A Simple Model of Balancing in the Plane and a Simple Preview Balance Controller

Roy Featherstone
Dept. Advanced Robotics, Istituto Italiano di Tecnologia
via Morego 30, 16163 Genova, Italy

to appear in IJRR, submitted January 6, 2017

Abstract

This paper presents a new model of the dynamics of a general planar robot balancing on a point 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. This model gives rise to a simple preview balance controller consisting of a four-term control law with easily calculated gains and a reverse-time low-pass filter acting on a preview of the command signal. The filter makes the robot lean in anticipation of future movements. Simulation results are presented showing the balance controller achieving excellent tracking of large fast motion commands while simultaneously maintaining the robot's balance and accurately rejecting disturbances caused by other motions being performed by the robot. The controller is also robust to effects such as actuator saturation and sensor noise.

Keywords: legged robots, planar robots, balance control, preview control, acausal filter, non-minimum-phase behaviour, physics of balancing, leaning in anticipation, angular momentum.

1 Introduction

This paper considers the problem of a general 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 (Hauser and Murray, 1990) and wheeled robots that use the same motion freedom both for balancing and for transport (Double Robotics, 2015; Segway, 2015).

The first 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 caused by other

movements being performed by the robot. 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.

The second contribution is the new balance controller derived from the plant model. It resembles the one presented in Azad (2014); Azad and Featherstone (2016), and shares its robustness to effects such as torque limits, sensor noise and time delays. However, it is simpler, and it applies to general planar robots. Compared with the typical approach to balance control in the control theory literature, as exemplified by Grizzle et al. (2005); Miyashita et al. (2006); Yonemura and Yamakita (2004), the new controller is a four-term controller using full state feedback, rather than a three-term output-zeroing controller with a one-dimensional zero dynamics. Another difference is that the new controller requires only the numeric values of the coefficients of a general equation of motion, such as can be calculated by standard dynamics software, whereas the typical approach is to start with the symbolic equations of motion of a particular robot. Note that the great majority of literature in this area is actually on swing-up control (e.g. Spong (1995); Xin and Kaneda (2007)) which is not considered here.

The third contribution is a simple technique for making the robot lean in anticipation of future command signals. It consists of a first-order low-pass filter that is interposed between the command signal and the input to the balance controller. However, this filter runs backwards in time, starting from a point sufficiently far in the future, so that its output is a function of present and future command signals. The filter's time constant matches the robot's natural rate of toppling, and its immediate effect is to cancel the robot's non-minimum-phase behaviour. The result is a large improvement in tracking accuracy, as well as a large improvement in the balance controller's ability to reject anticipated disturbances caused by the other motions being performed by the robot.

The idea of leaning in anticipation of future movements is not new (Rabbani et al., 2014). More generally, the idea of employing knowledge about the future in a control system is known as *preview control*, and it is already an established technique in robotics for tasks such as walking pattern generation (Kajita et al., 2003) and improving humanoid balance (Ibanez et al., 2012). Model predictive control, as described in Wieber (2006) for example, is also a form of preview control because the optimization problem includes information about future events; and a robot controlled in this way will exhibit some degree of leaning in anticipation as a by-product of the optimization process. However, the idea of making a robot lean in anticipation by means of the low-pass filter described above appears to be new.

This paper is an extended version of Featherstone (2015b), and much of the material presented here appeared originally in that paper. The new materials in this paper are: the technique of leaning in anticipation, the transfer function analysis that leads up to it, new results demonstrating the performance of the controller when combined with leaning in anticipation, and an outline of how the plant model and balance controller can be extended to the problem of balancing in 3D.

The paper is organized as follows. Section 2 compares the proposed new balance

controller with existing methods in the literature. Then Section 3 describes the new model of the physical process of balancing for the special case of a planar inverted double pendulum. Section 4 describes the balance controller and the technique of leaning in anticipation. Then Section 5 presents several simulation results investigating the controller's performance, and compares it with some previously published balance control systems. Then Section 6 extends both the model and the controller to the case of a general planar mechanism, and presents simulation results for a planar triple pendulum. Finally, Section 7 explains briefly how the ideas in this paper can be extended to the case of a general robot balancing on a point in 3D.

1.1 A Note on Balancing

As the term 'balancing' is used in the robotics literature with two distinct meanings, a few words of clarification are required. The kind of balancing considered here arises when a robot balances on a line segment or a point. It is characterized by the presence of a motion freedom over which the robot has no direct control; namely, the freedom to rotate about the support. In this circumstance, the robot is underactuated, and every balanced configuration of the robot is unstable. The task of balancing requires a control system that controls the unactuated freedom(s) indirectly by manipulating the moment of gravity about the support.

In contrast, the kind of balancing that has been most extensively studied in the robotics literature arises when the robot has a polygon of support. In this circumstance, the robot is equivalent to a fixed-base robot and is effectively fully actuated. However, it does have a tipping point, and there are constraints on the possible values of the ground reaction force. A robot with a polygon of support has infinitely many configurations of statically-stable balance. In any one of these configurations, the robot can balance indefinitely simply by doing nothing. However, if movement is required then it is necessary to identify movements that satisfy the constraints on ground reaction force and keep the robot away from its tipping point. This is typically accomplished using a zero-moment-point (ZMP) controller, and such controllers generally work by planning a safe movement and then executing it using a simple trajectory-following controller.

Humanoids are usually designed with large flat feet in order to maximize the size of their polygons of support. However, even a humanoid would have to balance on a point or a line if it wanted to stand on a hump, or on tip toes.

2 Comparison with Existing Methods

The usual approach to balance control begins with the closed-form equations of motion of the robot mechanism. These are a set of symbolic equations that express the force variables explicitly in terms of the state variables, the acceleration variables and the kinematic and inertia parameters of the robot. Examples include Eqs. 1 and 2 in Spong (1995), Eqs. 1 and 2 in Hauser and Murray (1990), Eqs. 1–3 in Yonemura and Yamakita (2004), Eqs. 1–4 in Xin and Kaneda (2007), Eqs. 1 and 2 in Azad and Featherstone (2016) and Eqs. 3.1 and 3.2 in Azad (2014). These equations are then manipulated in various ways in order to arrive at a formula for a feedback control law.

This is a good approach when the mechanism is simple, but it does not scale well to more complicated mechanisms, and it does not generalize to closed-loop mechanisms. For a kinematic tree, the size of the closed-form equations of motion grows with the fourth power of the number of degrees of freedom; and the problem with kinematic loops is that only a few special cases have closed-form solutions. If a mechanism contains a kinematic loop without a closed-form solution then the mechanism as a whole does not have a closed-form equation of motion. None of the works mentioned above considers closed-loop mechanisms; and Grizzle et al. (2005) explicitly rules out kinematic loops near the beginning of the paper.

For these reasons, the new balance controller is developed in a way that requires only the numeric values of the coefficients of the equation of motion. In particular, if the equation is written in the form

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{C}\,, \tag{1}$$

where $\boldsymbol{\tau}$ and $\ddot{\boldsymbol{q}}$ are vectors of generalized forces and accelerations, respectively, then the theory behind the new controller assumes knowledge of only the numeric values of the elements of $\boldsymbol{H}$ and $\boldsymbol{C}$, not their closed-form expressions. This immediately solves the kinematic-loop problem because it allows loop-closure constraints to be incorporated numerically into the equation of motion, which is always possible. It also improves the scalability, because it permits the use of efficient standard algorithms to calculate $\boldsymbol{H}$ and $\boldsymbol{C}$. These algorithms have a computational cost that is linear in the number of elements to be calculated, in the case of a kinematic tree, or at most cubic in the number of open-loop freedoms, in the case of a closed-loop mechanism (Featherstone, 2008).

A second major difference between the new controller and those described in the literature is that the new one does not control the robot directly, but instead controls a new model of the physical process of balancing, treating it as a plant. This new model is both general and exact, and it encapsulates a surprising new result: the balancing behaviour of any planar robot mechanism, no matter how complicated, is described by just two numbers. In effect, the new model extracts the essence of a robot's balancing behaviour from the rest of its dynamics, and expresses it in its simplest form. By choosing to control the behaviour of this model, instead of the original robot, one obtains a simple controller that is concerned only with balancing behaviour, not the detailed dynamics of any particular robot. No other balance controller works in this way.

Another difference is that the command signal is passed through an acausal filter before being fed to the balance controller. The filter is a simple first-order low-pass filter running backwards in time, as mentioned in the introduction, so that the signal received by the controller is a function of both the present value of the command signal and near-future values. The filter is motivated by an analysis of the closed-loop transfer function of the balancing behaviour model, and its primary purpose is to cancel the zero that is responsible for the non-minimum-phase behaviour. However, its effect on the robot is to make it lean in anticipation of future balance disturbances.

This filter greatly improves the tracking ability of the balance controller, and allows it to perform larger, faster movements without falling over than would have been possible if the robot had not leaned in anticipation. For example, in some of the results presented later in this paper, the robot leans almost 0.2 radians in anticipation of movements so large that they throw the robot 0.4 radians in the opposite direction. The filter also reduces the response time, as can be seen in Figure 12, and it allows the controller to use

higher gains than would have been feasible otherwise, so that it can track those larger, faster motion commands.

However, it should be understood that balancing is physically a non-minimum-phase activity, and the robot's response to unanticipated disturbances, including sensor noise, will always be a non-minimum-phase response.

The three items above are the most important differences between the new controller and existing balance controllers described in the literature. Together, they account for the new controller's generality, simplicity and performance. However, there are two more differences worth mentioning.

First, the new controller is designed to be part of a larger control system in which there exists another controller that is handling other aspects of the robot's motion. The only two things that the new controller assumes about this other controller are (1) it has a preview of the movements that the other controller intends to make, so that it can lean in anticipation of the balance disturbances that these movements will cause, and (2) the actual movements made by the other controller accurately follow its intentions. In this respect, the new controller resembles the one in Ibanez et al. (2012). Most other balance controllers are designed to be complete control systems by themselves.

