Auton Robot (2016) 40:93–107
DOI 10.1007/s10514-015-9446-z

Angular momentum based balance controller for an under-actuated planar robot

Morteza Azad[1] · **Roy Featherstone**[2]

Received: 10 October 2014 / Accepted: 5 June 2015 / Published online: 20 June 2015
© Springer Science+Business Media New York 2015

Abstract In this paper, a new control algorithm based on angular momentum is presented for balancing an under-actuated planar robot. The controller is able to stabilize the robot in any unstable balanced configuration in which the robot is controllable, and also it is able to follow a class of arbitrary trajectories without losing balance. Simulation results show the good performance of the controller in balancing and trajectory tracking motions of the robot. The simulations also show that the proposed controller is robust to significant imperfections in the system, such as errors in the controller's dynamic model of the robot and imperfections in the sensors and actuators. The new controller is compared with three existing balance controllers and is shown to equal or outperform them.

Keywords Under-actuated robots · Momentum based control · Balance control

1 Introduction

This paper introduces a new and simple balancing control algorithm for an under-actuated planar robot based on its angular momentum. The use of angular momentum in the balance control of humanoids was pioneered by Goswami and Kallem (2004), and has been recently employed by other researchers as well (Lee and Goswami 2012; Macchietto et al. 2009). However, these works are aimed at robots with sufficiently large feet that they are capable of static balancing, whereas this paper considers the problem of balancing on a point or a line, for which static balancing is impossible. Such robots are called *under-actuated* because they posess at least one motion freedom—usually the freedom to topple—over which they have no direct control, and must instead control indirectly via other motion freedoms.

The problem of balancing on a point or a line is relevant to robots with point feet, as well as to robots that roll on a sphere (Lauwers et al. 2006) or a pair of wheels (Segway Inc. 2015), and to humanoids standing on a hump, traversing a log or walking a tightrope. Most quadruped robots have point feet, and would need to balance on a line if they raise two feet off the ground. Also, many biped and monopod robots have point feet (Raibert 1986; Westervelt et al. 2007). In this paper we study the special case of balancing in the vertical plane, although the technique described here can be extended to the case of balancing in 3D (Azad 2014; Azad and Featherstone 2014).

The robot that we study in this paper is called "2D balancer" and consists of two bodies connected to each other by an actuated revolute joint (hip joint). The lower body makes contact with the ground at a single sharp point, representing a point foot, or along a single sharp edge, representing a knife-edge foot. In the former case, we assume that the robot is somehow constrained to move in a vertical plane. In the latter case, the foot already provides the necessary constraint. In both cases, we also assume that the foot neither slips nor loses contact with the ground. This assumption allows us to model the contact as a passive revolute joint, so that the robot becomes a planar 2R mechanism.

The 2D balancer therefore resembles the point-foot balancer introduced in (Azad and Featherstone 2012), and also

✉ Morteza Azad
m.azad@bham.ac.uk

Roy Featherstone
roy.featherstone@ieee.org

[1] Department of Computer Science, University of Birmingham, Edgbaston, UK

[2] Deparment of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova, Italy

the acrobot (for acrobatic robot) introduced by (Hauser and Murray 1990). The latter has received much attention in the literature. However, the majority of studies on balancing this robot have been performed as a part of its swing-up motion control (Deng et al. 2007; Lai et al. 2005; Spong 1995; Xin and Yamasaki 2012), etc. In most of these studies, linear controllers, such as linear-quadratic regulators, are used to balance the robot when it is close to its upright unstable balanced configuration. The new controller presented in this paper is intended for balancing only, and includes an assumption that the robot's center of mass (CoM) is above the supporting surface.

There are a few studies in the literature that focus on the balancing motion of the acrobot (Formal'skii 2006; Olfati-saber 2000; Yamakita et al. 2002; Yonemura and Yamakita 2004). Berkemeier and Fearing (1999) introduced a linear controller based on zero dynamics for balancing and also trajectory tracking of the acrobot. They found a class of interesting feasible trajectories for the acrobot and achieved theoretically accurate trajectory tracking performance using their proposed controller.

Grizzle et al. (2005) considered the general case and designed a nonlinear controller for mechanical systems with one degree of under-actuation. Their proposed controller for the acrobot case becomes equivalent to the controllers that were already introduced in Yamakita et al. (2002) and Yonemura and Yamakita (2004) using output zeroing approach. Their proposed controllers use an output function which is defined by the angular momentum and one other new state.

In this paper, we focus on the balancing motion of the robot and introduce a new and simple non-linear controller for a 2D balancer mechanism which is based on its angular momentum about the first joint. It will be shown that controlling the angular momentum of the robot regulates the location and velocity (i.e. linear momentum) of the CoM of the robot in the horizontal direction. The new controller, first described in Azad (2014), is an extended version of the controller which was introduced in Azad and Featherstone (2012) and Azad and Featherstone (2013). The differences between the new controller in this paper and the previous version in Azad and Featherstone (2012) are discussed in Sect. 3.1. The improvements in the performance of the controller are shown by simulation results in Sect. 7. It is shown that the new controller is much faster than the old one and therefore follows desired trajectories much more accurately. Also the new control algorithm is compared in simulation with three other balancing controllers in the literature. These controllers are (1) Spong's LQR controller, (2) Berkemeier and Fearing's linear controller and (3) Grizzle's nonlinear controller.

It is proved that the new controller is able to stabilize the robot at any unstable balanced configuration in which the robot is physically controllable. This controller is able to follow setpoint commands, in which only the target configuration is given, and also motion trajectory commands, in which the motion of the actuated joint is a prescribed function of time. However, the latter necessarily involves tracking errors for the purpose of maintaining balance.

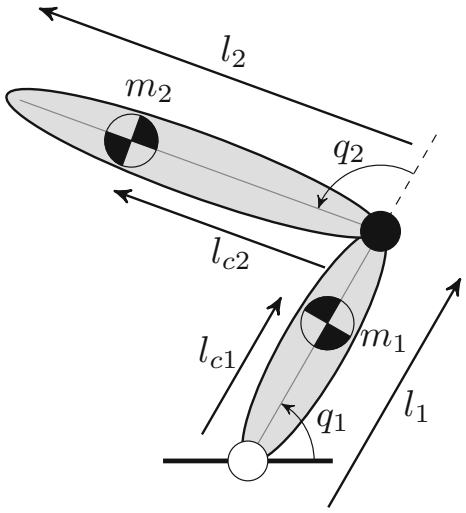

Fig. 1 Robot model and parameters

Our new proposed controller is also able to cope with imperfections in the system such as model errors, state estimation errors and quantization errors. It is shown in the simulations that the controller still shows a reasonably good performance even in the presence of a significant amount of imperfections in the system.

