A New Simple Model of Balancing in the Plane

Roy Featherstone

Abstract This paper presents a new model of the dynamics of balancing in the plane, in which the essential parameters of the robot's balancing behaviour are reduced to just two numbers, both of which are simple functions of basic physical properties of the robot mechanism. A third number describes the effect of other movements on the robot's balance. Given this model, a high-performance balance controller can be constructed as a simple four-term control law with gains that are trivial functions of the two model parameters and a single value chosen by the user that determines the overall speed of balancing. The model is first developed for a double pendulum, and then extended to a general planar mechanism. Simulation results are presented showing the controller's performance at following commanded motion trajectories while simultaneously maintaining the robot's balance.

1 Introduction

This paper considers the problem of a planar robot that is actively balancing on a single point of support while simultaneously executing motion commands. In particular, the same motion freedom that is used for balancing is also subject to motion commands. The robot is therefore overloaded in the sense that the number of task variables to be controlled exceeds the number of actuator variables. Such overloading is physically possible, and is routinely exhibited by circus performers and the like, as well as by inverted pendulum robots [8] and wheeled robots that use the same motion freedom both for balancing and for transport [4, 10].

The main contribution of this paper is a new model of the plant (i.e., the robot mechanism) in which the essential features of the robot's balancing behaviour have been reduced to just two numbers. A third number summarizes the disturbance

Roy Featherstone
Dept. Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, Genova 16163, Italy
e-mail: roy.featherstone at iit dot it

caused by other movements being performed by the robot. The model is obtained by exploiting a property of joint-space momentum variables. The advantages of this model are: (1) it is exceptionally simple; (2) it applies to general planar robots, including robots with kinematic loops; (3) it takes into account the effect of other movements of the robot (i.e., movements for accomplishing tasks other than balancing); (4) the model parameters have a clear physical meaning that is easy to understand; (5) they can be computed efficiently using standard dynamics algorithms; and (6) a high-performance balance controller is easily obtained by a simple feedback control law acting directly on the new plant model.

A second contribution is the new balance controller derived from the plant model. It resembles the one presented in [1, 3], and shares its robustness to effects such as torque limits, modelling errors and slippage at the point of support. However, it is simpler, and it can easily be applied to a general planar robot. It differs from the typical approach to balance control in the literature, as exemplified by [7, 9, 13], in that it is a four-term controller using full state feedback, rather than a three-term output-zeroing controller with a one-dimensional zero dynamics. Note that the great majority of literature in this area is actually on swing-up control (e.g. [11, 12]) which is not considered here. The paper concludes with some simulation results showing the performance of the new controller at balancing an inverted triple pendulum while simultaneously following a variety of motion commands.

2 The New Model

A fundamental aspect of balancing is that the controller must control more state variables than the available controls. To understand how this can be done, consider the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u)$, in which $\boldsymbol{x}$ is a vector of state variables and u is a control input. (Bold letters denote vectors.) If $\boldsymbol{x}$ has the property that $x_{i+1} = \dot{x}_i$ for every i, then any control policy that successfully controls x_1 has the side-effect of controlling all of the other elements of $\boldsymbol{x}$. Furthermore, the condition $x_{i+1} = \dot{x}_i$ is sufficient but not necessary, and can be relaxed to some extent. Balancing is an activity that can be accomplished in this way; and the new model described here is essentially a good choice of $\boldsymbol{x}$, having a simple function f, which allows balancing to be achieved using a simple control law for u.

Figure 1 shows a planar 2R mechanism representing an inverted double pendulum. Joint 1 is passive and represents the point contact between the foot of the mechanism and a supporting surface (the ground). It is assumed that the foot neither slips nor loses contact with the ground. The state variables of this robot are q_1, q_2, $\dot{q}_1$ and $\dot{q}_2$. The total mass of the robot is m; the coordinates of its centre of mass (CoM) relative to the support point are c_x and c_y; and it is assumed that the support point is stationary, i.e., it is not a rolling contact. The equation of motion of the robot is

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix}, \tag{1}$$

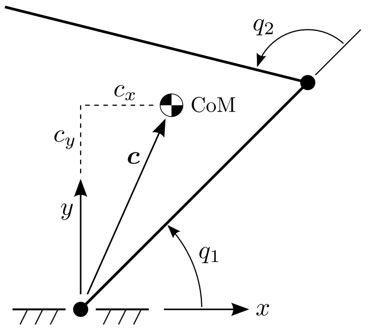

Fig. 1 Planar 2R robot mechanism representing an inverted double pendulum actuated at joint 2

where H_{ij} are elements of the joint-space inertia matrix, C_i are elements of the bias vector containing Coriolis, centrifugal and gravitational terms, $\ddot{q}_i$ are the joint accelerations, and τ_2 is the torque at joint 2. The conditions for the robot to be in a balanced position are: $c_x = 0$, $\dot{q}_1 = 0$ and $\dot{q}_2 = 0$. The robot is also subject to the position command $q_2 = q_\mathrm{c}$, where q_c is an input to the controller.

Any mechanism that balances on a single point has the following special property, which is central to the activity of balancing: the only force that can exert a moment about the support point is gravity. If we define L to be the total angular momentum of the robot about the support point then we find that

$$\dot{L} = -mgc_x \,, \tag{2}$$

where g is the magnitude of gravitational acceleration (a positive number). This equation implies

$$\ddot{L} = -mg\dot{c}_x \tag{3}$$

and

$$\dddot{L} = -mg\ddot{c}_x \,. \tag{4}$$

We also have

$$L = p_1 = H_{11}\dot{q}_1 + H_{12}\dot{q}_2 \,, \tag{5}$$

which follows from a special property of joint-space momentum that is proved in the appendix: if p_i is the momentum variable of joint i then, by definition, $p_i = \sum_j H_{ij}\dot{q}_j$; but if the mechanism is a kinematic tree then p_i is also the component in the direction of motion of joint i of the total momentum of the subtree beginning at body i. As the whole robot rotates about joint 1, it follows that p_1 is the total angular momentum of the robot about the support point, hence $p_1 = L$.