The second difference concerns a technical detail in the way the control law works. The new controller resembles Azad's controller (Azad, 2014; Azad and Featherstone, 2016) in that they both use a four-term control law that implements full state feedback. (They also happen to use the same state variables.) However, several other controllers, such as those described in Grizzle et al. (2005); Miyashita et al. (2006); Yonemura and Yamakita (2004), employ a three-term control law that seeks to control a fictitious output that is a linear combination of the robot's angular momentum about the support with a quantity whose derivative is algebraically related to the angular momentum. This is a mathematically rigorous approach that allows one to make stronger statements about stability than anything that is claimed here about the new controller. However this approach comes with a performance penalty that can be seen in Figure 6 of Grizzle et al. (2005). In this figure, the top two graphs show the fictitious output settling to the command signal in under 10 seconds; but the bottom left graph shows that the robot's joint variables take more than 10 seconds to settle. This effect can be attributed to the zero dynamics of the system, which are provably stable but nevertheless take some time to settle. For comparison, the settling time of the new controller on a robot of similar dimensions is about 20 times faster.

3 A New Model of Balancing Behaviour

This section describes a new model of the physical process of balancing for the special case of a planar 2R mechanism (an inverted double pendulum). It is extended to general planar mechanisms in Section 6, and its extension to 3D is outlined in Section 7. The new model resembles the idea of a template, as described in Full and Koditschek (1999). However, there is an important difference: a template typically describes a simplified mechanical system, whereas the model presented here describes the exact dynamics of balancing expressed in its simplest form. A good example of the former is the spring-loaded inverted pendulum template for hopping and running (Blickhan, 1989; Full and Koditschek, 1999), in which the leg is assumed to be massless. This assumption simplifies

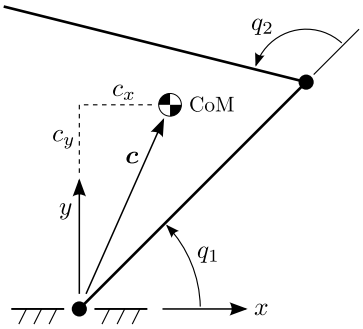

Figure 1: Planar 2R robot mechanism representing an inverted double pendulum actuated at joint 2

the dynamics considerably, but at the expense of ignoring all effects due to the nonzero mass of a physical leg. In contrast, the model presented here expresses the essence of balancing behaviour without ignoring any dynamic effect.

3.1 The New Model

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

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix}, \tag{2}$$

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 signal $q_2 = q_\mathrm{c}(t)$, 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

$$\dot{L} = -mgc_x\,, \tag{3}$$

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

$$\ddot{L} = -mg\dot{c}_x \tag{4}$$

and

$$\dddot{L} = -mg\ddot{c}_x\,. \tag{5}$$

We also have

$$L = p_1 = H_{11}\dot{q}_1 + H_{12}\dot{q}_2\,, \tag{6}$$

which follows from a special property of joint-space momentum that is proved in Appendix B: 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, or if the mechanism is general but joint i does not participate in any kinematic loop, 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 $\dot{q}_1 = \dot{q}_2 = 0$ (assuming linear independence). So the three conditions for balance can be written as

$$L = \dot{L} = \ddot{L} = 0\,. \tag{7}$$

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. (This is not a new result—see, for example, Miyashita et al. (2006, §4).)

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

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ \tau_2 \end{bmatrix}. \tag{8}$$

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. 8 we can write

$$p_0 = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 = m\dot{c}_x = -\ddot{L}/g\,, \tag{9}$$

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

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}. \tag{10}$$

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

$$\dot{q}_2 = Y_1 L + Y_2 \ddot{L}\,, \tag{11}$$

where

$$Y_1(\boldsymbol{q}) = \frac{H_{01}}{D}\,, \qquad Y_2(\boldsymbol{q}) = \frac{H_{11}}{gD} \tag{12}$$

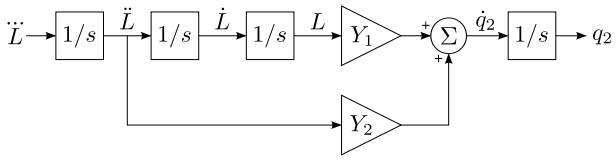

Figure 2: New plant model for balancing

and

$$D = H_{12}H_{01} - H_{11}H_{02}\,. \tag{13}$$

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.

We now have all the component parts of the new model, which is shown in Figure 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 Section 4, a simple feedback control law closed around this model, which serves as the plant from the control system's point of view, 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. 7.

Observe that the new plant model has only two parameters: the two configuration-dependent gains Y_1 and Y_2. These gains are calculated directly from the elements of the joint-space inertia matrix in Eq. 8, 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.

3.2 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 (Featherstone, 2015a). 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:

$$I\ddot{\theta} = mgc(\cos(\theta_0) - \cos(\theta)) \tag{14}$$

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:

$$I\ddot{\phi} = mgc_y\phi\,, \tag{15}$$

which has solutions of the form

$$\phi = A\mathrm{e}^{t/T_\mathrm{c}} + B\mathrm{e}^{-t/T_\mathrm{c}} \tag{16}$$

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

$$T_\mathrm{c}^2 = \frac{I}{mgc_y}\,. \tag{17}$$

If $c_y > 0$ then T_c is real and Eq. 16 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. 16 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 (Featherstone, 2008, §6.2) we have $H_{01} = \boldsymbol{s}_0^\mathrm{T}\boldsymbol{I}_1^\mathrm{c}\boldsymbol{s}_1$ 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

$$\boldsymbol{I}_1^\mathrm{c} = \begin{bmatrix} I & -mc_y & mc_x \\ -mc_y & m & 0 \\ mc_x & 0 & m \end{bmatrix} \tag{18}$$

(planar vectors and matrices—see Featherstone (2008, §2.16)). It therefore follows that $H_{01} = -mc_y$ and $H_{11} = I$, implying that

$$T_\mathrm{c}^2 = \frac{-H_{11}}{gH_{01}}\,. \tag{19}$$

On comparing this with Eq. 12 it can be seen that

$$T_\mathrm{c}^2 = \frac{-Y_2}{Y_1}\,. \tag{20}$$

The linear velocity gain of a robot mechanism, G_v, as defined in Featherstone (2015a), 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 Figure 1 the velocity gain is

$$G_\mathrm{v} = \frac{\Delta\dot{c}_x}{\Delta\dot{q}_2}\,, \tag{21}$$

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. 8:

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

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

$$\begin{aligned} \iota_0 &= H_{01}\Delta\dot{q}_1 + H_{02}\Delta\dot{q}_2 \\ &= \big(H_{02} - \frac{H_{01}H_{12}}{H_{11}}\big)\Delta\dot{q}_2 = \frac{-D}{H_{11}}\Delta\dot{q}_2\,. \end{aligned} \tag{23}$$

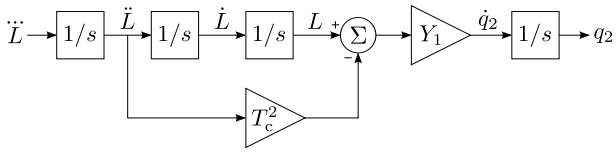

Figure 3: Alternative version of new plant model for balancing

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 $\iota_0 = m\Delta\dot{c}_x$, and the velocity gain is therefore

$$G_{\mathrm{v}} = \frac{\Delta\dot{c}_x}{\Delta\dot{q}_2} = \frac{\iota_0}{m\Delta\dot{q}_2} = \frac{-D}{mH_{11}}. \tag{24}$$

The two plant gains can now be written in terms of T_{c} and G_{v} as follows:

$$Y_1 = \frac{1}{mgT_{\mathrm{c}}^2 G_{\mathrm{v}}}\,, \qquad Y_2 = \frac{-1}{mgG_{\mathrm{v}}}\,, \tag{25}$$

and another interesting formula for Y_1 is

$$Y_1 = \frac{c_y}{IG_{\mathrm{v}}}\,. \tag{26}$$

Equation 20 suggests a small modification to the plant model in Figure 2, in which Y_2 is replaced with T_{c}^2 as shown in Figure 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 Figure 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. Given that the robot lies above a supporting surface located at $y = 0$, we may safely assume that $c_y > 0$, implying $Y_1 \neq 0$. A similar analysis appears in Azad (2014); Azad and Featherstone (2016).

4 The Balance Controller

The new plant model is interesting in its own right, but its usefulness lies in the simplicity and high performance of the balance controller that is suggested by the model, and the ease with which the controller can be designed and implemented. This section describes the controller in stages: first the control law, then an analysis of the transfer function, which leads to the technique of leaning in anticipation.

4.1 Control Law

Given the input, output and state variables of the plant in Figure 2, the obvious strategy to try is full state feedback. Therefore, consider the following four-term control law:

$$\dddot{L} = k_{dd}\ddot{L} + k_d\dot{L} + k_L L + k_q(q_2 - u)\,, \tag{27}$$

where the input u is given by

$$u = q_c + \alpha_1\dot{q}_c + \alpha_2\ddot{q}_c\,. \tag{28}$$

k_{dd}, k_d, k_L and k_q are the feedback gains; inputs q_c, $\dot{q}_c$ and $\ddot{q}_c$ are the position command signal for joint 2 and its derivatives; and α_i are optional feedforward gains that have a similar effect to L_c and $\dot{L}_c$ in Featherstone (2015b) and L_d in Azad (2014) and Azad and Featherstone (2016). In the simplest case, α_i can be set to zero. Alternatively, nonzero values can be used to improve the tracking accuracy and/or cancel selected poles in the transfer function. Equation 28 will be modified in Section 4.3.

When the plant in Figure 2 is subjected to the control law in Eq. 27, the resulting closed-loop equation of motion is

$$\begin{bmatrix} \dddot{L} \\ \ddot{L} \\ \dot{L} \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} k_{dd} & k_d & k_L & k_q \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2 & 0 & Y_1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} - \begin{bmatrix} k_q u \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{29}$$

and the characteristic equation of the coefficient matrix is

$$\lambda^4 - k_{dd}\lambda^3 - (k_d + k_q Y_2)\lambda^2 - k_L\lambda - k_q Y_1 = 0\,. \tag{30}$$

At this point we introduce an approximation. The next step is to linearize the dynamics about the current configuration. However, instead of an exact linearization, we shall use an approximate linearization that is obtained by assuming that Y_1 and Y_2 are constant. This is equivalent to assuming that $\partial Y_i/\partial\dot{L} = \partial Y_i/\partial q_2 = 0$ in the exact linear dynamics. Having made this approximation, the roots of Eq. 30 are the poles of the linearized system.

