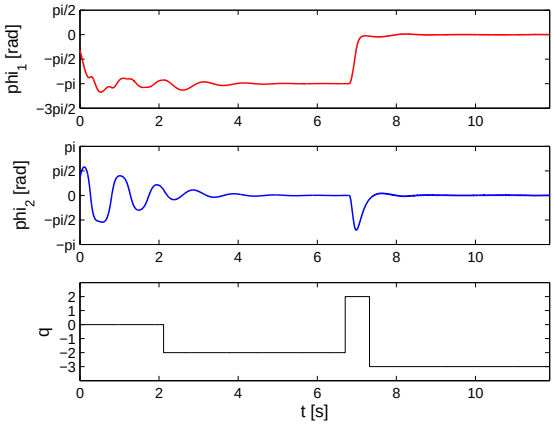

Fig. 6. Experiment 1: results using the hybrid control strategy to swing up the pendubot from a point not in the neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. Initial conditions: $x^0 = [-0.96, -9.68, 1.12, 12.5]^\mathsf{T}$, $q^0 = 0$, $p^0 = 1$, $\tau^0 = 0$. Pendubot angles: ϕ_1 (red), ϕ_2 (blue), q (black). While in recovery mode ($q = 0$), $\mathbf{x}$ is directed towards $\mathcal{A}_r$. A "catch" is performed (at around 2 $sec.$) to bring $\mathbf{x}$ to a neighborhood of $\mathcal{A}_r$. A "throw" is performed (at around 6.5 $sec.$) from nearby $\mathcal{A}_r$ to nearby $\mathcal{A}_u$. Finally, another "catch" is performed (at around 7.5 $sec.$) to steer $\mathbf{x}$ to $\mathcal{A}_r$.

Figure 6 represents ϕ_1, the second plot (red curve) represents ϕ_2, and third plot (the black curve) represents q. In this experiment, the pendubot starts from an arbitrarily chosen point not in neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. From this initial condition, the hybrid controller applies κ_e bringing $\mathbf{x}$ to a neighborhood of $\mathcal{A}_r$. When $W_r(\mathbf{x}) \leq c_r$ the hybrid controller has a jump that maps q to -2, and therefore, the control κ_r is applied to the system. When the pendubot state is such that $W_r \leq c_{r\to u}$, a jump to $q = 2$ follows and the open-loop control $\alpha_{r\to u}$ is invoked to "throw" the pendubot to a neighborhood of $\mathcal{A}_u$. At about $t = 7.5$, the state $\mathbf{x}$ is such that $W_u(\mathbf{x}) \leq c_u$, q jumps to -3 and the links are caught by κ_u, and consequently, $\mathcal{A}_u$ locally stabilized.

Figure 7(a) shows a planar plot of this same experiment. Each subfigure of Figure 7(a) shows the experiment in a different controller mode q. The blue $*$ is the point where the pendubot initially start and the solid red curve is the path they follow while the controller, corresponding to the current mode, is applied to the system. For aiding the visualization of these results, we included the blue dotted curve corresponding to the trajectories generated by the previous controller to the current mode. The black $\times$ marks represent the equilibrium points of the pendubot system (projected to the plane (ϕ_1, ϕ_2)).

Figure 7(b) demonstrates another representation of the same experiment that shows snapshots of the pendubot links as a function of time for each of the control modes determined by q. The red lines represent link 1 while the blue line represent link 2.

B. Experiment 2

Figure 8 depicts an experiment that demonstrates the pendubot's ability to start in the equilibrium point of $\mathcal{A}_{ur}$ and be stabilized to $\mathcal{A}_u$. Starting from a small neighborhood of