2 Robot model and motion equations

The 2D balancer is essentially an inverted double pendulum mechanism which is fixed passively to the ground via its first joint. This system is under-actuated with only one actuator which is applying torque to the hip joint. Figure 1 shows a schematic diagram of the 2D balancer and the parameters of the system.

The motion equations for this robot are:

$$\begin{aligned} 0 = {} & \Big(c_1 + c_2 + 2c_3\cos(q_2)\Big)\ddot{q}_1 + \Big(c_2 + c_3\cos(q_2)\Big)\ddot{q}_2 \\ & - \Big(2c_3\sin(q_2)\dot{q}_1\dot{q}_2 + c_3\sin(q_2)\dot{q}_2^2\Big) \\ & + c_4 g\cos(q_1) + c_5 g\cos(q_1+q_2) \end{aligned} \tag{1}$$

$$\begin{aligned} \tau = {} & \Big(c_2 + c_3\cos(q_2)\Big)\ddot{q}_1 + c_2\ddot{q}_2 \\ & + c_3\sin(q_2)\dot{q}_1^2 + c_5 g\cos(q_1+q_2) \end{aligned} \tag{2}$$

where I_1 and I_2 are the moments of inertia of the links about their centers of mass, g is the acceleration of gravity, and

$$\begin{aligned} c &= m_1 + m_2, & c_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1, \\ c_2 &= m_2 l_{c2}^2 + I_2, & c_3 &= m_2 l_1 l_{c2}, \\ c_4 &= m_1 l_{c1} + m_2 l_1, & c_5 &= m_2 l_{c2}. \end{aligned}$$

3 Balancing controller

In this section a feedback control law for balancing motion of the robot is derived. We first formulate the balancing conditions of the robot and then define an output function based on those conditions in a way that zeroing the output function is equivalent to satisfying the conditions for balance. So the feedback control law will drive the output function to zero exponentially and therefore it will stabilize the robot in its unstable balanced configuration.

Let X denote the horizontal displacement of the CoM relative to the first joint, so

$$X = \frac{1}{c}\Big(c_4\cos(q_1) + c_5\cos(q_1+q_2)\Big)\,, \tag{3}$$

and let L denote the angular momentum of the robot about the first joint:

$$L = \Big(c_1 + c_2 + 2c_3\cos(q_2)\Big)\dot{q}_1 + \Big(c_2 + c_3\cos(q_2)\Big)\dot{q}_2\,. \tag{4}$$

The conditions for balance are: $X = 0$ and $\dot{q}_1 = \dot{q}_2 = 0$. However, as $\dot{X}$ and L are both linear functions of $\dot{q}_1$ and $\dot{q}_2$, the two velocity constraints can be replaced with $\dot{X} = 0$ and $L = 0$. As L is chosen to be expressed at the first joint, which is passive, it follows from elementary classical mechanics that $\dot{L}$ must equal the moment of the gravitational force about the first joint. So

$$\dot{L} = -cgX\,, \tag{5}$$

and therefore also

$$\ddot{L} = -cg\dot{X}\,. \tag{6}$$

Equation (5) could also be derived by differentiating (4) and plugging it into (1). The conditions for balance can now be written $L = \dot{L} = \ddot{L} = 0$; so it is possible to consider the angular momentum as an output function and define a feedback controller that drives it to zero exponentially. One possible control law is

$$\tau = k_{dd}\ddot{L} + k_d\dot{L} + k_pL\,, \tag{7}$$

where k_{dd}, k_d and k_p are controller gains. Another possibility is

$$\tau = -k_v\dot{X} - k_xX + k_pL\,, \tag{8}$$

where $k_v = cgk_{dd}$ and $k_x = cgk_d$. This equation is obtained by substituting (5) and (6) into (7). The latter option clearly shows that controlling the angular momentum of the robot about its fixed joint regulates the CoM and the linear momentum as well.

3.1 Modifying the controller

In the proposed control laws in (7) and (8), the control torque in the balanced configuration is zero, which happens only if the robot is in its vertical configuration. It means that the proposed controller is able to stabilize the robot only in the vertical unstable balanced configuration. In this subsection we modify the controller to be able to stabilize the robot in any other unstable balanced configuration as well as the vertical one.

Let τ_g and τ_g^d denote the gravity terms of the second link's motion equation at the current and desired configurations. So from (2) we have

$$\tau_g = c_5g\cos\big(q_1+q_2\big) \quad\text{and}\quad \tau_g^d = c_5g\cos\Big(q_1^d+q_2^d\Big)\,, \tag{9}$$

where q_1^d and q_2^d are the desired values of q_1 and q_2 at the desired configuration, respectively. Any desired balanced configuration is characterized by the value of q_2^d. The value of q_1^d can then be calculated using the balance condition:

$$c_4\cos\Big(q_1^d\Big) + c_5\cos\Big(q_1^d+q_2^d\Big) = 0\,. \tag{10}$$

One possible way to modify the controller, to reach the above mentioned goal, is to add τ_g^d as an extra feed-forward term to the control law. So the discrepency between τ_g in the motion equation and τ_g^d in the control law drives q_2 to its desired value q_2^d. In this case, the control law will be

$$\tau = k_{dd}\ddot{L} + k_d\dot{L} + k_pL + \tau_g^d\,. \tag{11}$$

This is the approach taken in Azad and Featherstone (2012). The problems with this method are:

1. The range of the desired configurations in which the controller can balance the robot is limited to those with the links pointing upward (i.e., $\sin(q_1) > 0$ and $\sin(q_1+q_2) > 0$).
3. In the presence of modeling error, which is unavoidable in practice, the value of τ_g^d is not accurate so the robot will not converge exactly to the desired configuration, but to a nearby one instead.
3. Because of using a feed-forward term in the controller, any error in the actuator directly affects the accuracy of convergence to the desired configuration of the robot.