The simplest way to choose the feedback gains is by pole placement. If λ_1, λ_2, λ_3 and λ_4 are the desired values of the poles, then the corresponding polynomial is

$$\lambda^4 + a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0\,, \tag{31}$$

where

$$\begin{aligned} a_0 &= \lambda_1\lambda_2\lambda_3\lambda_4 \\ a_1 &= -\lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_2\lambda_4 - \lambda_1\lambda_3\lambda_4 - \lambda_2\lambda_3\lambda_4 \\ a_2 &= \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_1\lambda_4 + \lambda_2\lambda_3 + \lambda_2\lambda_4 + \lambda_3\lambda_4 \\ a_3 &= -\lambda_1 - \lambda_2 - \lambda_3 - \lambda_4\,. \end{aligned} \tag{32}$$

The gains are then obtained by matching the coefficients in Eqs. 30 and 31, resulting in

$$\begin{aligned} k_{dd} &= -a_3 \qquad & k_d &= -a_2 + a_0 Y_2/Y_1 \\ k_L &= -a_1 \qquad & k_q &= -a_0/Y_1\,. \end{aligned} \tag{33}$$

A suitable choice of poles is discussed below.

For the linearized system to be stable, it is necessary that every pole has a negative real part. However, it can be seen from Eq. 30 that if $Y_1 = 0$ then $\lambda = 0$ is always a root of the characteristic equation regardless of the choice of gains. So we require $Y_1 \neq 0$ as a condition of stability.

The signal q_c 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 (such as the special trajectories found by Berkemeier and Fearing (1999)). However, it is always possible to find a signal that will cause the robot to fall over. Thus, even if every pole is stable, a sufficiently bad input can always make the robot fall. (An example is given in Section 5.)

The value computed by Eq. 27 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. 5 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 $\dddot{L} = -g\tau_0$. Substituting this into Eq. 8 and rearranging to put all of the unknowns into a single vector produces the equation

$$\begin{bmatrix} 0 & H_{01} & H_{02} \\ 0 & H_{11} & H_{12} \\ -1 & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \tau_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 \\ -C_1 \\ -C_2 \end{bmatrix}, \tag{34}$$

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

4.2 Transfer Function

A linear system that is described in state-space form by the equations

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{35}$$

has a transfer function that can be expressed in the frequency domain as

$$\boldsymbol{y}(s) = \boldsymbol{C}(\mathbf{1}s - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{u}(s) \tag{36}$$

(Anderson and Moore, 1971), where $\mathbf{1}$ denotes an identity matrix of the appropriate size. Applying this formula to the linearized system, as described above, produces the following formula for the transfer function from the input u to the output q_2:

$$q_2(s) = \frac{a_0(1 - T_c^2 s^2)}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\, u(s)\,, \tag{37}$$

assuming that the gains have been set according to Eq. 33. The complete transfer function from q_c to q_2 is therefore

$$q_2(s) = \frac{a_0(1 - T_c^2 s^2)(1 + \alpha_1 s + \alpha_2 s^2)}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\, q_c(s)\,. \tag{38}$$

Thus, the closed-loop system has four poles at frequencies chosen by the designer, two zeros at frequencies determined by the physical properties of the robot, and optional zeros chosen by the designer if one or both of the feedforward gains α_i are nonzero.

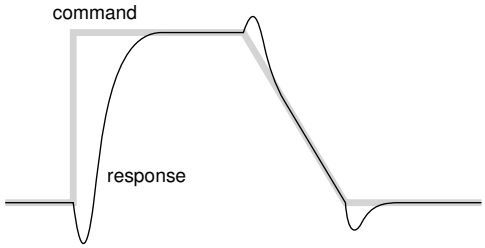

Figure 4: Typical non-minimum-phase behaviour: each change in the command signal provokes an initial response in the opposite direction

The two zeros determined by the mechanism lie at $\pm 1/T_c$. The zero at $-1/T_c$ can be eliminated by setting one of the poles equal to $-1/T_c$. This is a good idea, partly because it simplifies the behaviour of the system, and partly because it reduces the coupling between the conflicting activities of maintaining the robot's balance and following the command signal. (The eigenvector associated with this pole involves only L, $\dot{L}$ and $\ddot{L}$, so it can be regarded as being devoted exclusively to balancing.)

However, the zero at $1/T_c$ is more problematic. This is the zero responsible for the non-minimum-phase behaviour of the system, which degrades the tracking response of the system in the way illustrated in Figure 4. Put simply, each time the command signal changes, the robot has to alter its state of balance before it can respond, and in order to do this the robot has to move briefly in the wrong direction. The magnitudes of these excursions in the wrong direction can be large, especially if the command signal specifies large, fast movements and/or the designer is aiming for high performance by setting the gains high.

The conventional wisdom is that non-minimum-phase behaviour cannot be eliminated, because the corresponding pole is unstable. However, there is a way to do it, which is the subject of the next section.

4.3 Leaning in Anticipation

If one watches how the robot behaves in response to command signals, the impression is that the robot is continually being 'surprised' by changes in the command, and is continually having to make large excursions in order to adjust its state of balance as quickly as possible to suit the new command. This is not how humans behave. We nearly always know what movements we intend to make in the immediate future (i.e. the next 1 or 2 seconds), and our current movements are tweaked in ways that facilitate the movements that we intend to make next (Rabbani et al., 2014).

Consider, for example, the task of pulling open a heavy door. We know that this will require a large horizontal force, and so we begin to lean backwards *slightly in advance* of exerting the force in order to avoid being pulled off balance. This is a good strategy, and a good balance controller ought to be able to replicate it. However, to do so requires that the command signal be modified so that it contains information about the future. This is feasible because a robot's high-level controller, which is involved in activities such as planning, typically does know what movements it intends to make in the immediate future.

So we have two problems to solve: how to eliminate the zero at $1/T_c$, and how to

incorporate information about the future into the command signal. The solution turns out to be very simple: starting sufficiently far in the future, pass the command signal through a first-order low-pass filter with a pole at $-1/T_c$, running *backwards in time* to the current instant. This solves the first problem because a pole at $-1/T_c$ in reverse time becomes a pole at $1/T_c$ in forward time, so the filtered signal already incorporates the correct pole to eliminate the zero at $1/T_c$. And the second problem is solved because the output of the filter is a function of both the current value of q_c and expected future values. Technically, this makes it an acausal filter.

Let q_f denote the filtered command signal, and let us redefine u as follows:

$$u = q_f + \alpha_1 \dot{q}_f + \alpha_2 \ddot{q}_f \tag{39}$$

which replaces Eq. 28. The transfer function from q_c to q_f is $1/(1 - T_c s)$, so the complete transfer function from q_c to q_2 is now

$$q_2(s) = \frac{a_0(1 + T_c s)(1 + \alpha_1 s + \alpha_2 s^2)}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}\, q_c(s)\,, \tag{40}$$

which is no longer a non-minimum-phase system. Furthermore, by setting one of the poles to $-1/T_c$, as mentioned above, the last of the mechanism-dependent zeros is removed, and the transfer function simplifies to three poles and up to two zeros, all of them freely selectable by the control system designer. If the plant were linear, then it would really be true that all dependence on the behaviour of the robot has been eliminated from the transfer function (but not from the total behaviour of the robot, which depends also on the values of L, $\dot{L}$ and $\ddot{L}$). However, the plant is nonlinear, so the robot's dynamics will always have some effect on the transfer function, especially when the robot is making large, fast movements.

Ideally, the acausal filter should start at a point so far into the future as to be indistinguishable from infinity. However, looking ahead by $3T_c$ is enough to get 95% of the desired effect, and looking ahead by $4T_c$ gets 98%.

5 Performance Evaluation

This section presents several simulation experiments aimed at evaluating the tracking performance of the control system, investigating its sensitivity to model errors and limitations in the robot's sensors and actuators, and comparing it against some previously published balance controllers.

The control system was tested on a robot consisting of a flywheel on a stick. The flywheel has a mass of 1kg and a radius of gyration of 0.5m. The stick has a length of 1m and a mass of 1kg which is treated as a point mass located at the mid point of the stick. The robot is balanced when the stick is upright, and the stick is upright when $q_1 = 0$. The plant parameters for this robot are $Y_1 = 4$ and $Y_2 = -0.407747/\cos(q_1)$, so the plant is only slightly nonlinear. The value of Y_2 depends on gravity, which is set at $g = 9.81\mathrm{ms}^{-2}$. Other quantities of interest are $G_v = \cos(q_1)/8$ and $T_c = 1/\sqrt{g\cos(q_1)}$.

The simulations reported in this paper were all performed in Simulink using the integrator ode45 with relative tolerance set at 10^{-6}; and the simulation start time was set at -2s so that all rising exponentials associated with the acausal filter are very close to zero when the simulation starts.