Observe that $\dot{L}$ is simply a constant multiple of c_x, and that L and $\ddot{L}$ are both linear functions of the robot's velocity, implying that the condition $L = \ddot{L} = 0$ is equivalent to $\dot{q}_1 = \dot{q}_2 = 0$. So the three conditions for balance can be written as

$$L = \dot{L} = \ddot{L} = 0 \,. \tag{6}$$

Thus, any controller that successfully drives L to zero will cause the robot to balance, but will not necessarily bring q_2 to the commanded angle.

We now introduce a fictitious extra joint between joint 1 and the base, which is a prismatic joint acting in the x direction. To preserve the numbering of the existing joints, the extra joint is called joint 0. This joint never moves, and therefore never has any effect on the dynamics of the robot. Its purpose is to increase the number of coefficients in the equation of motion, which now reads

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ \tau_2 \end{bmatrix} . \tag{7}$$

The position and velocity variables of joint 0 are always zero, and τ_0 takes whatever value is necessary to ensure that $\ddot{q}_0 = 0$. The reason for adding this joint is that the special property of joint-space momentum, which we used earlier to deduce that $p_1 = L$, also implies that p_0 is the linear momentum of the whole robot in the x direction. So $p_0 = m\dot{c}_x$. With the extra coefficients in Eq. 7 we can write

$$p_0 = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 = m\dot{c}_x = -\ddot{L}/g \, , \tag{8}$$

so that we now have a pair of linear equations relating L and $\ddot{L}$ to the two joint velocities:

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{9}$$

Solving this equation for $\dot{q}_2$ gives

$$\dot{q}_2 = Y_1 L + Y_2 \ddot{L} \, , \tag{10}$$

where

$$Y_1 = \frac{H_{01}}{D} \, , \qquad Y_2 = \frac{H_{11}}{gD} \tag{11}$$

and

$$D = H_{12}H_{01} - H_{11}H_{02} \, . \tag{12}$$

Clearly, this only works if $D \neq 0$. The physical significance of $D = 0$ is explained below. From a control point of view, a problem also arises if $Y_1 = 0$, and this too is discussed below.

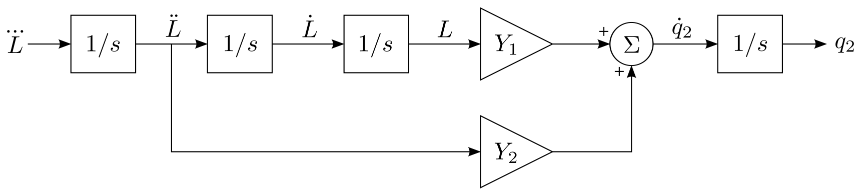

Fig. 2 New plant model for balancing

We now have all the component parts of the new plant model, which is shown in Fig. 2 in the form of a block diagram. The state variables are q_2, L, $\dot{L}$ and $\ddot{L}$, which replace the original state variables. As will be shown in the next section, a simple feedback control law closed around this plant can make q_2 follow a commanded trajectory while maintaining the robot's balance. To be more accurate, what really happens is that the control law tips the robot slightly off balance so that the necessary balance recovery movement just happens to make q_2 follow the commanded trajectory. Once q_2 has reached its final position, the other state variables settle to zero, thereby satisfying the conditions for balance in Eq. 6.

Observe that the new plant model has only two parameters: the two gains Y_1 and Y_2. These gains are calculated directly from the elements of the joint-space inertia matrix in Eq. 7, which in turn can be calculated using any standard method for calculating the joint-space inertia matrix of a robot. Thus, no special code is needed to calculate the model parameters.

Physical Meaning of Y_1 and Y_2

The two gains Y_1 and Y_2 are related in a simple way to two physical properties of the mechanism: the natural time constant of toppling and the linear velocity gain [6]. The former quantifies the rate at which the robot begins to fall in the absence of movement of the actuated joint. The latter measures the degree to which motion of the actuated joint influences the motion of the CoM.

If there is no movement in the actuated joint then the robot behaves as if it were a single rigid body, and its motion is governed by the equation of motion of a simple pendulum:

$$I\ddot{\theta} = mgc(\cos(\theta_0) - \cos(\theta)) \tag{13}$$

where I is the rotational inertia of the robot about the support point, $c = |\boldsymbol{c}|$ is the distance between the CoM and the support point, $\theta = \tan^{-1}(c_y/c_x)$ is the angle of the CoM from the x axis, and the term $mgc\cos(\theta_0)$ is a hypothetical constant torque acting at the support point, which serves to make θ_0 an equilibrium point of the pendulum. Linearizing this equation about θ_0, and defining $\phi = \theta - \theta_0$, results in the following equation:

$$I\ddot{\phi} = mgc_y\phi\,, \tag{14}$$

which has solutions of the form

$$\phi = A\mathrm{e}^{t/T_\mathrm{c}} + B\mathrm{e}^{-t/T_\mathrm{c}} \tag{15}$$

where A and B are constants depending on the initial conditions, and T_c is the natural time constant of the pendulum, given by

$$T_\mathrm{c}^2 = \frac{I}{mgc_y}\,. \tag{16}$$

If $c_y > 0$ then T_c is real and Eq. 15 contains both a rising and a decaying exponential. This is characteristic of an unstable equilibrium. If $c_y < 0$ then T_c is imaginary and Eq. 15 is a combination of sines and cosines, which is characteristic of a stable equilibrium. But if $c_y = 0$ then we are at the boundary between stable and unstable equilibrium and T_c is unbounded. As we are considering the problem of a robot balancing on a supporting surface, it is reasonable to assume $c_y > 0$.