In this paper, we improve upon the work described in Azad and Featherstone (2012) by adding a gravity compensator to the controller and a virtual spring to the robot model. The gravity compensator (τ_g) cancels out the gravity term in (2) and the virtual spring (i.e., a linear rotational spring between the two links) adds an extra term in the left-hand side of (2).

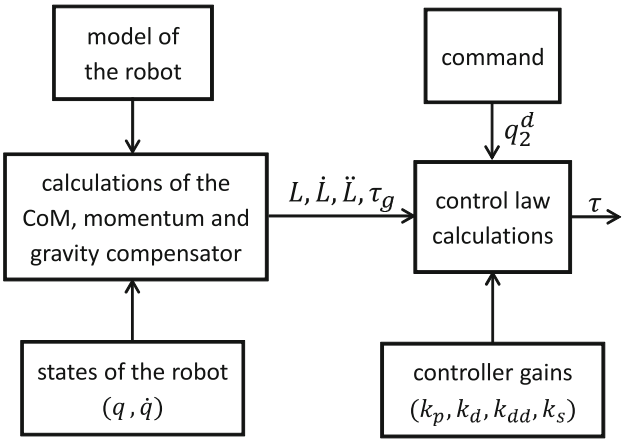

Fig. 2 Block diagram of the controller

The effect of the virtual spring is to provide a feedback term in the control law to decrease the error between q_2 and q_2^d. The new modified control law is then

$$\tau = k_{dd}\ddot{L} + k_d\dot{L} + k_pL + k_s\left(q_2^d - q_2\right) + \tau_g, \tag{12}$$

where k_s is the stiffness of the virtual spring in the robot model. The difference between this controller and the one proposed in Azad and Featherstone (2012) is that this new controller is relying on the virtual spring force, which is a feedback term, to drive q_2 to q_2^d whereas the previous controller is relying on the discrepency between τ_g and τ_g^d to do that job. The new method is preferable because it eliminates reliance on τ_g^d being very accurate. Also the new controller gives us one more parameter to tune (k_s) which can be used to improve the performance of the controller and extend the range of configurations in which the controller can successfully balance the robot.

Figure 2 shows how the control torque can be calculated by using the proposed controller. As can be seen in this diagram, the model and the states of the robot (q and $\dot{q}$) are employed to calculate the location of the CoM, linear and angular momentum of the robot and the gravity compensator. Then, by using these values, and also the command and the controller gains, the control torque will be computed from (12).

4 Stability analysis

In this section we study the stability of the proposed controller. We linearize the controller about its equilibrium state to study its local stability, but note that the controller itself is nonlinear. Considering the nonlinear state-space equations for the robot as

$$\dot{\eta} = f(\eta) + g(\eta)\ \tau,$$

where $\eta = (q_1 - q_1^d, q_2 - q_2^d, \dot{q}_1, \dot{q}_2)$, and knowing that τ is a function of η, it follows that $\dot{\eta}$ is a function of η, so the system as a whole can be described by a nonlinear equation of the form $\dot{\eta} = h(\eta)$. Linearizing about $\eta = 0$ gives

$$\dot{\eta} = A\,\eta,$$

where $A = \frac{\partial h}{\partial \eta}|_{\eta=0}$ is a 4×4 matrix. Analytical expressions for h and A are presented in Appendix I. To check the asymptotic stability of the system and calculate the controller gains, the eigenvalues of matrix A, which are the roots of its characteristic equation, need to be calculated. The characteristic equation for matrix A has the general form:

$$\begin{aligned} 0 = a\lambda^4 + \left(gKk_{dd}\right)\lambda^3 + \left(gKk_d + \beta + \gamma_1 k_s\right)\lambda^2 \\ + \left(gKk_p\right)\lambda + \left(\gamma_2 k_s\right), \end{aligned} \tag{13}$$

where

$$\begin{aligned} a &= c_1c_2 - \left(c_3\cos\left(q_2^d\right)\right)^2, \\ K &= c_4\sin\left(q_1^d\right)\left(c_2 + c_3\cos\left(q_2^d\right)\right) \\ &\quad - c_5\sin\left(q_1^d + q_2^d\right)\left(c_1 + c_3\cos\left(q_2^d\right)\right), \\ \beta &= g\left(c_3c_5\cos\left(q_2^d\right)\sin\left(q_1^d + q_2^d\right) - c_2c_4\sin\left(q_1^d\right)\right), \\ \gamma_1 &= c_1 + c_2 + 2c_3\cos\left(q_2^d\right), \\ \gamma_2 &= -g\left(c_4\sin\left(q_1^d\right) + c_5\sin\left(q_1^d + q_2^d\right)\right). \end{aligned} \tag{14}$$

According to the Routh–Hurwitz stability criterion, and because a is always a positive constant independent of the choice of gains, this system is stable if and only if

$$gKk_{dd} > 0\,, \quad \gamma_2 k_s > 0\,, \tag{15}$$

$$\left(gKk_d + \beta + \gamma_1 k_s\right) - \frac{k_p}{k_{dd}}a > 0\,, \tag{16}$$

$$gKk_p - \frac{gKk_{dd}\,\gamma_2 k_s}{\left(gKk_d + \beta + \gamma_1 k_s\right) - \frac{k_p}{k_{dd}}a} > 0\,. \tag{17}$$

It is obvious that the inequalities in (15) can be satisfied by choosing proper values for k_{dd} and k_s, if $\gamma_2 \neq 0$ and $K \neq 0$. Also it can be proved that under the same conditions, there always exist a set of values for k_d and k_p that satisfy (16) and (17), simultaneously.

Proof Let k_d be

$$k_d = \frac{\frac{k_p}{k_{dd}}a - \beta - \gamma_1 k_s + \delta}{gK}\,,$$

where δ is a positive number. Inequality (16) then simplifies to $\delta > 0$ and inequality (17) can be written as

$$gKk_p - \frac{gKk_{dd}\gamma_2 k_s}{\delta} > 0\,.$$

Therefore, by choosing a proper value for k_p which satisfies

$$Kk_p > \frac{k_{dd}\gamma_2 k_s}{\delta}K$$

both inequalities in (16) and (17) will be satisfied, simultaneously. □

Consequently, the Routh–Hurwitz stability criterion for the 2D balancer simplifies to

$$\gamma_2 \neq 0 \quad \text{and} \quad K \neq 0\,. \tag{18}$$

It will now be shown that $\gamma_2 = 0$ happens only when the CoM of the robot coincides with its first joint, which is called the neutral balance, and $K = 0$ only happens at desired configurations in which the robot is uncontrollable.