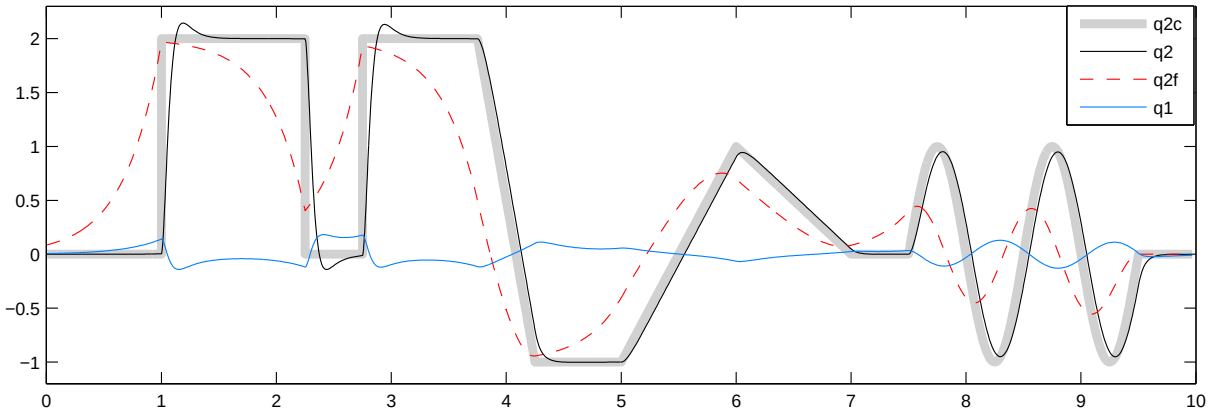

Figure 5: The balance controller's performance on the flywheel-and-stick robot (times in seconds, angles in radians). This movement is shown in Extension 1

5.1 Tracking Accuracy

Figure 5 shows the results of the first simulation experiment. In this experiment, the controller's gains were set as follows: one pole at $-1/T_c$, three poles at -20rad/s and two zeros at -20rad/s, for a theoretical transfer function from q_c to q_2 of $1/(1+0.05s)$. The acausal filter was implemented analytically assuming a constant value for the natural time constant of $T_c = 0.319275$s, which is the value it takes when $q_1 = 0$. As q_1 never exceeds ± 0.2rad, the error in this approximation is never more than 1%.

The graph in Figure 5 shows a sequence of motion commands and the robot's response. The command sequence consists of a step at $t = 1$s, followed by two more steps in quick succession, followed by three linear ramps having slopes of 6, 2 and 1rad/s, followed by a sine wave at 1Hz. The graph also shows the acausally filtered command signal q_f.

Overall, the tracking is very accurate everywhere, except for small overshoots in response to the three steps. In particular, the delays in following the ramps and sine wave are very close to the theoretical value of 50ms implied by the transfer function, and there is no sign of non-minimum-phase behaviour. It is quite noticeable that q_c and q_2 are almost the same, whereas q_f follows a very different curve. This shows that the acausal filter is accurately cancelling out the zero at $1/T_c$. The action of leaning in anticipation can be seen in the value of q_1, especially between $t = 0$ and $t = 1$. However, this behaviour is more easily seen in the video in Extension 1.

The overshoots are caused by discontinuities in q_c: a step in the value of q_c causes a step in the value of $\dot{q}_f$, and therefore an infinite spike in $\ddot{q}_f$; but the spike invalidates Eq. 39. One remedy is to use one zero instead of two, so that $\alpha_2 = 0$ in Eq. 39 and $\ddot{q}_f$ is no longer required. However, a better solution is to avoid sending discontinuous command signals to the controller by replacing each step with an alternative waveform, such as a fast ramp. (Discontinuities in $\dot{q}_c$ do not cause overshoots.) The response to a fast ramp can be seen in Figure 6.

Returning to the period between $t = 0$ and $t = 1$, it can be seen that for as long as the input q_f is a rising exponential with time constant T_c, the robot simply leans over and falls without doing anything with its actuated joint. If the exponential were to continue, instead of stopping at $t = 1$, then the robot would continue falling until it hits the ground. Thus, an input signal of this form is guaranteed to make the robot fall over, even though

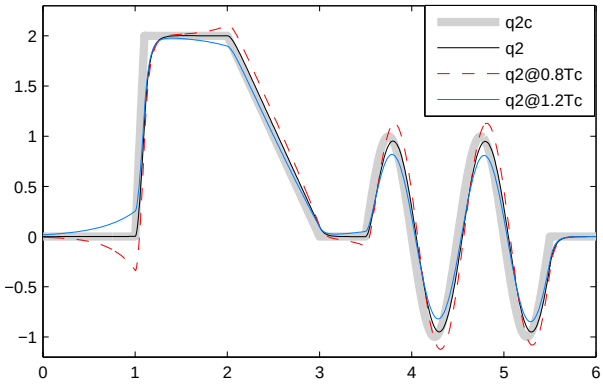

Figure 6: The effect of varying the value of T_c used by the acausal filter

all of the poles are stable.

5.2 Inaccurate T_c in Acausal Filter

The good performance of the balance controller relies on accurate cancellation of the zero at $1/T_c$ by the acausal filter. This raises the question of what happens if the value of T_c used by the filter is inaccurate. The value used in the first experiment is already slightly inaccurate because it is assumed to be a constant; but the consequences are too small to show up in Figure 5. Therefore, a second experiment was performed in which the acausal filter was programmed to use substantially incorrect values of T_c.

The results of this experiment are plotted in the graph in Figure 6, which shows the command signal and three responses: one using the correct value of T_c when $q_1 = 0$, one using 0.8 times the correct value, and one using 1.2 times the correct value. The command signal now consists of a fast ramp with a slope of 20rad/s, which replaces the initial step in Figure 5, followed by a ramp with a slope of -2rad/s and a 1Hz sine wave. When the filter uses the correct value of T_c, the resulting response is very good. When the filter uses 0.8 times the correct value, the controller under-estimates the amount of leaning required, and ends up making a small excursion in the opposite direction before each movement. It also overshoots on the sine wave. And when the filter uses 1.2 times the correct value, the controller over-estimates the amount of leaning required, and ends up creeping in the correct direction ahead of each movement. It also undershoots on the sine wave.

To put Figure 6 into context, Figure 7 shows the response of the balance controller without the acausal filter. On comparing these two figures, it can be seen that the tracking errors in Figure 7 are far greater than those in Figure 6. Thus, the acausal filter greatly improves the performance of the balance controller, even if it is using a value of T_c that is out by 20%.

Graphs resembling Figure 7 have been published previously. See, for example, Figure 5 in Azad and Featherstone (2016) or Figure 3.4 in Azad (2014). When comparing Figure 7 with these earlier graphs, a few differences should be borne in mind. First, the command signal contains much larger, faster movements. For example, the command signal in Figure 7 includes a 1Hz sine wave with a magnitude of 1rad, compared with a 0.125Hz sine wave with a magnitude of 0.2rad in the earlier graphs.

Another difference is that the gains are set much higher in the new controller—poles

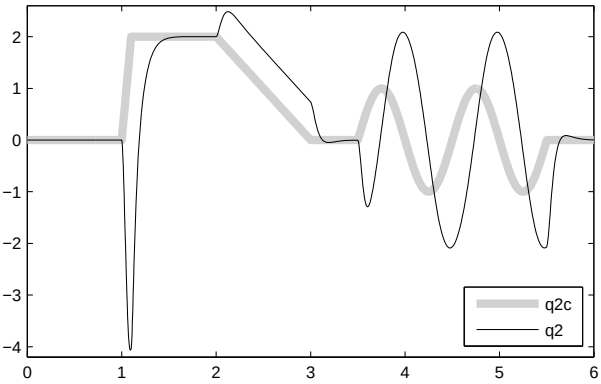

Figure 7: The response without the acausal filter

at −20rad/s instead of −7rad/s. Such high gains produce crisp, fast, accurate responses when the controller is used in conjunction with the acausal filter, but they produce unacceptably large excursions when used without the filter, such as the 4rad spike at the beginning of the fast ramp. In the absence of the acausal filter, there is a trade-off between the speed of response and the magnitudes of excursions at the beginnings and ends of movements. This trade-off is illustrated in Figure 6 in Azad and Featherstone (2016) and Figure 3.5 in Azad (2014). For a robot that stands approximately 1m tall, the sweet spot occurs when the poles are set at approximately −7rad/s. However, the acausal filter removes the need for this compromise, and therefore permits much faster poles to be used.

Finally, in Figure 7 the feedforward gains have not been tuned for accurate tracking of linear ramps. To track a linear ramp with zero delay, as illustrated in Figure 4, the transfer function must have the form $(a_0 + a_1 s + \cdots)/(b_0 + b_1 s + \cdots)$ with $a_0/b_0 = a_1/b_1$; and α_1 in Eq. 28 (not Eq. 39) must be chosen accordingly. This choice of α_1 replicates the effect of L_d in the earlier works.

Before moving on, it should be understood that T_c is an easily-measured property of a robot mechanism, and there should be no difficulty in measuring it to an accuracy better than 20%. So errors as large as those shown in Figure 6 should not occur in practice.

5.3 IMU Drift

All balance controllers are sensitive to errors in the estimate of the vertical direction. In a mobile robot, this estimate comes from an inertial measurement unit (IMU), possibly combined with other sensors. The output of an IMU can be regarded as being contaminated with noise having a slowly drifting mean; and the mean error can be regarded as a constant over short time intervals such as a few seconds.

Figure 8 shows the balance controller's response in the presence of a constant error in q_1, which mimicks a slowly varying error in an IMU's estimate of the vertical direction. The magnitude of the error is 0.5° (0.009rad), which is typical of the maximum error in the kinds of high-quality IMU that are currently used in humanoids and legged robots.

Two responses are shown. The one labelled 'q2 fast' refers to the controller as described earlier, and the one labelled 'q2 slow' refers to a new setting of the gains in which the two cancelled poles have been reduced from −20rad/s to −3.5rad/s (a value close to $-1/T_c$), and the corresponding zeros have been reduced to match. So the two versions of the

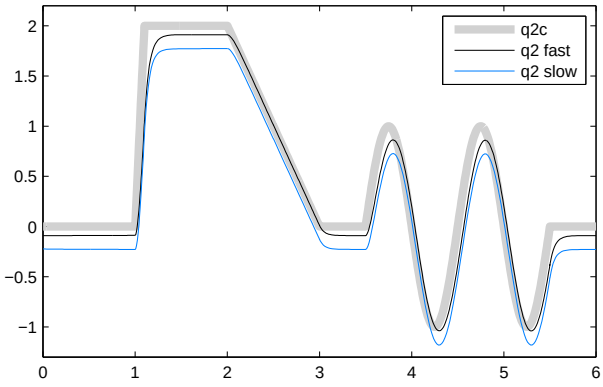

Figure 8: The effect of a constant error in q_1 mimicking a slowly varying error in the IMU's estimate of the vertical direction

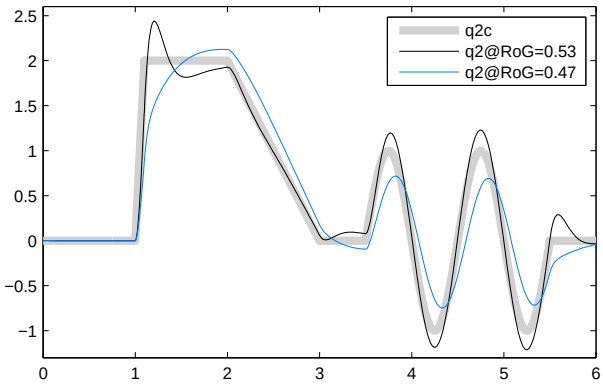

Figure 9: The effect of error in the dynamic model used by the controller

controller theoretically have the same transfer function, but the latter has substantially lower feedback gains.