From the definition of the joint-space inertia matrix [5, §6.2] we have $H_{01} = \boldsymbol{s}_0^\mathrm{T} \boldsymbol{I}_0^\mathrm{c} \boldsymbol{s}_1$ and $H_{11} = \boldsymbol{s}_1^\mathrm{T} \boldsymbol{I}_1^\mathrm{c} \boldsymbol{s}_1$, where $\boldsymbol{s}_0 = [0\ 1\ 0]^\mathrm{T}$, $\boldsymbol{s}_1 = [1\ 0\ 0]^\mathrm{T}$ and

$$\boldsymbol{I}_0^\mathrm{c} = \boldsymbol{I}_1^\mathrm{c} = \begin{bmatrix} I & -mc_y & mc_x \\ -mc_y & m & 0 \\ mc_x & 0 & m \end{bmatrix} \tag{17}$$

(planar vectors and matrices—see [5, §2.16]). It therefore follows that $H_{01} = -mc_y$ and $H_{11} = I$, implying that

$$T_\mathrm{c}^2 = \frac{-H_{11}}{gH_{01}} \,. \tag{18}$$

On comparing this with Eq. 11 it can be seen that

$$T_\mathrm{c}^2 = \frac{-Y_2}{Y_1} \,. \tag{19}$$

The linear velocity gain of a robot mechanism, G_v, as defined in [6], is the ratio of a change in the horizontal velocity of the CoM to the change in velocity of the joint (or combination of joints) that is being used to manipulate the CoM. For the robot in Fig. 1 the velocity gain is

$$G_\mathrm{v} = \frac{\varDelta \dot{c}_x}{\varDelta \dot{q}_2} \,, \tag{20}$$

where both velocity changes are caused by an impulse about joint 2. The value of G_v can be worked out via the impulsive equation of motion derived from Eq. 7:

$$\begin{bmatrix} \iota_0 \\ 0 \\ \iota_2 \end{bmatrix} = \begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \varDelta \dot{q}_1 \\ \varDelta \dot{q}_2 \end{bmatrix} \,, \tag{21}$$

where ι_2 is an arbitrary nonzero impulse. Solving this equation for ι_0 gives

$$\begin{aligned} \iota_0 &= H_{01} \varDelta \dot{q}_1 + H_{02} \varDelta \dot{q}_2 \\ &= \big(H_{02} - \frac{H_{01} H_{12}}{H_{11}} \big) \varDelta \dot{q}_2 = \frac{-D}{H_{11}} \varDelta \dot{q}_2 \,. \end{aligned} \tag{22}$$

But ι_0 is the ground-reaction impulse in the x direction, which is the step change in horizontal momentum of the whole robot; so we also have $\iota_0 = m \varDelta \dot{c}_x$, and the velocity gain is therefore

$$G_\mathrm{v} = \frac{\varDelta \dot{c}_x}{\varDelta \dot{q}_2} = \frac{\iota_0}{m \varDelta \dot{q}_2} = \frac{-D}{mH_{11}} . \tag{23}$$

The two plant gains can now be written in terms of T_c and G_v as follows:

$$Y_1 = \frac{1}{mgT_\mathrm{c}^2 G_\mathrm{v}}\,, \qquad Y_2 = \frac{-1}{mgG_\mathrm{v}}\,, \tag{24}$$

and another interesting formula for Y_1 is

$$Y_1 = \frac{c_y}{IG_\mathrm{v}}\,. \tag{25}$$

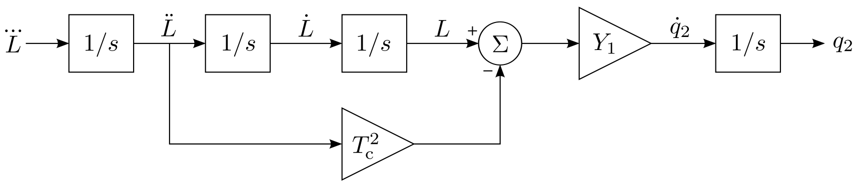

Fig. 3 Alternative version of new plant model for balancing

Equation 19 suggests a small modification to the plant model in Fig. 2, in which Y_2 is replaced with T_c^2 as shown in Fig. 3. In this version of the model, it can be seen that everything to the left of Y_1 is concerned with the balancing motion of the robot, while Y_1 describes how the balancing motion affects joint 2. It was mentioned earlier that the balance controller works by tipping the robot slightly off balance, so that the corrective motion causes q_2 to follow the commanded trajectory. The model in Fig. 3 makes this idea a little clearer.

We are now in a position to explain the physical significance of the conditions $D \neq 0$, which is required by the plant model, and $Y_1 \neq 0$, which is required by the control law in the next section. $D \neq 0$ is equivalent to $G_\mathrm{v} \neq 0$, and it is the condition for joint 2 to have an effect on the horizontal motion of the CoM. If $D = 0$ in some particular configuration then it is physically impossible for the robot to balance itself in that configuration. $Y_1 = 0$ occurs when $c_y = 0$, which is on the boundary between unstable and stable equilibrium. A similar analysis appears in [1, 3].