Neutral balance at $\gamma_2 = 0$: According to (10) we have

$$c_4^2 \cos\left(q_1^d\right)^2 = c_5^2 \cos\left(q_1^d + q_2^d\right)^2$$

at every balanced configuration, which implies

$$c_4^2 - c_5^2 = c_4^2 \sin\left(q_1^d\right)^2 - c_5^2 \sin\left(q_1^d + q_2^d\right)^2\,. \tag{19}$$

So if $c_4 \neq c_5$ then $c_4^2 - c_5^2 \neq 0$ and consequently

$$c_4 \left|\sin\left(q_1^d\right)\right| \neq c_5 \left|\sin\left(q_1^d + q_2^d\right)\right|\,. \tag{20}$$

Therefore, the only condition that permits $\gamma_2 = 0$ is $c_4 = c_5$; and the only value of q_2^d that satisfies both (10) and $\gamma_2 = 0$ when $c_4 = c_5$ is $q_2^d = \pm\pi$. In this case, the robot is in a neutral balanced configuration with its CoM at the rotation center of the first joint.

Uncontrollability at $\mathbf{K = 0}$: It is proved in Appendix II that if $K = 0$ at a desired configuration then the velocity gain (G_V) is also zero at that configuration. The velocity gain, which was introduced in Featherstone (2012), is a dimensionless measure that expresses an input–output relationship of an under-actuated robot in which the input is an impulse at the actuated joint and the output is the resulting change in the motion of the CoM. It differs from the CoM Jacobian in that the latter expresses a kinematic relationship between joint velocities and CoM velocities, whereas the former expresses the dynamic response to an impulse (or a force) at the actuated joint. In the present context, the velocity gain is the ratio $\Delta\dot{\phi}/\Delta\dot{q}_2$, in which $\Delta\dot{q}_2$ is the step change in $\dot{q}_2$ caused by an arbitrary nonzero impulse at joint 2, ϕ is the angle of the CoM relative to the x axis, measured at the support point (i.e., joint 1 axis), and $\Delta\dot{\phi}$ is the step change in $\dot{\phi}$ caused by the same impulse that caused $\Delta\dot{q}_2$. When the velocity gain is zero, it means that applying instantaneous torque to the actuator has no effect on the CoM of the robot. In other words, if G_V at a desired configuration is zero then the robot is intrinsically uncontrollable at that configuration.

So our proposed controller is able to stabilize the robot in any unstable balanced configurations except the neutral ones ($\gamma_2 = 0$) or those in which the robot is uncontrollable ($K = 0$).

5 Gain calculations

Given the stability analysis in the previous section, we can define an optimal controller by the pole placement method to be one that maximizes the speed of the slowest pole. In this case all poles of the closed loop system are placed at the same point which means that

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = -p\,,$$

where p is related directly to k_s as

$$\lambda_1\,\lambda_2\,\lambda_3\,\lambda_4 = p^4 = \frac{\gamma_2 k_s}{a}\,. \tag{21}$$

The relationship between k_s and p means that the value of k_s determines the overall rate at which the controller converges to the desired configuration. Increasing k_s will increase the rate of convergence, but at the cost of a reduced region of stability around the desired configuration and larger, more energetic movements of the robot (see Fig. 6). Once k_s has been chosen, the other gains must then be chosen so that (13) matches the polynomial

$$(\lambda + p)^4 = \lambda^4 + 4p\lambda^3 + 6p^2\lambda^2 + 4p^3\lambda + p^4 = 0\,. \tag{22}$$

With this choice of gains, all poles of the closed loop system are negative, so the system is asymptotically stable and the controller gains become

$$k_p = \frac{4\,p^3\,a}{gK}\,,\quad k_d = \frac{6\,p^2 a - \beta - \gamma_1 k_s}{gK}\,,\quad k_{dd} = \frac{4\,p\,a}{gK}\,. \tag{23}$$

6 Following a trajectory

As described in Sect. 3, making the robot move from one balanced configuration to another can easily be done by feeding

the desired value of q_2^d to the control law in (12). However, making the robot follow a prescribed trajectory is harder. We define an arbitrary motion trajectory for the 2D balancer to be an equation that specifies q_2^d as an explicit function of time. By 'arbitrary' we mean that the trajectory is not constrained to have any particular algebraic form. In particular, we do not require the trajectory to be a member of a special class of trajectories that the robot can follow exactly, such as the trajectories described by Berkemeier and Fearing (1999). Thus, the trajectory-following behavior of our controller is fundamentally different from that of Berkemeier and Fearing's controller in that our controller follows trajectories that are physically impossible to follow exactly without losing balance, implying that some degree of tracking error is physically necessary for maintaining balance and we expect the controller to generate an appropriate (small) tracking error automatically.

Clearly, there will exist trajectories that will cause the controller to lose balance. We have made no attempt to characterize these trajectories. Instead, the simulations in the next section consider only trajectories that the controller can follow successfully.

To implement trajectory following, we use the control law in (12), but replace L with $(L - L_d)$. L_d is the theoretical value of L assuming that the robot is perfectly following the desired trajectory at the current instant. Since the desired trajectory of the robot, which is determined by q_2^d, is a function of time then L_d is also a function of time depending on q_2^d and the desired velocity of the robot. So according to (4) we have

$$L_d = \left(c_1 + c_2 + 2c_3 \cos\left(q_2^d\right)\right)\dot{q}_1^d + \left(c_2 + c_3 \cos\left(q_2^d\right)\right)\dot{q}_2^d , \tag{24}$$

where $\dot{q}_2^d$ is computed from the reference trajectory and $\dot{q}_1^d$ is calculated from (3) assuming that $\dot{X} = 0$ and the robot is balanced.

During a trajectory-following motion of the robot, the gains of the controller are not constants but vary as a function of the desired configuration at each instant of the motion.

7 Simulations

The parameters for the balancer that are used in the simulations are the same as the "*good balancer*" in Featherstone (2012), and are listed in the middle column in Table 1. The right-hand column in this table lists the incorrect parameters that will be used later to test the controller's robustness to modeling errors.