The result of a constant error in q_1 is a constant error in q_2. For the high-gain controller the error is 0.09rad (5°), and for the low-gain controller it is 0.23rad (13°). These results bracket the results reported in Figure 14 in Azad and Featherstone (2016), where a 1° error in the sensor produces approximately a 0.3rad error in both Azad's controller and the controller of Grizzle et al. (2005), admittedly on a different robot.

This result indicates that accurate tracking of command signals will require accurate compensation of IMU drift. The obvious way to implement this is to use the steady-state error in the robot's tracking response as a measure of the drift in the IMU.

5.4 Model Error

The next effect to be investigated is an error in the dynamic model used by the controller. Figure 9 shows what happens when the controller's model has an incorrect value for the radius of gyration of the flywheel: either 0.53m or 0.47m. (The correct value is 0.5m.) This error causes only a 1% error in the controller's calculated value of T_c, which is unlikely to have much effect, but a 10% error in the calculated value of the velocity gain (G_v), which is the main effect visible in the graph.

As Figure 9 shows, when the controller believes the radius of gyration to be 0.53,

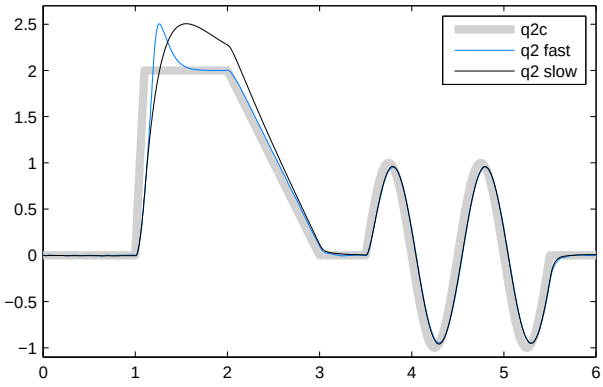

Figure 10: Time delays, sensor noise and actuator saturation

it over-reacts to the command signal, resulting in overshoots and ringing; and when it believes the radius of gyration to be 0.47, it under-reacts, resulting in a sluggish response but still with some evidence of overshoot.

One important result from this experiment is that the high-gain controller (cancelled poles at -20) goes unstable with this magnitude of model error, so the results in Figure 9 were obtained using the low-gain controller (cancelled poles at -3.5). This suggests that the gain settings in the high-gain controller are too high for practical use.

5.5 Time Delay, Noise and Saturation

The next experiment investigates the effect of computation time delays, sensor noise and actuator saturation. In particular:

1. The controller is modelled as a servo running at a rate of 200Hz, with a full 5ms delay between inputs and outputs.

2. Uncorrelated Gaussian noise of magnitude 0.08°/s RMS is fed into the controller's reading of $\dot{q}_1$, and its integral is fed into the reading of q_1, which simulates the effect of noise in an IMU's estimate of the vertical direction.

3. The actuator torque is limited to the range $(-2\dot{q}_2 \pm 40)$Nm, which sets both a speed limit of 20rad/s and a power limit of 200W when the actuator is performing positive work.

The velocity noise magnitude is based on the data sheet of the 3DM-GX4 IMU from Lord Microstrain, which specifies gyro noise density of $0.005°/\text{s}/\sqrt{\text{Hz}}$ and a maximum bandwidth of 250Hz. In theory, the integral of this noise should drift without limit. However, during the relatively short period of the simulation, the amount of drift is so small that the noise signal injected into q_1 stays well below the datasheet value of roll and pitch error. The actuator torque limit is consistent with an electric motor having a nominal speed of about 6000rpm and a stall torque of a little more than 1.5Nm (to account for frictional losses) driving a 30 : 1 ratio harmonic gear. 40Nm is about one ninth of the peak torque requested by the high-gain controller to follow the fast ramp.

Figure 10 shows the response of both the high-gain controller (labelled 'q2 fast') and the low-gain controller ('q2 slow') under these conditions. It turns out that the 5ms delay

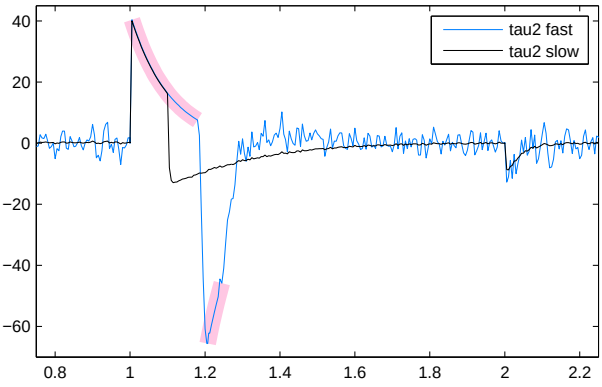

Figure 11: Actuator output torque during and after the fast ramp. Saturated outputs are highlighted

has almost no effect. Likewise, the noise has too little effect to be visible in this graph. However, the effect of actuator saturation is clearly visible.

In Figure 10, the actuator saturates only on the fast ramp. The primary effect of this saturation is that the robot moves more slowly than assumed by the acausal filter, which gives gravity more time to tip the robot in whichever direction it happens to be leaning. The end result is that the robot has leaned a little too much, and has to overshoot in order to correct the mistake. Figure 10 also shows one of the negative consequences of lowering the gains: a slower response to unanticipated disturbances.

To further illustrate the actuator saturation, Figure 11 plots the actuator output torque over the period from 0.8s to 2.2s, which covers the whole of the fast ramp and the beginning of the slow one. The portions of the curve where the actuator is in saturation are highlighted in pink. When the actuator is not saturated, the output torque is exactly the value computed by the control system.

This figure also shows a sharp difference between the two controllers' response to the IMU noise, the response of the high-gain controller being more than 10 times greater than that of the low-gain controller. An analysis of the velocity signals reveals the following: in response to the 0.08°/s RMS noise injected into the measurement of $\dot{q}_1$, the resulting RMS noise in the value of $\dot{q}_1$ is 0.07°/s in the case of the low-gain controller and 0.86°/s in the case of the high-gain controller. So the high-gain controller has greatly over-reacted to the noise, and the low-gain controller has not.

In both cases, the RMS noise in the value of $\dot{q}_2$ is six times greater than that in $\dot{q}_1$. This is in accordance with expectation based on the angular velocity gain of the robot (Featherstone, 2015a). Angular velocity gain is the ratio of a change in the angular velocity of the CoM about the support point to a change in the angular velocity of the actuated joint, both changes being caused by an impulse at the actuated joint. For this particular robot, the angular velocity gain is the ratio $\Delta\dot{q}_1/\Delta\dot{q}_2$, and its value is $-1/6$.

5.6 Comparison with Other Balance Controllers

The final experiment compares the step response of the new controller with those of four other controllers: the ones described in Spong (1995), Berkemeier and Fearing (1999), Azad and Featherstone (2016) and Grizzle et al. (2005). Specifically, the test leading to

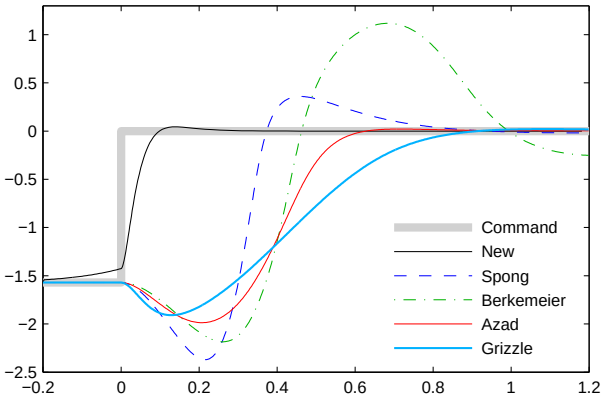

Figure 12: Comparison of the step response of the new controller with those of four existing balance controllers (times in seconds, angles in radians)

	link 1	link 2
link length (m)	0.4	0.6
mass (kg)	0.49	0.11
CoM (m)	0.1714	0.4364
inertia@CoM (kgm^2)	0.0036	0.0043

Table 1: Parameters of the double pendulum used in Figure 12

the data plotted in Figure 13 of Azad and Featherstone (2016) was repeated with the new controller, and the results are shown in Figure 12.

This experiment uses the same robot as in Azad and Featherstone (2016), and its parameters are given in Table 1. This robot's balancing dynamics are approximately 50% faster than the flywheel and stick, so the new controller's gains were set as follows: one pole at $-1/T_c$, three poles at -30rad/s and two zeros at -30rad/s. ($T_c = 0.213$s for this robot in its initial configuration, and this is the value used in the acausal filter.) The gains of the other controllers were set as explained in Azad and Featherstone (2016), which also explains the steps that were taken to achieve as fair a comparison as possible.

Figure 12 plots the command signal and all five responses. The command is a step from $q_2 = -\pi/2$ to $q_2 = 0$ at $t = 0$, and the responses of the four existing controllers are identical to those shown in Figure 13 of Azad and Featherstone (2016). In the new controller's response, it can be seen that q_2 begins to creep forward in anticipation of the step, so that the new controller has a small head start on the other four when the step takes place at $t = 0$. This creep resembles the creep that appears in Figure 6, but it has a different cause: the nonlinearity of this robot's dynamics in its initial configuration.