3 The Balance Controller

The new plant model is interesting in its own right, but its usefulness lies in the simplicity of the balance controller and the ease with which it can be designed and implemented. Consider the following four-term control law:

$$\dddot{L} = k_{dd}(\ddot{L} - \ddot{L}_\mathrm{c}) + k_d(\dot{L} - \dot{L}_\mathrm{c}) + k_L(L - L_\mathrm{c}) + k_q(q_2 - q_\mathrm{c})\,. \tag{26}$$

When the plant in Fig. 2 is subjected to this control law, the resulting closed-loop equation of motion is

$$\begin{bmatrix} \dddot{L} \\ \ddot{L} \\ \dot{L} \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} k_{dd} & k_d & k_L & k_q \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2 & 0 & Y_1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} - \begin{bmatrix} k_{dd}\ddot{L}_\mathrm{c} + k_d\dot{L}_\mathrm{c} + k_L L_\mathrm{c} + k_q q_\mathrm{c} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad (27)$$

and the characteristic equation of the coefficient matrix is

$$\lambda^4 - k_{dd}\lambda^3 - (k_d + k_q Y_2)\lambda^2 - k_L\lambda - k_q Y_1 = 0\,. \qquad (28)$$

The simplest way to choose the gains is by pole placement. As the speed of balancing is determined mainly by the slowest pole, a sensible approach is to place all of the poles at a point $-p$ on the negative real axis, the value of p being chosen by the user, and choose the gains to make Eq. 28 match the polynomial

$$(\lambda + p)^4 = \lambda^4 + 4p\lambda^3 + 6p^2\lambda^2 + 4p^3\lambda + p^4 = 0\,. \qquad (29)$$

The resulting gains are

$$\begin{aligned} k_{dd} &= -4p & k_L &= -4p^3 \\ k_d &= -6p^2 + p^4 Y_2/Y_1 & k_q &= -p^4/Y_1\,. \end{aligned} \qquad (30)$$

Clearly, another polynomial could be used in place of Eq. 29. The choice of p is not critical, but also not arbitrary: if it is too small then balancing happens too slowly, and if it is too large then the robot overshoots too much. A graph illustrating this effect can be found in [1, p. 37]. Simulation studies suggest that a value around 1.2 to 1.5 times $1/T_\mathrm{c}$ is about right.

It can be seen from Eq. 30 that this choice of gains is not possible if $Y_1 = 0$. However, this problem is unavoidable because Y_1 appears in the constant term of Eq. 28, so if $Y_1 = 0$ then $\lambda = 0$ is always a root of the characteristic equation regardless of the choice of gains.

The input q_c in Eq. 26 specifies the trajectory that q_2 is being commanded to follow. It can be arbitrary in the sense of not being required to have any particular algebraic form. However, a sufficiently wild or pathological command will cause the robot to fall over. Simulation studies suggest that the most likely cause of failure is if the command makes the robot enter a region of configuration space where the velocity gain is close to zero.

The inputs L_c, $\dot{L}_\mathrm{c}$ and $\ddot{L}_\mathrm{c}$ in Eq. 26 help to improve the tracking accuracy of time-varying trajectories. The simplest choice for these variables is to set them to zero. In this case, the balance controller converges accurately to q_c when it is constant, but does not track accurately when q_c is changing. Nonzero values can improve the tracking accuracy. For example, setting

$$L_\mathrm{c} = \frac{\dot{q}_\mathrm{c}}{Y_1} \qquad (31)$$

produces accurate tracking of linear ramps (constant $\dot{q}_\mathrm{c}$). (L_d in [1, 3] achieves the same effect.) Additionally setting

$$\dot{L}_c = \frac{\ddot{q}_c}{Y_1} - \frac{\dot{Y}_1}{Y_1} L_c \tag{32}$$

achieves accurate tracking of parabolic curves (constant $\ddot{q}_c$). However, the improved tracking comes at the expense of increased overshoots and a tendency to over-react to the high-frequency component of the command signal.

The value computed by Eq. 26 is $\dddot{L}$, but the output of the control system has to be either a torque command or an acceleration command for joint 2; that is, either τ_2 or $\ddot{q}_2$. These quantities are computed as follows. First, from Eq. 4 we have $\ddot{L} = -mg\ddot{c}_x$; but $m\ddot{c}_x$ is the x component of the ground reaction force acting on the robot, which is τ_0. So $\ddot{L} = -g\tau_0$. Substituting this into Eq. 7 and rearranging to put all of the unknowns into a single vector produces the equation

$$\begin{bmatrix} 0 & H_{01} & H_{02} \\ 0 & H_{11} & H_{12} \\ -1 & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \tau_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -\ddot{L}/g - C_0 \\ -C_1 \\ -C_2 \end{bmatrix}, \tag{33}$$

which can be solved for both τ_2 and $\ddot{q}_2$.

4 Extension to More General Robots

If a robot has more than one actuated motion freedom then two aspects of the balance problem change: (1) there is now a choice of which motion to use for balancing, and (2) there are now motion freedoms that are separate from the balancing activity, and can be controlled with little regard to the balancing activity. These motions can be designed to lie in the balance null space, which is the space of motions that the robot can make that do not affect c_x. However, it may be easier to design these motions to have very little effect on c_x rather than no effect at all.

Let us now replace the double pendulum with a general planar mechanism, retaining only the fictitious prismatic joint and the passive revolute joint that models the contact with the ground. The rest of the mechanism is assumed to be fully actuated, and it may contain kinematic loops. Let $\boldsymbol{y} = [y_0 \; y_1 \; y_2 \; \boldsymbol{y}_3^{\mathrm{T}}]^{\mathrm{T}}$ be a vector of generalized coordinates in which $y_0 = q_0$, $y_1 = q_1$, y_2 is the coordinate expressing the movement to be used for balancing, and $\boldsymbol{y}_3$ is a vector containing the rest of the generalized coordinates. The movement expressed by y_2 can be any desired combination of the actuated joint motions. In effect, y_2 is the variable of a user-defined virtual joint that is a generalization of joint 2 in the previous sections. The equation of motion of this system is