To demonstrate the performance of the new controller, the results of three sets of simulations are presented in this section. All simulations have been done using Simulink. In the first set, presented in Sect. 7.1 below, it is assumed that there are no imperfections in the system and the controller has access to the exact model of the plant (i.e., the controller uses the same model parameters as the simulator). In this set of simulations, a continuous variable time step 4$^{\text{th}}$ order Runge–Kutta integrator (ode45) with maximum step size of 10ms is used. In the second and third sets, presented in Sects. 7.2 and 7.3, we add some imperfections to the system to model a variety of practical effects. Section 7.2 considers imperfections that do not affect the controller's calculation of the balanced configuration (the calculation of q_1^d from q_2^d via (10)), and Sect. 7.3 considers imperfections that do. The reason for this division is that the controller is more robust to the former than the latter. For these sets of simulations, the controller is modeled as a sampled-data system running at a servo rate of 1 kHz, and so a continuous variable time step 4th order Runge–Kutta integrator (ode45) with maximum step size of 1ms is used to simulate the dynamics of the robot, but the controller uses a first-order integrator for its own internal calculations.

Table 1 Model parameters used in the simulations

Parameter	Simulator	Controller
m_1	0.49	0.44
l_1	0.4	0.4
l_{c1}	0.1714	0.1543
I_1	0.0036	0.0032
m_2	0.11	0.1
l_2	0.6	0.6
l_{c2}	0.4364	0.4078
I_2	0.0043	0.0039

7.1 Perfect modeling

To show the performance of the controller in theory, simulations have been done for a system which is theoretically perfect: the controller uses the same model parameters as the simulator (so all of its model-dependent calculations are correct); it is given access to the actual values of the robot's state variables; and the actuator is an ideal torque source without any torque limits. The gains are calculated so as to place the poles of the closed-loop system on -7 at the desired configuration.

Figure 3 shows the balancer moving from a crouched position ($q_2 = -\frac{\pi}{2}$) to the upright balanced position ($q_2^d = 0$) and Fig. 4 shows the robot starting from the upright position and moving to a balanced crouching position ($q_2 = -\frac{\pi}{2}$). These figures also show the performance of our previous controller, as described in Azad and Featherstone (2012).

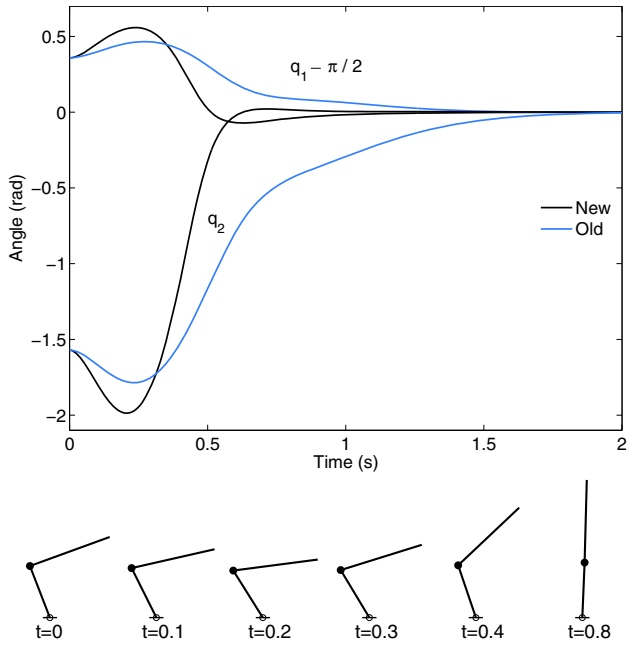

Fig. 3 Robot's straightening motion—perfect modeling

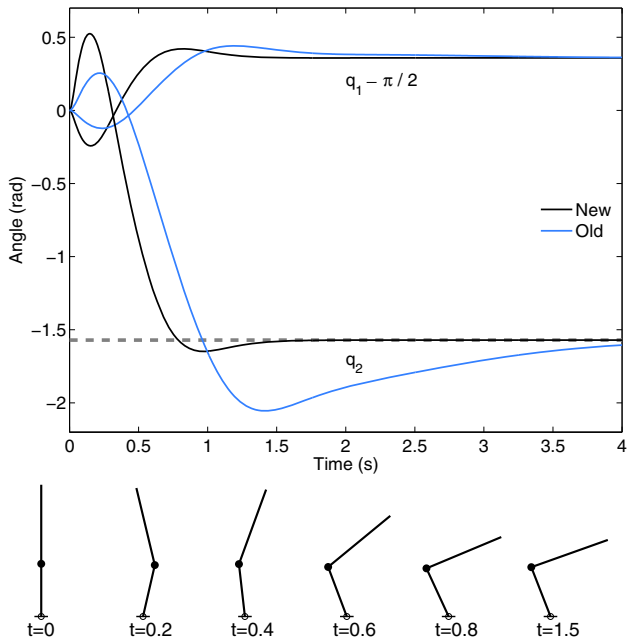

Fig. 4 Robot's crouching motion—perfect modeling

Comparing the two controllers, it is obvious that the new one is faster. The new controller takes about 1.2 s to complete the straightening motion, and 1.5 s to complete the crouching motion, compared with 1.8 s and >4 s for the old controller. Note that the stick diagrams show the robot's motion with the new controller.

Figure 5 shows the tracking response of the two controllers when the command for q_2 consists of two steps (which repeat the motions shown in more detail in Figs. 3 and 4), a linear ramp and a sine wave function. It can be seen that the new controller tracks the linear ramp and the sine wave accurately, after an initial transient, whereas the old controller follows these trajectories with a substantial phase lag. Though, the new controller overshoots slightly on the sine wave.

It is also noticeable in all Figs. 3, 4 and 5 that once the robot receives a command to follow, it first starts moving in the opposite direction of the command and then it quickly comes back to the right direction of the desired trajectory. This behavior is a property of all double-pendulum under-actuated robots with negative velocity gains, and tracking errors due to that behavior are physically unavoidable.