When the step occurs, the new controller's response is swift and accurate, with no sign of non-minimum-phase behaviour. It takes 0.1s to reach $q_2 = 0$, and a further 0.15s to settle to within 0.01rad of the commanded position after an overshoot of 0.044rad. The other controllers are all slower by more than a factor of 3, and the main reason can be seen to be their non-minimum-phase behaviour. As explained in Section 5.2, non-minimum-phase behaviour imposes a compromise between speed and excursions in the wrong direction. Some of these controllers can be sped up by using higher gains, but only at the expense of bigger excursions.

6 Extension to General Planar Robots

This section extends the theory of sections 3 and 4 to the case of a general planar robot, which may contain kinematic loops, balancing on a single point. The performance of the extended controller is illustrated with a simulation of a triple pendulum making a variety of movements.

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, which can be devoted to tasks other than balancing. Another thing that changes is that the robot now has a balance null space, which is the space of movements of the actuated joints that do not affect c_x. Ideally, the non-balance motions should be designed to lie in or close to this null space, so that they cause little or no disturbance to the robot's balance. However, if the robot is able to lean in anticipation then substantial disturbances can be accommodated.

Let us 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. As the mechanism may contain kinematic loops, not all of the joint variables will be independent. We therefore introduce a vector of independent generalized coordinates, $\boldsymbol{y} = [y_0\ y_1\ y_2\ \boldsymbol{y}_3^{\mathrm{T}}]^{\mathrm{T}}$, 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 one actuated joint motion, or any desired combination of actuated joint motions, provided that the chosen movement has a nonzero effect on c_x. However, in practice one should choose a movement that has a large effect. y_2 can be regarded as 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 robot is

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} & \boldsymbol{H}_{03} \\ H_{10} & H_{11} & H_{12} & \boldsymbol{H}_{13} \\ H_{20} & H_{21} & H_{22} & \boldsymbol{H}_{23} \\ \boldsymbol{H}_{30} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} & \boldsymbol{H}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{y}_2 \\ \ddot{\boldsymbol{y}}_3 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ \boldsymbol{C}_3 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ w_2 \\ \boldsymbol{w}_3 \end{bmatrix}, \tag{41}$$

in which w_2 and $\boldsymbol{w}_3$ are the generalized forces corresponding to y_2 and $\boldsymbol{y}_3$, and H_{ij} are the elements and submatrices of a generalized inertia matrix. This equation replaces Eq. 8. Equations 3–5 and 7 remain valid, but Eq. 6 becomes

$$L = H_{11}\dot{q}_1 + H_{12}\dot{y}_2 + \boldsymbol{H}_{13}\dot{\boldsymbol{y}}_3\,. \tag{42}$$

Likewise, Eq. 9 becomes

$$-\ddot{L}/g = H_{01}\dot{q}_1 + H_{02}\dot{y}_2 + \boldsymbol{H}_{03}\dot{\boldsymbol{y}}_3\,, \tag{43}$$

and so Eq. 10 becomes

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{y}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_{13} \\ -g\boldsymbol{H}_{03} \end{bmatrix} \dot{\boldsymbol{y}}_3\,. \tag{44}$$

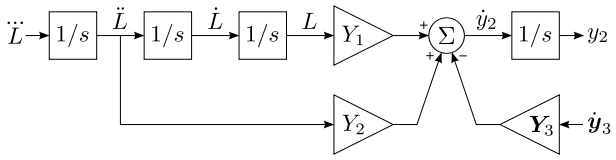

Figure 13: Modified plant model for a general planar robot

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

$$\dot{y}_2 = Y_1 L + Y_2 \ddot{L} - \boldsymbol{Y}_3 \dot{\boldsymbol{y}}_3 \,, \tag{45}$$

where Y_1 and Y_2 are as given in Eq. 12, and

$$\boldsymbol{Y}_3 = \frac{\boldsymbol{E}}{D} \tag{46}$$

where

$$\boldsymbol{E} = \boldsymbol{H}_{13} H_{01} - H_{11} \boldsymbol{H}_{03} \tag{47}$$

(cf. Eq. 13). The modified plant model is shown in Figure 13. 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 do not disturb the robot's balance.

The complete control system now consists of a balance controller, which is responsible for y_2, and a motion controller, which is responsible for $\boldsymbol{y}_3$. The motion controller receives as input a command signal $\boldsymbol{y}_{3\mathrm{c}}$, and produces as output the value of $\ddot{\boldsymbol{y}}_3$ calculated according to a suitable control law. The balance controller receives both the control signal $y_{2\mathrm{f}}$, which replaces q_{f}, and the signal $\dot{\boldsymbol{y}}_{3\mathrm{f}}$ which is explained below. The output is still $\dddot{L}$.

The design of the balance controller is largely unaffected by $\dot{\boldsymbol{y}}_3$. In particular, Eq. 30 is unaffected, and the gains are still as given in Eq. 33. However, there is now a need to compensate for the disturbances caused by $\dot{\boldsymbol{y}}_3$. Furthermore, it is desirable that the robot should lean in anticipation of these disturbances. This can be accomplished with the modified control law

$$\dddot{L} = k_{dd} \ddot{L} + k_d \dot{L} + k_L (L - \frac{\boldsymbol{Y}_3 \dot{\boldsymbol{y}}_{3\mathrm{f}}}{Y_1}) + k_q (q_2 - u) \,, \tag{48}$$

which replaces Eq. 27. In this equation, $\dot{\boldsymbol{y}}_{3\mathrm{f}}$ is the acausally filtered value of $\dot{\boldsymbol{y}}_{3\mathrm{c}}$, which is the derivative of the command signal $\boldsymbol{y}_{3\mathrm{c}}$. As the actual disturbance depends on the actual velocity, rather than the commanded velocity, it follows that there is an assumption of accurate tracking of $\dot{\boldsymbol{y}}_{3\mathrm{c}}$ built into this control law. The rationale behind Eq. 48 is that cancellation of the incoming signal $\boldsymbol{Y}_3 \dot{\boldsymbol{y}}_3$ requires L to be offset by an amount $\boldsymbol{Y}_3 \dot{\boldsymbol{y}}_3 / Y_1$.

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

$$\begin{bmatrix} 0 & \mathbf{0} & H_{01} & H_{02} \\ 0 & \mathbf{0} & H_{11} & H_{12} \\ -1 & \mathbf{0} & H_{21} & H_{22} \\ \mathbf{0} & -\mathbf{1} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} \end{bmatrix} \begin{bmatrix} w_2 \\ \boldsymbol{w}_3 \\ \ddot{q}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 - \boldsymbol{H}_{03} \ddot{\boldsymbol{y}}_3 \\ -C_1 - \boldsymbol{H}_{13} \ddot{\boldsymbol{y}}_3 \\ -C_2 - \boldsymbol{H}_{23} \ddot{\boldsymbol{y}}_3 \\ -\boldsymbol{C}_3 - \boldsymbol{H}_{33} \ddot{\boldsymbol{y}}_3 \end{bmatrix} , \tag{49}$$

which is the generalization of Eq. 34. This is the point where the outputs of the motion and balance controllers ($\ddot{\boldsymbol{y}}_3$ and $\dddot{L}$) are combined. The final step is to solve

$$\boldsymbol{G}^{\mathrm{T}}\boldsymbol{\tau}_{\mathrm{a}} = \begin{bmatrix} w_2 \\ \boldsymbol{w}_3 \end{bmatrix}, \tag{50}$$

where $\boldsymbol{\tau}_{\mathrm{a}}$ is the vector of force variables at the actuated joints, and $\boldsymbol{G}$ is the matrix that maps $[\dot{y}_2\ \dot{\boldsymbol{y}}_3^{\mathrm{T}}]^{\mathrm{T}}$ to the vector of actuated joint velocities (i.e., a submatrix of the Jacobian from $\dot{\boldsymbol{y}}$ to $\dot{\boldsymbol{q}}$). If the mechanism has the same number of actuators as actuated motion freedoms then $\boldsymbol{G}$ is square and Eq. 50 has a unique solution; but if the mechanism is redundantly actuated then $\boldsymbol{G}$ is rectangular and Eq. 50 has infinitely many solutions. In this case it is necessary to choose one particular solution in accordance with a user-defined policy on the distribution of forces among the actuators.

6.1 Simulation and Analysis

This subsection demonstrates the performance of the generalized balance controller by means of a simulation in which the controller makes an inverted triple pendulum perform a variety of movements while maintaining its balance. A triple pendulum is chosen because it is the simplest mechanism that exhibits all of the dynamics discussed in this section.

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 0.2m, 0.25m and 0.35m; the masses are 0.7kg, 0.5kg and 0.3kg; and the links are modelled as point masses with the mass located at the far end of each link. These are the parameters of a robot identified in Featherstone (2015a) as being good at balancing, and also the robot used in Featherstone (2015b). To facilitate comparison, the commanded movements are the same as those in Featherstone (2015b), but on a faster time scale in view of the higher performance of the newer control system.

The position controller employed in this simulation is a simple PD controller with acceleration feedforward (from the command input $\ddot{\boldsymbol{y}}_{3\mathrm{c}}$) and two poles at -20rad/s, so the proportional and derivative gains are 400 and 40, respectively. As the controller benefits from the dynamics calculation in Eq. 49, it is effectively a computed-torque controller, and its tracking accuracy is perfect everywhere except where there is a step change in the position or velocity command.

The balance controller uses the control law in Eq. 48 with feedback gains determined by placing one pole at $-1/T_{\mathrm{c}}$ and the other three all at -20rad/s. The input u is calculated from $y_{2\mathrm{f}}$ and its derivatives, as per Eq. 39, using feedforward gains of $\alpha_1 = 0.1$ and $\alpha_2 = 0.0025$, which put two zeros at -20rad/s, thereby cancelling two of the poles. The theoretical closed-loop transfer function for the balance variable, y_2, is therefore $1/(1 + 0.05s)$.