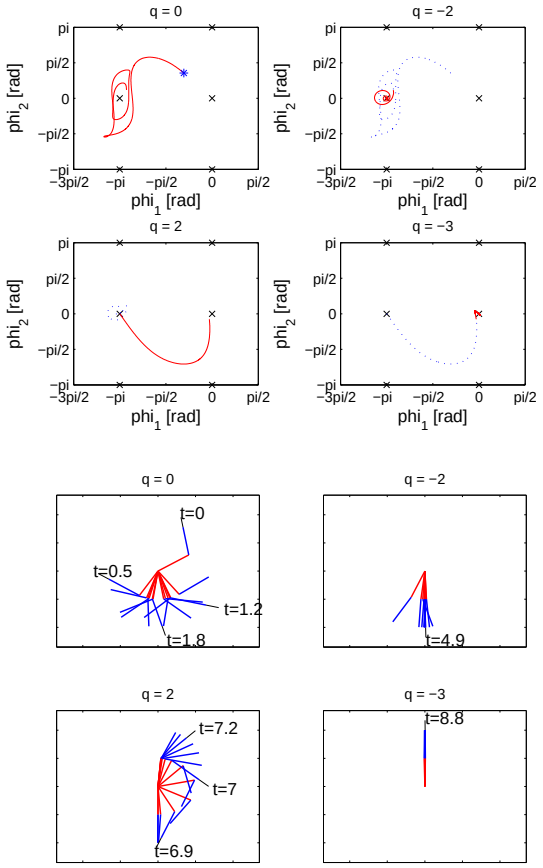

Fig. 7. Results from Experiment 1. Top 4x4 array depicts a planar plot of the pendubot trajectories: initial condition (blue $*$), the trajectory path for the current controller (solid red curve), trajectory path for the previous controller (dotted blue curve), and equilibrium points (black $\times$ marks). Bottom 4x4 array depicts snap shots of link 1 (red line) and link 2 (blue line) from a frontal view while the different controllers are applied. In all three figures, it can be seen that when $q = 0$ (i.e. applying κ_e) the trajectories approach $\mathcal{A}_r$ during the recovery mode. When $q = -2$ (i.e. applying κ_r), the system is in catch mode and the trajectories approach a very small neighborhood of $\mathcal{A}_r$. When $q = 2$ (i.e. applying $\alpha_{r\to u}$), the system is in throw mode and the trajectories move towards $\mathcal{A}_u$. Finally, again in catch mode but with $q = -3$ (i.e. applying κ_u), the trajectories approach a very small neighborhood of $\mathcal{A}_u$.

$\mathcal{A}_{ur}$, the controller jumps to throw mode bringing $\mathbf{x}$ towards $\mathcal{A}_r$. A "catch" is performed (at around 1.5 $sec.$) to bring $\mathbf{x}$ to a neighborhood of $\mathcal{A}_r$. A "throw" is performed (at around 4.5 $sec.$) from the resting configuration $\mathcal{A}_r$ to a neighborhood of the upright configuration $\mathcal{A}_u$. Finally, another "catch" performed (at around 5.5 $sec.$) steers $\mathbf{x}$ to the $\mathcal{A}_r$.

C. Experiment 3

Figure 9 depicts an experiment that demonstrates the pendubot's robustness to small and large disturbances while in the upright position. In this experiment, the pendubot starts at an arbitrary point not in neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. The same sequence of controllers are applied as in Experiment 1 to swing up the pendubot, which takes about 8 $sec.$ Three small perturbations are applied while $\mathbf{x}$ in a neighborhood of $\mathcal{A}_u$ (between $t = 11$ and $t = 17$

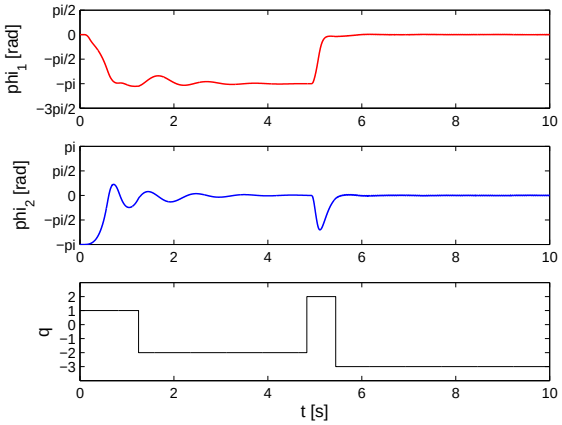

Fig. 8. Experiment 2: results using the hybrid control strategy to swing up the pendubot from a small neighborhood of $\mathcal{A}_{ur}$. Initial conditions: $x^0 = [0, 0, pi, 0]^\mathsf{T}$, $q^0 = 0$, $p^0 = 1$, $\tau^0 = 0$. Starting in recovery mode, the controller jumps to throw mode and brings $\mathbf{x}$ to a neighborhood of $\mathcal{A}_r$. A "catch" is performed (at around 1.5 $sec.$) to bring $\mathbf{x}$ to $\mathcal{A}_r$. A "throw" is performed (at around 4.5 $sec.$) from nearby the resting configuration $\mathcal{A}_r$ to a neighborhood of the upright configuration $\mathcal{A}_u$. Finally, another "catch" performed (at around 5.5 $sec.$) steers $\mathbf{x}$ to $\mathcal{A}_r$.

sec.). The plots in Figure 9 show that the system recovers and returns and stabilizes the links back to $\mathcal{A}_u$. At around 18 sec, a large disturbance is applied. At this event, the local stabilizer κ_u is not able to reject the disturbance and the system jumps to $q = 0$ since $W_u \geq \overline{c}_u$. From such condition, a 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 $sec.$).