As can be seen in Fig. 5, the controller overshoots substantially at the beginning of the step commands at $t = 0$ s and $t = 5$ s. This is the price of a fast response that we see in this figure and is dependent on the locations of the poles. Figure 6 shows the effect of choosing different pole locations on the robot's straightening motion from a crouched ($q_2^d = -\frac{\pi}{2}$) to an upright position. As can be seen, more negative poles result in faster balancing maneuvers, but at the cost of larger overshoots and larger initial motions in the opposite direction. There is no single optimum location for the poles: it will depend on the dynamics of the robot mechanism, the torque available from the actuator and the user's preferred tradeoff between speed and overshoot.

7.2 Considering the imperfections

In the second set of simulations we demonstrate the performance of the controller in more realistic situations by including imperfections in the simulations. The imperfections considered in this subsection are those that do not affect the controller's calculation of the balanced configuration of the robot (i.e., the calculation of q_1^d from q_2^d via (10)). Specifically, the following imperfections are considered.

i. *Discrete execution* The controller is modeled as a sampled-data system that samples its inputs once per millisecond, performs calculations based on the values of these inputs and stored data from previous executions, and updates the outputs and stored data. The new outputs appear 1ms after the inputs are sampled, in order to model computation time delay.
ii. *Modeling error* To simulate errors in the controller's dynamic model of the robot, the controller uses a set of model parameters that differ by up to 10 % from the values used by the simulator. These values are listed in the right-hand column of Table 1. To preserve the accuracy of the controller's calculation of balanced configurations, these values have been chosen to have the same ratio

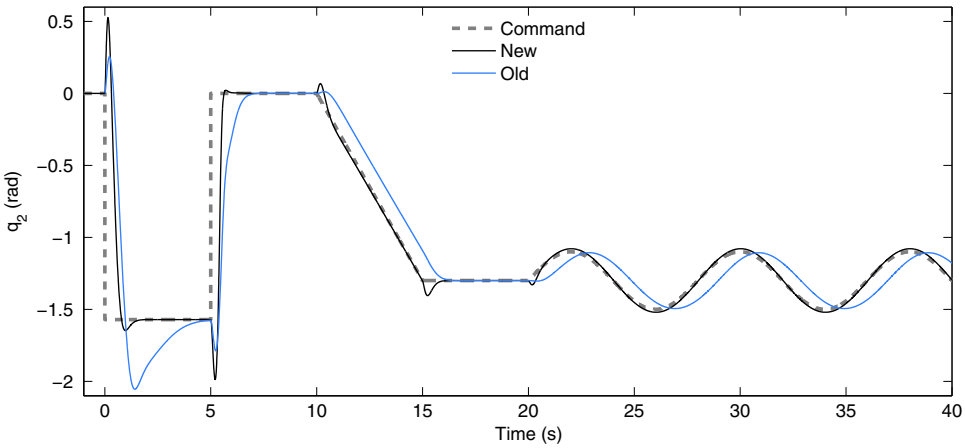

Fig. 5 Trajectory-tracking performance of the robot—perfect modeling

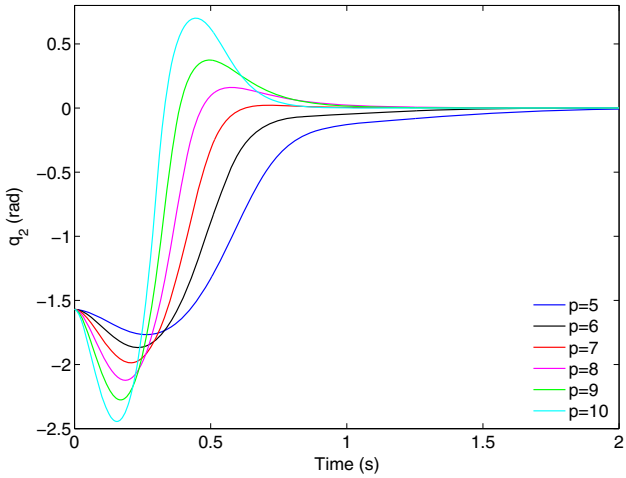

Fig. 6 Effect of different pole locations on straightening motion

c_4/c_5 as the parameters used by the simulator. We call the model used by the simulator the 'exact' model, and the one used by the controller the 'estimated' model.

iii. *Quantization* The controller reads the joint angles via two encoders that provide quantized readings of the joint angles. The encoder in the passive joint has a resolution of 8192 counts per revolution, and the encoder embedded in the motor that actuates joint 2 has a resolution of 512 counts per revolution. However, the motor is connected to the joint via a gearbox with a reduction ratio of 66, so the resolution at the joint is 66×512 counts per revolution.

iv. *Velocity estimation* The controller does not have access to the velocity state variables. Instead, it must estimate them from the quantized joint position readings. To do this, the controller uses the linear observer described in Harnefors and Nee (2000), which works as follows. If $\theta(n)$ is the angle measurement at the n^{th} time step, coming from an encoder, then the estimated velocity of that angle at the next time step is

$$\hat{\dot{\theta}}(n+1) = \hat{\dot{\theta}}(n) + T_s \rho^2 \Big(\theta(n) - \hat{\theta}(n)\Big), \tag{25}$$

where $\hat{\theta}(n)$ is an internal state variable that is updated using

$$\hat{\theta}(n+1) = \hat{\theta}(n) + T_s \Big(\hat{\dot{\theta}}(n) + 2\rho(\theta(n) - \hat{\theta}(n))\Big). \tag{26}$$

In the above equations, T_s is the time step, which is 1ms, and ρ is a parameter which is set to 200 in our simulations.

v. *DC motor model and velocity servo* – The actuator consists of a DC motor, a gearbox, and a velocity servo. The servo is implemented as a PI controller with proportional and integral gains of 15 and 200, respectively. Its input is the velocity command from the balance controller (see below), and its output is the voltage across the motor. The latter is limited to ±48 V, which imposes a speed-dependent torque limit on the actuator as a whole. The sampling rate is assumed to be so fast that the servo can be modeled as a continuous-time system. For the sake of simplicity, the servo is given the exact value of the motor's velocity.