The acausal filter was implemented analytically as follows. For the step and the sine wave, a constant value of T_{c} was used, equal to the correct value at configuration $[0, 0, 0]^{\mathrm{T}}$. For each ramp, two values were used: the correct values at the beginning and end of the ramp. This results in slightly incompatible curve segments because it ignores the effect of T_{c} varying continuously during the course of the ramp. So the segments were stitched together in such a way that the position signal was continuous, but the velocity signal

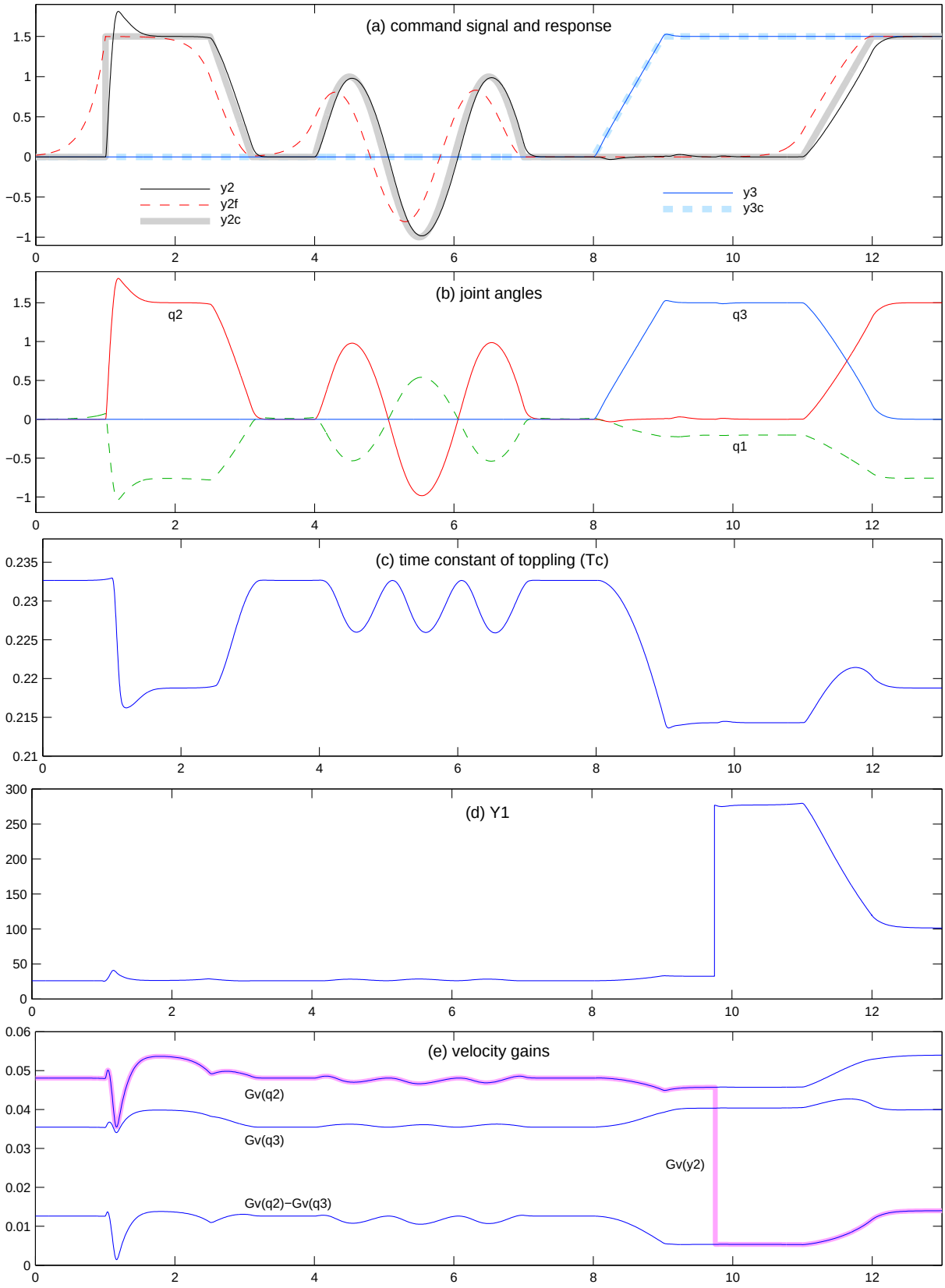

Figure 14: Simulation results for balancing a triple pendulum (times in seconds, angles in radians). This movement is shown in Extension 2

contained small steps. The consequences of these approximations are too small to show up in the graphs.

Figure 14(a) shows the command signals and responses, as well as the acausally filtered signal $y_{2\mathrm{f}}$. Times are expressed in seconds and angles in radians. 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 1.5rad. 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 $y_2 = q_2 - q_3$ from $t = 9.75$s onwards. So the final ramp involves q_2 ramping from 0 to 1.5rad while q_3 ramps from 1.5rad to 0. This can be seen clearly in Figure 14(b), which shows the motion of the robot expressed in joint space.

Note that these are relatively large, fast motion commands. Comparable graphs in the literature, such as Figure 5 in Azad and Featherstone (2016) and Figure 6 in Grizzle et al. (2005), show responses to much slower command signals; and the responses themselves are often sluggish. For example, Figure 6 in Grizzle et al. (2005) shows that the joint variables do eventually track the 0.2Hz sine wave command, but it takes them more than 10 seconds to settle into this pattern. In contrast, the new balance controller's settling time on the triple pendulum is about 0.5s after a step, and even less going into or out of a sine wave or ramp. Furthermore, the example mechanism in Grizzle et al. (2005) is of similar dimensions to the triple pendulum; so the balancing behaviour of the two robots will be similar, and the difference in settling times can be attributed almost entirely to the performance of the two control systems.

Figure 14(a) shows excellent tracking accuracy everywhere except for an overshoot on the step at $t = 1$ and a delay in tracking the final ramp that is larger than for the other movements. The overshoot is essentially the same phenomenon as discussed in Section 5.1. Again, the best remedy is simply not to issue a step command to the balance controller. The problem with the final ramp is discussed below. Both the first ramp and the sine wave are tracked with a delay only a few milliseconds greater than the theoretical delay of 50ms implied by the transfer function for y_2.

Figure 14(a) also shows small tracking errors in y_2 at times 8s, 9s and 9.75s. They are also noticeable in Extension 2. The first two are caused by the beginning and end of the ramp in y_3. As the ramp involves two step changes in velocity, the position controller experiences a small tracking error at both the beginning and the end of the ramp. The small tracking errors in y_2 are of similar magnitude to the momentary tracking errors in y_3, and can be attributed to the small difference between the actual and anticipated values of the balance disturbance caused by the ramp. These tracking errors are much smaller than the equivalent tracking errors appearing in Figure 5(a) of Featherstone (2015b), which shows that leaning in anticipation greatly improves the balance controller's ability to reject disturbances caused by the other movements.

The third tracking error is caused by the step change in the definition of y_2 at $t = 9.75$s mentioned above. The redefinition causes a disturbance that is proportional to the amount of activity in the state variables at the time that the change occurs. If the robot were completely stationary then the disturbance would be zero. This is one aspect of the balance control system that needs more work: there needs to be a way to vary the mapping between $\boldsymbol{y}$ and $\boldsymbol{q}$ while the robot is in motion without significantly degrading the performance of the controller.

Figure 14(c) plots the value of T_{c}, which can be seen to vary in a narrow range from

approximately 0.214s to 0.233s 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 an acceptable approximation.

Figure 14(d) plots the value of Y_1. For the first 9.75 seconds this quantity varies in a range from approximately 26 to 33, with a spike of 41 coinciding with the overshoot following the step in y_{2c} at $t = 1$. However, at the point where y_2 is redefined, Y_1 jumps to 277, and then climbs slightly to 279 before dropping steadily back down to 101 during the final ramp. So, for the first 9.75 seconds 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 Figure 14(e), which plots the velocity gains of joints 2 and 3 along with their difference, which is the velocity gain of the motion freedom $q_2 - q_3$ (Featherstone, 2015a). For the first 9.75 seconds $G_v(y_2) = G_v(q_2)$; but then y_2 is redefined, and for the remaining time $G_v(y_2) = G_v(q_2) - G_v(q_3)$. As $G_v(y_2)$ appears in the denominator of Eq. 25, this accounts for the large change in Y_1.

The large drop in $G_v(y_2)$ provides an explanation for the relatively poor tracking of the final ramp: with G_v so close to zero, the balance controller is attempting to balance the robot using a movement that has almost no effect of the horizontal coordinate of the CoM, making the task of balancing much more difficult. In a practical system with noisy sensors, there would be a large increase in the magnitude of the quivering of the robot due to sensor noise, beginning at the moment when y_2 is redefined.

Without a model of the physical process of balancing, in which the behaviour of the plant is described in terms of physically meaningful properties like velocity gain, it would not be easy to understand the dependency of balancing performance on the commanded behaviour.

7 Extension to 3D

The ultimate objective of the work presented in this paper is a simple, high-performance balancer in 3D. This section outlines briefly some of the progress made so far, and what still needs to be done.

In principle, balancing in 3D involves controlling two horizontal coordinates of the CoM instead of only one. This can be accommodated in the model in Figure 2 by replacing the state variables L, $\dot{L}$, $\ddot{L}$ and q_2 with 2D vectors, and replacing the gains Y_1 and Y_2 with 2×2 matrices. The resulting plant model has two inputs, two outputs and 8 state variables, and the balancing activity requires two actuated degrees of freedom.

The robot also has two state variables relating to its freedom to spin about the contact normal. Of these, only the velocity variable appears in the robot's equation of motion and influences how the robot and the balance controller behave. However, neither variable is directly related to balancing, so neither variable appears in the 3D balancing plant model.

The coefficient matrix in the closed-loop equation of motion is now an 8×8 matrix. However, it is at least sometimes possible to find a coordinate system in which both of the 2×2 matrices replacing Y_1 and Y_2 are triangular, in which case the characteristic equation of the 8×8 matrix simplifies into the product of two quartic polynomials, thereby allowing

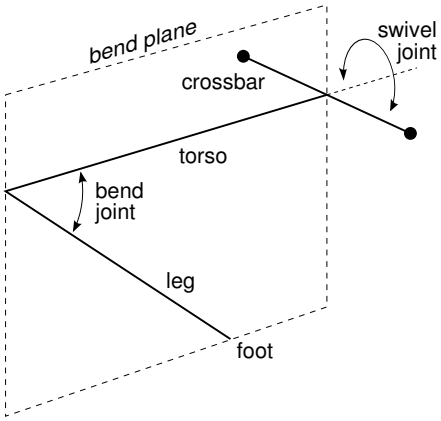

Figure 15: A simple 3D hopping and balancing robot

the gains of the 3D balance controller to be determined in a manner very similar to that described in Section 4.