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} & \boldsymbol{H}_{03} \\ H_{10} & H_{11} & H_{12} & \boldsymbol{H}_{13} \\ H_{20} & H_{21} & H_{22} & \boldsymbol{H}_{23} \\ \boldsymbol{H}_{30} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} & \boldsymbol{H}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{y}_2 \\ \ddot{\boldsymbol{y}}_3 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ \boldsymbol{C}_3 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ u_2 \\ \boldsymbol{u}_3 \end{bmatrix}, \tag{34}$$

in which u_2 and $\boldsymbol{u}_3$ are the generalized forces corresponding to y_2 and $\boldsymbol{y}_3$, and H_{ij} are now the elements and submatrices of a generalized inertia matrix. This equation replaces Eq. 7. Equations 2–4 and 6 remain valid, but Eq. 5 becomes

$$L = H_{11}\dot{q}_1 + H_{12}\dot{y}_2 + \boldsymbol{H}_{13}\dot{\boldsymbol{y}}_3 \,. \tag{35}$$

Likewise, Eq. 8 becomes

$$-\ddot{L}/g = H_{01}\dot{q}_1 + H_{02}\dot{y}_2 + \boldsymbol{H}_{03}\dot{\boldsymbol{y}}_3 \,, \tag{36}$$

and so Eq. 9 becomes

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{y}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_{13} \\ -g\boldsymbol{H}_{03} \end{bmatrix} \dot{\boldsymbol{y}}_3 \,. \tag{37}$$

Solving this equation for $\dot{y}_2$ gives

$$\dot{y}_2 = Y_1 L + Y_2 \ddot{L} - \boldsymbol{Y}_3 \dot{\boldsymbol{y}}_3 \,, \tag{38}$$

where Y_1 and Y_2 are as given in Eq. 11, and

$$\boldsymbol{Y}_3 = \frac{\boldsymbol{E}}{D} \tag{39}$$

where

$$\boldsymbol{E} = \boldsymbol{H}_{13} H_{01} - H_{11} \boldsymbol{H}_{03} \tag{40}$$

(cf. Eq. 12). The modified plant model is shown in Fig. 4. Observe that the influence of the non-balance motions is limited to the value of the scalar signal $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. If this signal is zero then these motions have no effect.

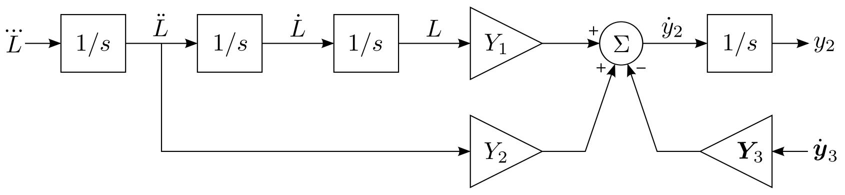

Fig. 4 Modified plant model for a general planar robot

The design of the control system is largely unaffected by $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. In particular, Eq. 28 is unaffected, and the gains are still as given in Eq. 30. However, there is scope to include terms in L_c and $\dot{L}_c$ to counteract the disturbances caused by $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. For example, one could use

$$L_c = \frac{\dot{y}_c}{Y_1} + \frac{\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3}{Y_1} \tag{41}$$

in place of Eq. 31. Simulation studies indicate that this modification successfully compensates for the low-frequency component of the disturbance, but causes the balance controller to over-react to the high-frequency component. A low-pass filter may help in this regard, but it is probably better to design the non-balance motion to lie substantially in the balance null space so that $\boldsymbol{Y}_3$ is close to zero.

Finally, the generalized forces must be calculated and mapped to the actuated joints. The first step is to solve

$$\begin{bmatrix} 0 & \mathbf{0} & H_{01} & H_{02} \\ 0 & \mathbf{0} & H_{11} & H_{12} \\ -1 & \mathbf{0} & H_{21} & H_{22} \\ \mathbf{0} & -\mathbf{1} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} \end{bmatrix} \begin{bmatrix} u_2 \\ \boldsymbol{u}_3 \\ \ddot{q}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 - \boldsymbol{H}_{03}\ddot{\boldsymbol{y}}_3 \\ -C_1 - \boldsymbol{H}_{13}\ddot{\boldsymbol{y}}_3 \\ -C_2 - \boldsymbol{H}_{23}\ddot{\boldsymbol{y}}_3 \\ -\boldsymbol{C}_3 - \boldsymbol{H}_{33}\ddot{\boldsymbol{y}}_3 \end{bmatrix}, \tag{42}$$

which is the generalization of Eq. 33. In this equation, $\ddot{\boldsymbol{y}}_3$ is the desired acceleration calculated by a separate motion control law responsible for $\boldsymbol{y}_3$. The final step is to calculate

$$\boldsymbol{\tau}_{\mathrm{a}} = \boldsymbol{G}^{-\mathrm{T}} \begin{bmatrix} u_2 \\ \boldsymbol{u}_3 \end{bmatrix}, \tag{43}$$

where $\boldsymbol{\tau}_{\mathrm{a}}$ is the vector of force variables at the actuated joints, and $\boldsymbol{G}$ is the matrix (chosen by the user) that maps $[\dot{y}_2 \; \dot{\boldsymbol{y}}_3^{\mathrm{T}}]^{\mathrm{T}}$ to the vector of actuated joint velocities.

5 Simulation and Analysis

This section presents a simulation experiment in which the balance controller makes an inverted triple pendulum perform a variety of manoeuvres while maintaining its balance. A triple pendulum is chosen because it is the simplest mechanism that exhibits all of the phenomena discussed in this paper.

The robot is a 3R planar kinematic chain that moves in the vertical plane. Joint 1 is passive, and the robot is pointing straight up in the configuration $q_1 = q_2 = q_3 = 0$. The link lengths are $0.2\mathrm{m}$, $0.25\mathrm{m}$ and $0.35\mathrm{m}$, and the masses are $0.7\mathrm{kg}$, $0.5\mathrm{kg}$ and $0.3\mathrm{kg}$. The links are modelled as point masses with the mass located at the far end of each link. These are the parameters of a mechanism identified in [6] as being good at balancing.