The motor is modeled by the following equations:

$$T = K_t\, i \quad \text{and} \quad V - K_e \dot{\phi} = \mathcal{L}\frac{\mathrm{d}i}{\mathrm{d}t} + R\, i\,, \tag{27}$$

where V is the voltage across the motor, T is the output torque, $\dot{\phi}$ is the rotor velocity (computed from $\dot{q}_2$), i is the current through the motor (a state variable), and K_t, K_e, $\mathcal{L}$ and R are parameters listed in Table 2. These parameters were taken from the data sheet for part number 283860 from MaxonMotors.[1] This motor was chosen for no reason other than as a source of realistic parameter values.

The motor is connected to the joint via a zero-backlash gearbox with a reduction ratio of 66 and an efficiency

[1] http://www.maxonmotor.com (Accessed 18/5/2015).

Table 2 DC motor parameters

Stall torque	127 m Nm
Nominal torque	23.1 m Nm
No-load speed	12,900 rpm
Nominal speed	10,500 rpm
Resistance(R)	13.1 Ω
Inductance($\mathcal{L}$)	0.729 mH
Torque constant(K_t)	34.8 mN m A^{-1}
Rotor inertia	4.45 g cm^2
Speed constant (K_e)	274 rpm V^{-1}
Power	25 W

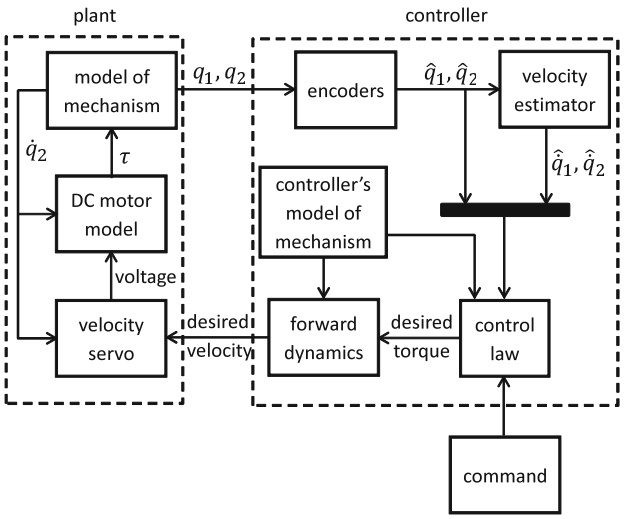

Fig. 7 Block diagram of the system with imperfections

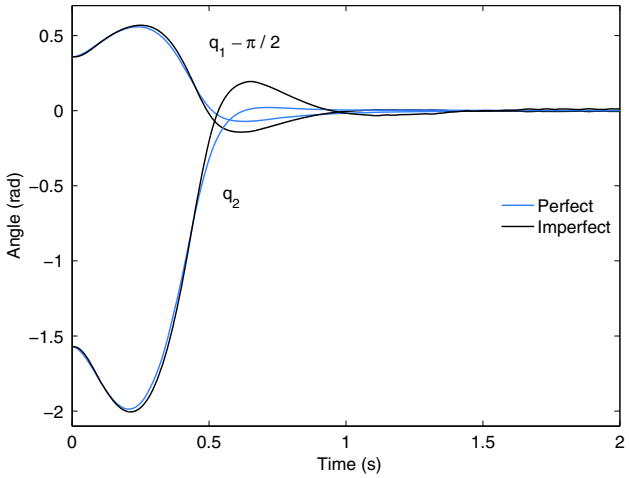

Fig. 8 Robot's straightening motion with imperfections

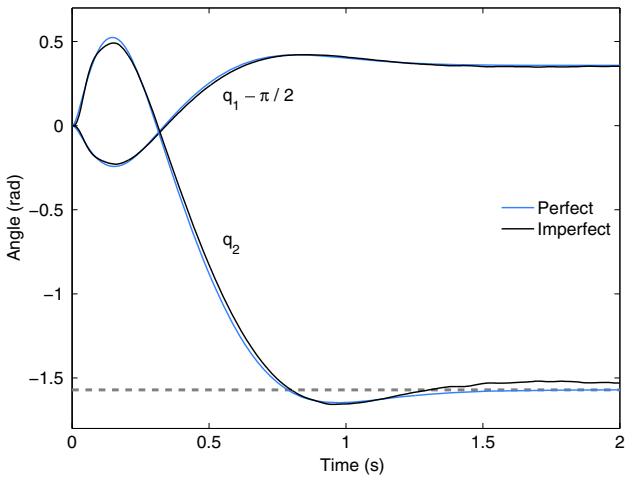

Fig. 9 Robot's crouching motion with imperfections

of 70 % (based on part number 166940 from MaxonMotors). The gearbox itself is assumed to have no inertia, but the motor's rotor inertia is incorporated into the equation of motion of the robot after being multiplied by the square of the gear ratio.

Figure 7 shows a schematic diagram of the subsystems used in the simulations. Those that are part of the plant are treated as continuous-time subsystems, whereas those that are part of the controller are treated as sampled-data subsystems with a sampling time of 1 ms. Strictly speaking, the encoders are part of the robot, but it is convenient to treat them as part of the controller.

According to the signal flow in Fig. 7, the plant outputs the joint angles (q_1 and q_2) to the encoder block, which passes the quantized values of the joint angles ($\hat{q}_1$ and $\hat{q}_2$) to the velocity estimator block. The control law block calculates the desired value of the torque on the basis of the estimated joint angles, the estimated angular velocities ($\hat{\dot{q}}_1$ and $\hat{\dot{q}}_2$) and the command signal. Then the forward dynamics block converts the desired torque to a desired acceleration and integrates it to obtain the desired velocity ($\dot{q}_2^d$) which is output to the velocity servo. Both the control law and forward dynamics blocks use the model parameters stored in the controller's model of the robot mechanism (i.e., the estimated model).

Figure 8 shows the balancing performance of the robot when it starts from a crouched position ($q_2 = -\frac{\pi}{2}$) and is aiming for the upright position. Figure 9 shows the crouching motion of the robot when it starts from the balanced upright position. Both figures also show the perfect-model responses for these two motions. It can be seen that there is more overshoot in the straightening motion and a small steady-state error in the crouching motion with imperfections. The latter is due to the modeling error, which causes the gravity compensation term (τ_g) to be calculated incorrectly in the crouched position but not in the upright position (where $\tau_g = 0$). Nevertheless, the controller's balancing performance is very nearly as good as in the perfect modeling case, which shows that the controller is robust to the kinds of imperfection considered in this subsection.