D. Experiment 4

Figure 10 depicts an experiment that demonstrates the pendubot's robustness to a large disturbances during a "throw". In this experiment the pendubot starts at a random initial condition, a point not in neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. The same sequence of controllers are applied as were applied in Experiment 1 to swing up the pendubot. The "throw" performed (at around $5.5sec.$) from the resting configuration $\mathcal{A}_r$ is interrupted by a disturbance which obstructs the pendubot from reaching a neighborhood of the upright configuration $\mathcal{A}_u$. However, the hybrid controller is capable to detect that the throw failed (by using the timer state) and 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 $\mathbf{x}$ to $\mathcal{A}_u$. Since throw are an open-loop maneuver, the recovery feature of our hybrid control algorithm under perturbations is needed to have a robust closed-loop system.

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

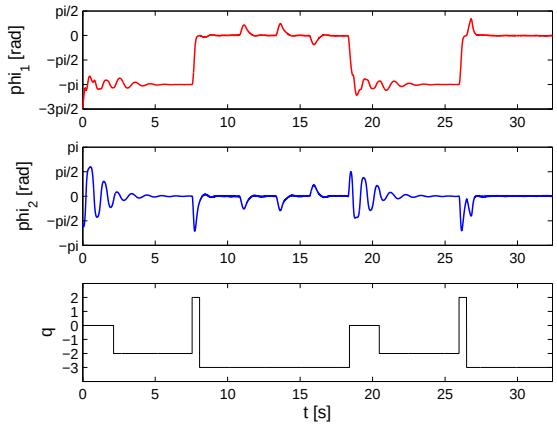

Fig. 9. Experiment 3: results using the hybrid control strategy to swing up the pendubot under the presence of disturbances. Initial conditions: $x^0 = [-4.66, 10.82, -1.76, -7.68]^\mathsf{T}$, $q^0 = 0$, $p^0 = 1$, $\tau^0 = 0$. The figure depicts: pendubot angles ϕ_1 (red), ϕ_2 (blue), q (black). Starting in recovery mode to bring $\mathbf{x}$ towards $\mathcal{A}_r$, a "catch" is performed (at around 2 $sec.$) to bring $\mathbf{x}$ to a point nearby $\mathcal{A}_r$. A "throw" is performed (at around 6.5 $sec.$) from the resting configuration $\mathcal{A}_r$ to a neighborhood of the upright configuration $\mathcal{A}_u$. A jump to the local stabilizer κ_u (at around 7.5 $sec.$) steers $\mathbf{x}$ to $\mathcal{A}_u$. While $\mathbf{x}$ is at $\mathcal{A}_u$, three small perturbations are applied (between $t = 11$ and $t = 17$ sec.). The plot shows that the system can recover and return back to the swing-up configuration. When a large disturbance is applied (at around 18 sec.), the system recovers from being destabilized by performing a successful throw-catch sequence: first takes $\mathbf{x}$ to nearby $\mathcal{A}_r$ (at around 25 $sec.$) followed by a throw-and-catch that stabilizes $\mathcal{A}_u$ (at around 27 $sec.$).

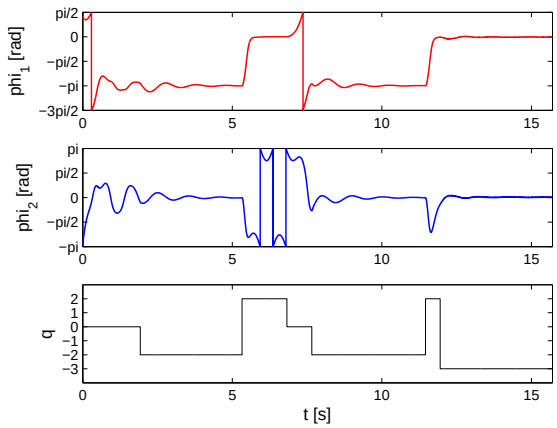

Fig. 10. Experiment 4: results using the hybrid control strategy to stabilize the pendubot from a random initial position, which is a point not in the neighborhood of $\mathcal{A}_r \cup \mathcal{A}_u \cup \mathcal{A}_{ur} \cup \mathcal{A}_{ru}$. Initial conditions: $x^0 = [1.30, -6.52, -2.99, 17.38]^\mathsf{T}$, $q^0 = 0, p^0 = 1, \tau^0 = 0$. The figure depicts: pendubot angles ϕ_1 (red), ϕ_2 (blue), q (black). From initially starting in recovery mode to bring $\mathbf{x}$ towards $\mathcal{A}_r$, a "catch" is performed (at around 2sec.) to bring $\mathbf{x}$ to $\mathcal{A}_r$. A "throw" is performed (at around 5.5 sec.) from the resting configuration $\mathcal{A}_r$. During the "throw" a disturbance obstructs the pendubot from reaching a neighborhood of the upright configuration $\mathcal{A}_u$. Recovery from this disturbances is done by attempting the throw again (at around 11 $sec.$).

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

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

[3] Isabelle Fantoni, Rogelio Lozano, and Mark W. Spong. Energy based control of the pendubot. *IEEE Transactions Automatic Control*, 45:725–729, April 2000.

[4] 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.

[5] Ricardo G. Sanfelice and Andrew 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.