The control system consists of the balance controller of Section 3, which controls the generalized coordinate y_2, plus a PD position controller with exact inverse dynamics, which controls y_3. The tracking accuracy of the latter is essentially perfect everywhere except where there is a step change in commanded velocity. The balance controller is based on Eq. 26; the gains are as given in Eq. 30 with $p = 7\mathrm{rad/s}$; L_{c} and $\dot{L}_{\mathrm{c}}$ are as given in Eqs. 41 and 32; and $\ddot{L}_{\mathrm{c}} = \dddot{y}_{\mathrm{c}}/Y_1$. The position controller's gains are chosen to put both poles at $14\mathrm{rad/s}$ in order to make the point that the control of variables not used for balancing can take place at a higher frequency than that chosen for the balance controller.

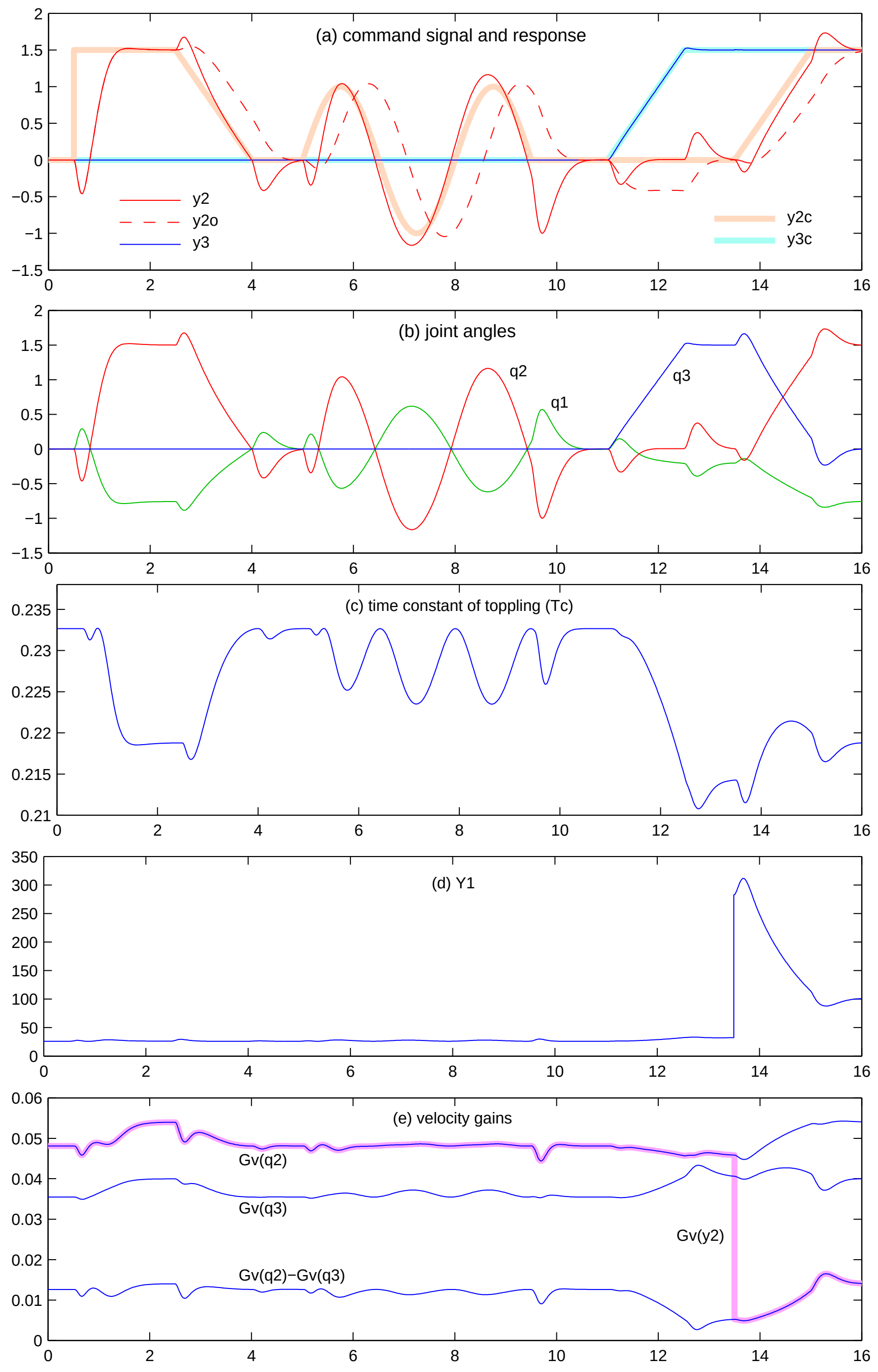

Fig. 5 Simulation results for balancing a triple pendulum (times in seconds, angles in radians)

Figure 5(a) shows the command signals and response, both expressed in generalized coordinates. Times are expressed in seconds and angles in radians. To show the effect of L_c, $\dot{L}_\mathrm{c}$ and $\ddot{L}_\mathrm{c}$, this graph includes a signal 'y2o' showing what the response of the balance controller would have been if $L_\mathrm{c} = \dot{L}_\mathrm{c} = \ddot{L}_\mathrm{c} = 0$. Note that these are relatively large, fast motion commands. Comparable graphs in the literature (e.g. [7]) typically show slower, smaller movements which can be tracked more accurately.

The commands consist of a step, a ramp and a sine wave for y_2 while y_3 is held at zero, a ramp of y_3 while y_2 is held at zero, and finally a ramp of y_2 with y_3 held at 1.5. Up until the final ramp, y_2 and y_3 are defined by $y_2 = q_2$ and $y_3 = q_3$, but then y_2 is redefined to be $y_2 = q_2 - q_3$. So the final ramp involves q_2 ramping from 0 to 1.5 while q_3 ramps from 1.5 to 0. This can be clearly seen in Fig. 5(b), which shows the motion of the robot expressed in joint space.