For a highly symmetrical robot, like the Ballbot (Lauwers et al., 2006), the problem would now be solved. However, for a less symmetrical robot there is more to do. For example, Figure 15 shows a simplified version of a 3D hopping and balancing robot called Skippy, which is still at the design stage. This robot consists of a leg, a torso and a crossbar, connected by actuated revolute joints, and it can be regarded as a planar double pendulum with a crossbar attached to the top.

For this robot, or any machine similar to it, it makes sense to decompose the task of 3D balancing into two subtasks: balancing in the plane, and keeping the plane vertical. This is the idea behind *bend-swivel control* (Azad, 2014; Azad and Featherstone, 2014), and an example of the resulting behaviour is shown in Extension 3. Clearly, the bend controller is just the balance controller described here. However, the swivel controller is a little different because it has an extra function: in addition to keeping the bend plane vertical, this controller also serves to rotate the plane about the vertical line through the contact point, thereby allowing the robot to reorient itself to face in any direction.

One issue that needs to be resolved is to identify the conditions under which the 2×2 gain matrices can be made to be triangular. Another issue is that gyroscopic forces arise in the 3D case, and they cause significant disturbance to the robot's balance. It will therefore be necessary for the plant model to account for them, and for the controller to compensate for them.

8 Conclusion

This paper has presented a new model of the physical process of balancing on a point 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; and a physical interpretation of these numbers is provided. All three numbers can be computed efficiently using standard dynamics algorithms.

The model gives rise to a simple balance controller, and also a simple method of leaning in anticipation. The latter consists of a time-reversed low-pass filter interposed

between the command signal and the controller's input, which runs from a point in the future back to the present. It therefore requires a preview of future values of the command signal. The effect of the filter is to cancel the non-minimum-phase behaviour of the robot, which allows the balance controller to track the command signal accurately and at a bandwidth substantially faster than the robot's natural frequency of toppling. It also helps the balance controller to reject disturbances caused by other commanded motions of the robot. Simulation results are presented showing the accuracy with which the controller tracks motion commands; its robustness to model errors, computation time delays, sensor noise and actuator saturation; a comparison of its step response with those of other balance controllers; and how it works in combination with a separate motion controller.

As planar balancing is a solved problem, the contribution of this paper is to simplify the problem and its solution without loss of generality, to present a controller with excellent performance at high-bandwidth tracking of large, fast motions, 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 the final section in this paper briefly explains how this can be done.

Acknowledgements

The author wishes to thank the anonymous reviewers whose helpful comments spurred a significant improvement in the content of this paper. Also, the work presented here owes much to the work of Morteza Azad, as described in Azad (2014), and the data used to compile Figure 12 was kindly supplied by Dr. Azad.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

Anderson, B. D. O., and Moore, J. B. (1971). *Linear Optimal Control*. Englewood Cliffs, New Jersey: Prentice-Hall.

Azad, M. (2014). Balancing and Hopping Motion Control Algorithms for an Under-actuated Robot. Ph.D. Thesis, The Australian National University, School of Engineering.

Azad, M., and Featherstone, R. (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.

Azad, M., and Featherstone, R. (2016). Angular momentum based balance controller for an under-actuated planar robot. *Autonomous Robots*, 40(1):93–107.

Berkemeier, M. D., and Fearing, R. S. (1999). Tracking Fast Inverted Trajectories of the Underactuated Acrobot. *IEEE Trans. Robotics & Automation*, 15(4):740–750.

Blickhan, R. (1989). The Spring-Mass Model for Running and Hopping. *J. Biomechanics*, 22(11/12):1217–1227.

Double Robotics. (2015). 'Double' telepresence robot. http://www.doublerobotics.com, accessed Dec. 2015.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. New York: Springer.

Featherstone, R. (2015a). Quantitative Measures of a Robot's Ability to Balance. In: *Proc. Robotics: Science and Systems*, Rome, Italy, July 13–17. Available at: http://-roboticsproceedings.org/rss11/p26.html

Featherstone, R. (2015b). A New Simple Model of Balancing in the Plane. *Int. Symp. Robotics Research*, Sestri Levante, Italy, Sept. 12–15.

Full, R. J., and Koditschek, D. E. (1999). Templates and Anchors: Neuromechanical Hypotheses of Legged Locomotion on Land. *J. Experimental Biology*, 202:3325–3332.

Grizzle, J. W., Moog, C. H., and Chevallereau, C. (2005). Nonlinear Control of Mechanical Systems With an Unactuated Cyclic Variable. *IEEE Trans. Automatic Control*, 50(5):559–576.

Hauser, J., and Murray, R. M. (1990). Nonlinear Controllers for Non-Integrable Systems: the Acrobot Example. In: *Proc. American Control Conf.*, San Diego, CA, May 23–25, pp. 669–671.

Ibanez, A., Bidaud, P., and Padois, V. (2012). Unified Preview Control for Humanoid Postural Stability and Upper-limb Interaction Adaptation. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct. 7–12, pp. 1801–1808.

Kajita, S., et al. (2003). Biped Walking Pattern Generation by Using Preview Control of Zero-Moment Point. In: *Proc. IEEE Int. Conf. Robotics and Automation*, Taipei, Taiwan, Sept. 14–19, pp. 1620–1626.

Lauwers, T. B., Kantor, G. A., and Hollis, R. L. (2006). A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive. In: *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, FL, May 15–19, pp. 2884–2889.

Miyashita, N., Kishikawa, M., and Yamakita, M. (2006). 3D Motion Control of 2 Links (5 DOF) Underactuated Manipulator Named AcroBOX. In: *Proc. American Control Conf.*, Minneapolis, MN, June 14–16, pp. 5614–5619.

Rabbani, A. H., van de Panne, M., and Kry, P. G. (2014). Anticipatory Balance Control. In: *Proc. 7th Int. Conf. Motion in Games*, Los Angeles, CA, Nov. 6–8, pp. 71–76.

Segway Inc. (2015). Personal Transporter. http://www.segway.com, accessed Dec. 2015.

Spong M. W. (1995). The Swing Up Control Problem for the Acrobot. *IEEE Control Systems Magazine*, 15(1):49–55.

Wieber P.-B. (2006). Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In: *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Genoa, Italy, Dec. 4–6, pp. 137–142.

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.

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.

Appendix A: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

Extension	Media type	Description
1	Video	The motion sequence shown in Figure 5
2	Video	The motion sequence shown in Figure 14
3	Video	Demonstration of bend-swivel control

Appendix B: Property of Joint-space Momentum

This appendix proves the result $p_i = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{h}_{\nu(i)}$ for any joint i in a general rigid-body system, provided that the joint does not participate in any kinematic loop. p_i and $\boldsymbol{s}_i$ are the momentum variable and axis vector of the joint, and $\boldsymbol{h}_{\nu(i)}$ is the total momentum of the subsystem consisting of body i and its descendants. 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.

Given a general rigid-body system, we first construct a spanning tree and number the bodies and joints therein 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. By definition, the two are related by the equation

$$\boldsymbol{p} = \boldsymbol{H}\dot{\boldsymbol{q}}\,, \tag{51}$$

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

$$p_i = \sum_{j=1}^{N} H_{ij}\dot{q}_j\,. \tag{52}$$

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

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j & \text{if } i \in \nu(j) \\ \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

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 $\boldsymbol{I}_{\nu(i)} = \sum_{j \in \nu(i)} \boldsymbol{I}_j$ (i.e., the composite-rigid-body inertia of the subtree).

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. 53 as follows:

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j & \text{if } j \in \bar{\kappa}(i) \\ \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{54}$$

which makes it clear that H_{ij} is nonzero only if $j \in \bar{\kappa}(i)$ or $j \in \nu(i)$. Substituting Eq. 54 into Eq. 52 gives

$$\begin{aligned} p_i &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{j \in \bar{\kappa}(i)} \boldsymbol{I}_{\nu(i)} \boldsymbol{s}_j \dot{q}_j + \sum_{j \in \nu(i)} \boldsymbol{I}_{\nu(j)} \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{j \in \bar{\kappa}(i)} \sum_{k \in \nu(i)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j + \sum_{j \in \nu(i)} \sum_{k \in \nu(j)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \Big(\sum_{k \in \nu(i)} \sum_{j \in \bar{\kappa}(i)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j + \sum_{k \in \nu(i)} \sum_{j \in \nu(i) \cap \kappa(k)} \boldsymbol{I}_k \boldsymbol{s}_j \dot{q}_j \Big) \\ &= \boldsymbol{s}_i^{\mathrm{T}} \sum_{k \in \nu(i)} \boldsymbol{I}_k \sum_{j \in \kappa(k)} \boldsymbol{s}_j \dot{q}_j \,. \end{aligned} \tag{55}$$

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 the tree is the sum of the joint velocities between it and the base, so

$$\boldsymbol{v}_k = \sum_{j \in \kappa(k)} \boldsymbol{s}_j \dot{q}_j \,. \tag{56}$$

We can now further simplify Eq. 55 as follows:

$$p_i \;=\; \boldsymbol{s}_i^{\mathrm{T}} \sum_{k \in \nu(i)} \boldsymbol{I}_k \boldsymbol{v}_k \;=\; \boldsymbol{s}_i^{\mathrm{T}} \sum_{k \in \nu(i)} \boldsymbol{h}_k \;=\; \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{h}_{\nu(i)} \,, \tag{57}$$

which establishes the desired result for the case of a kinematic tree.

If the original system contains kinematic loops then Eq. 51 must be replaced with the equivalent closed-loop equation, which changes the definition of $\boldsymbol{p}$. However, Eq. 52 remains valid for any joint i that does not participate in any kinematic loop. As the loop-closure constraints do not affect any of the other steps, it follows that Eq. 57 applies also in the case of a joint in a general rigid-body system that does not participate in any kinematic loop.