One obvious feature of Fig. 5(a) is the reverse movements at the beginning of each y_2 manoeuvre and the overshoots at the end. These movements are physically necessary for maintaining the robot's balance. However, the magnitudes of some of these movements (e.g. the one at $9.5\mathrm{s}$) are probably larger than the minimum necessary. Note also that the ramp in y_3 disturbs y_2 only at the beginning and end of the ramp. In the middle portion, the balance controller has successfully compensated for the disturbance caused by this motion, thanks to the term $Y_3\dot{y}_3/Y_1$ in Eq. 41.

Figures 5(c) and 5(d) show the values of T_c and Y_1. Observe that T_c varies in a narrow range from approximately $0.21\mathrm{s}$ to $0.233\mathrm{s}$ even though the robot is making large changes in its configuration. This is a property of the robot mechanism, and will vary from one robot to the next. However, for most balancing robots it will typically be the case that T_c does not vary very much. This suggests that assuming a constant value for T_c could be a good approximation.

For the first 13.5 seconds Y_1 varies in a range from approximately 26 to 33. However, at the point where y_2 is redefined, it jumps to 282, and then rises to 311 and drops to 88 over the course of the final ramp and its overshoot. So for the first $13.5\mathrm{s}$ the plant model is only slightly nonlinear, with the two gains varying in a narrow range, but then the situation changes when y_2 is redefined.

The explanation can be found in Fig. 5(e), which plots the velocity gains of joints 2 and 3 along with their difference, which is the velocity gain of the motion freedom $q_2 - q_3$ [6]. For the first $13.5\mathrm{s}$ $G_\mathrm{v}(y_2) = G_\mathrm{v}(q_2)$; but then y_2 is redefined, and for the remaining time $G_\mathrm{v}(y_2) = G_\mathrm{v}(q_2) - G_\mathrm{v}(q_3)$. As $G_\mathrm{v}(y_2)$ appears in the denominator of Eq. 24, this accounts for the large change in Y_1.

So from this brief analysis we can conclude the following: the robot is generally well-behaved, and the plant model is only slightly nonlinear, up until the beginning of the final ramp. But then the balance controller is given an especially bad new definition of y_2: a motion that has almost no effect on the CoM (i.e., a velocity gain close to zero). So the final ramp is an especially difficult command to follow, and that is why the controller does not track this ramp as accurately as the first ramp. Without an analysis of the physics of the balancing process, it is not at all obvious why the tracking of the final ramp is not as good as the first.

6 Conclusion

This paper has presented a new model of the physical process of balancing by a general planar robot. The essential parameters of the robot's balancing behaviour are reduced to just two numbers, plus a third number to describe the influence of all other movements on the balancing behaviour. All three numbers can be computed efficiently using standard dynamics algorithms. The model gives rise to a simple balance controller that allows the robot to balance while performing other motions; and simulation results are presented showing the controller making a triple pendulum perform a variety of large, fast movements while maintaining its balance. The controller allows complete freedom in choosing which movements are to be used for balancing and which for other tasks.

As planar balancing is a solved problem, the contribution of this paper is to simplify the problem and its solution without loss of generality, and to present an approach to balancing that appeals more to the physical process of balancing and less to the control theory. Clearly, the ultimate objective is a simpler theory of balancing in 3D, and a first step in that direction appears in [1, 2].

Acknowledgements The work presented here owes much to the work of Morteza Azad as described in [1].

Appendix

This appendix proves the result that $p_i = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{h}_{\nu(i)}$, where p_i and $\boldsymbol{s}_i$ are the momentum variable and axis vector of joint i, and $\boldsymbol{h}_{\nu(i)}$ is the total momentum of the subtree or self-contained subsystem consisting of body i and its descendants. A self-contained subsystem, in this context, is defined to be a subsystem in which every kinematic loop that involves at least one body in the subsystem is entirely contained within the subsystem. In general, $\boldsymbol{s}_i$ and $\boldsymbol{h}_{\nu(i)}$ will be spatial vectors. However, if the whole system is planar then they may instead be planar vectors.

Consider a kinematic tree consisting of N bodies and joints numbered from 1 to N according to a regular numbering scheme. Without loss of generality, we assume that every joint has only a single degree of freedom (DoF), which means that every multi-DoF joint has already been replaced by a kinematically equivalent chain of single-DoF joints connected by massless bodies, and that these extra bodies and joints are already included in N.

Let $\boldsymbol{p}$ and $\dot{\boldsymbol{q}}$ denote the joint-space momentum and velocity vectors of the tree, or the spanning tree if there are kinematic loops. By definition, the two are related by the equation

$$\boldsymbol{p} = \boldsymbol{H} \dot{\boldsymbol{q}} \, , \tag{44}$$

where $\boldsymbol{H}$ is the joint-space inertia matrix. This implies that

$$p_i = \sum_{j=1}^{N} H_{ij} \dot{q}_j \,. \tag{45}$$

The definition of $\boldsymbol{H}$ for a general kinematic tree with single-DoF joints is

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j & \text{if } i \in \nu(j) \\ \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{46}$$

where $\boldsymbol{s}_i$ is the joint axis vector (i.e., joint motion subspace vector) of joint i, $\boldsymbol{I}_i$ is the inertia of body i (spatial or planar as appropriate), $\nu(i)$ is the set of all bodies in the subtree beginning at body i, and $\boldsymbol{I}_{\nu(i)} = \sum_{j\in\nu(i)} \boldsymbol{I}_j$.

Let $\bar{\kappa}(i)$ be the set of all bodies on the path between body i and the base (body 0), excluding both body i and the base, and let $\kappa(i) = \bar{\kappa}(i) \cup \{i\}$ be the same set including body i. If we use the terms ‘ancestor’ and ‘descendant’ in an inclusive sense, meaning that body i is both an ancestor and a descendant of itself, and use the term ‘proper ancestor’ in an exclusive sense, then the sets $\nu(i)$, $\kappa(i)$ and $\bar{\kappa}(i)$ can be seen to be the sets of descendants, ancestors and proper ancestors, respectively, of body i. $\kappa(i)$ is also the set of joints on the path between body i and the base.

We now rewrite Eq. 46 as follows:

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j & \text{if } j \in \bar{\kappa}(i) \\ \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{47}$$

which makes it clear that H_{ij} is nonzero only if $j \in \bar{\kappa}(i)$ or $j \in \nu(i)$. Substituting Eq. 47 into Eq. 45 gives

$$\begin{aligned} p_i &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{j\in\bar{\kappa}(i)} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j \dot{q}_j + \sum_{j\in\nu(i)} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{j\in\bar{\kappa}(i)} \sum_{k\in\nu(i)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j + \sum_{j\in\nu(i)} \sum_{k\in\nu(j)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{k\in\nu(i)} \sum_{j\in\bar{\kappa}(i)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j + \sum_{k\in\nu(i)} \sum_{j\in\nu(i)\cap\kappa(k)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{I}_k \sum_{j\in\kappa(k)} \boldsymbol{s}_j \dot{q}_j \,. \end{aligned} \tag{48}$$

The step from the second to the third line works as follows: $\sum_{j\in\nu(i)} \sum_{k\in\nu(j)}$ is the sum over all j, k pairs where j is a descendant of i and k is a descendant of j, whereas $\sum_{k\in\nu(i)} \sum_{j\in\nu(i)\cap\kappa(k)}$ is the sum over all j, k pairs where k is a descendant of i and j is both a descendant of i and an ancestor of k; but these two sets of pairs are the same. The step from the third to the fourth line exploits the fact that $\nu(i) \cap \kappa(k)$ is the set of all ancestors of body k from i onwards, whereas $\bar{\kappa}(i)$ is the set of all ancestors of body k prior to i, so the union of the two sets is $\kappa(k)$.

Now let $\boldsymbol{v}_k$ be the velocity of body k, let $\boldsymbol{h}_k = \boldsymbol{I}_k \boldsymbol{v}_k$ be the momentum of body k, and let $\boldsymbol{h}_{\nu(i)} = \sum_{k\in\nu(i)} \boldsymbol{h}_k$ be the total momentum of the subtree beginning at body i. The velocity of any body in a rigid-body system is the sum of the joint velocities along any one path between that body and the base, so $\boldsymbol{v}_k = \sum_{j\in\kappa(k)} \boldsymbol{s}_j \dot{q}_j$. We can now further simplify Eq. 48 as follows:

$$p_i = \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{I}_k \boldsymbol{v}_k = \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{h}_k = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{h}_{\nu(i)} \,, \tag{49}$$

which establishes the desired result for the case of a kinematic tree. If the system contains kinematic loops then we find that Eq. 49 no longer holds for all joints, but does still hold for any joint that is not involved in any kinematic loop. This is equivalent to the condition stated at the beginning that the subsystem consisting of the bodies in $\nu(i)$ be self-contained.

References

1. M. Azad (2014). Balancing and Hopping Motion Control Algorithms for an Under-actuated Robot. Ph.D. Thesis, The Australian National University, School of Engineering.
2. M. Azad and R. Featherstone (2014). Balancing Control Algorithm for a 3D Under-actuated Robot. *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Systems*, Chicago, IL, Sept. 14–18, pp. 3233–3238.
3. M. Azad and R. Featherstone (2015). Angular momentum based balance controller for an under-actuated planar robot. *Autonomous Robots*, http://dx.doi.org/10.1007/s10514-015-9446-z.
4. Double Robotics (2014). 'Double' telepresence robot. http://www.doublerobotics.com, accessed Aug. 2015.
5. R. Featherstone (2008). *Rigid Body Dynamics Algorithms*. Springer, New York.
6. R. Featherstone (2015). Quantitative Measures of a Robot's Ability to Balance. *Robotics Science & Systems 2015*, Rome, July 13–17, http://www.roboticsproceedings.org/rss11/p26.html, accessed Aug. 2015.
7. J. W. Grizzle, C. H. Moog and C. Chevallereau (2005). Nonlinear Control of Mechanical Systems With an Unactuated Cyclic Variable. *IEEE Trans. Automatic Control*, 50(5):559–576, http://dx.doi.org/10.1109/TAC.2005.847057.
8. J. Hauser and R. M. Murray (1990). Nonlinear Controllers for Non-Integrable Systems: the Acrobot Example. *Proc. American Control Conf.*, Nov. 3–5, pp. 669–671.
9. N. Miyashita, M. Kishikawa and M. Yamakita (2006). 3D Motion Control of 2 Links (5 DOF) Underactuated Manipulator Named AcroBOX. *Proc. American Control Conf.*, Minneapolis, MN, June 14–16, pp. 5614–5619.
10. Segway Inc. (2015). Personal Transporter. http://www.segway.com, accessed Aug. 2015.
11. M. W. Spong (1995). The Swing Up Control Problem for the Acrobot. *IEEE Control Systems Magazine*, 15(1):49–55.
12. X. Xin and M. Kaneda (2007). Swing-Up Control for a 3-DOF Gymnastic Robot With Passive First Joint: Design and Analysis. *IEEE Trans. Robotics*, 23(6):1277–1285.
13. T. Yonemura and M. Yamakita (2004). Swing Up Control of Acrobot Based on Switched Output Functions. *Proc. SICE 2004*, Sapporo, Japan, Aug. 4–6, pp. 1909–1914.