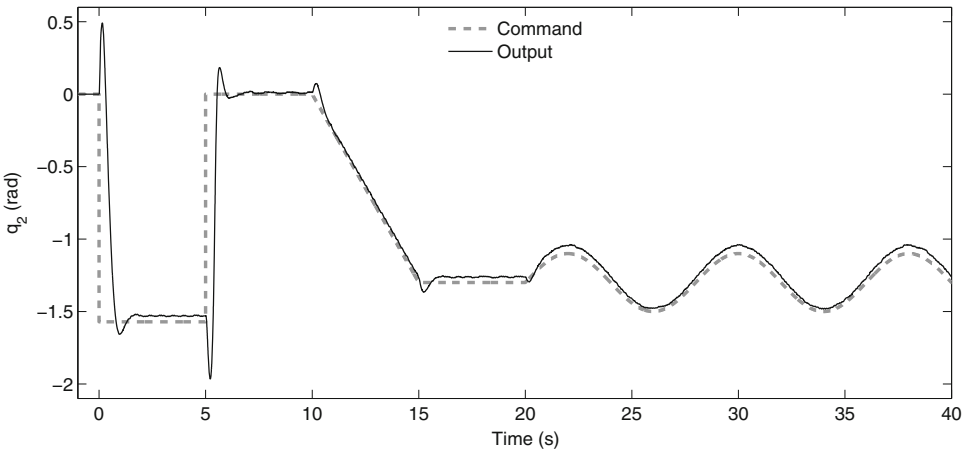

Fig. 10 Trajectory-tracking performance of the robot with imperfections

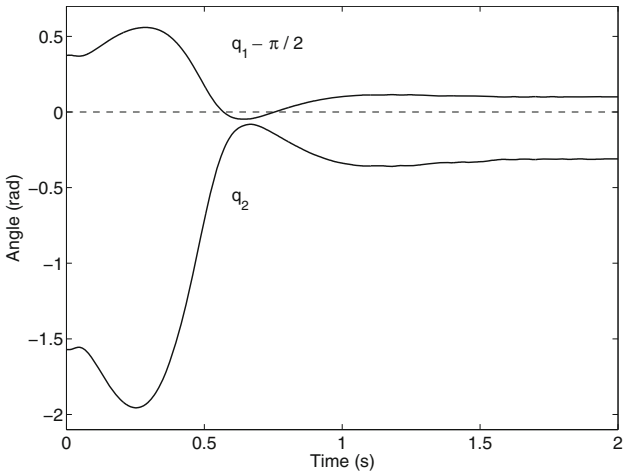

Fig. 11 Robot's straightening motion with imperfections and encoder bias

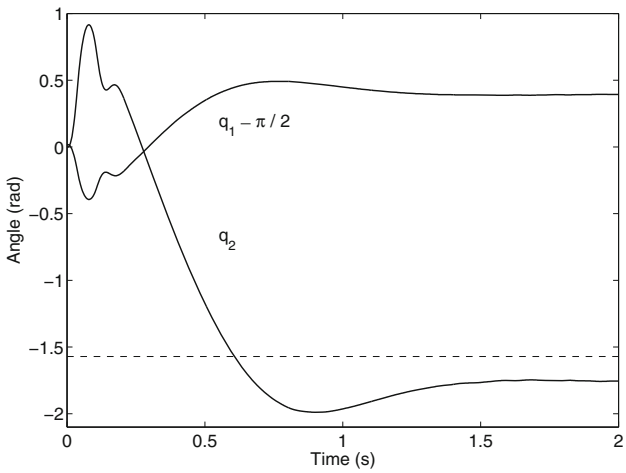

Fig. 12 Robot's crouching motion with imperfections and encoder bias

Figure 10 shows the trajectory tracking performance of the robot in the presence of the imperfections in the system. The robot follows the desired trajectory almost as accurately as in the perfect modeling case. However, there is a steady-state or offset error in every configuration other than the upright one, which is due to the error in the calculation of τ_g.

7.3 Imperfections that influence the balanced configuration

There are two types of error that influence the controller's calculation of the balanced configuration: errors in the dynamic model and bias in the encoders. As both types of error have the same effect, it is enough to consider just one of them. We therefore introduce this type of error by adding a small bias of 1° to the passive joint's encoder.

Figures 11 and 12 show the straightening and crouching motions of the robot in the presence of all imperfections, including the 1° bias. Both figures show significant steady-state errors. In the straightening motion there is almost a 6° error in the value of q_1 and 18° in q_2; while in the crouching motion the errors are 2° and 10.6°, respectively. Figure 12 also shows a wobble near the beginning, which is due to the actuator hitting its saturation limit.

So we can conclude that our proposed controller is robust to the kinds of imperfections which are likely to appear in a practical system except those that influence the controller's calculation of the robot's balanced configuration. In other words, the controller is sensitive to the estimation of the gravity direction.

8 Comparing with other control algorithms

In this section we compare the performance of our controller with three other balancing control algorithms in the literature: Spong's LQR controller[2] Spong (1995), Berkemeier and Fearing's linear controller Berkemeier and Fearing

[2] As already mentioned in the introduction section, this method is not specific to Spong and has been used by many other researchers as well. It is selected for comparison because of its wide use on the acrobot.

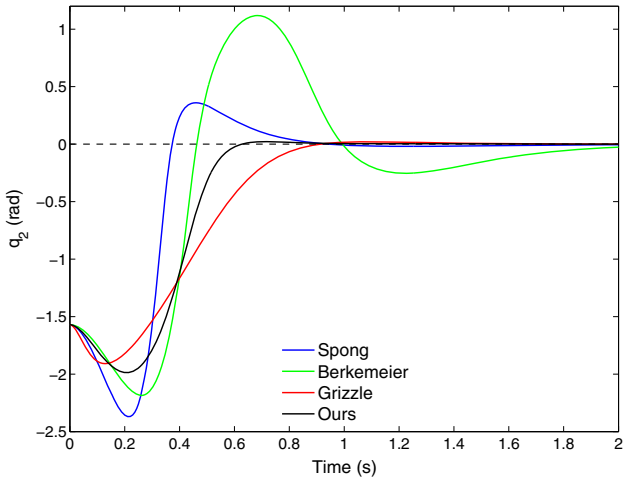

Fig. 13 Robot's straightening motion—perfect modeling

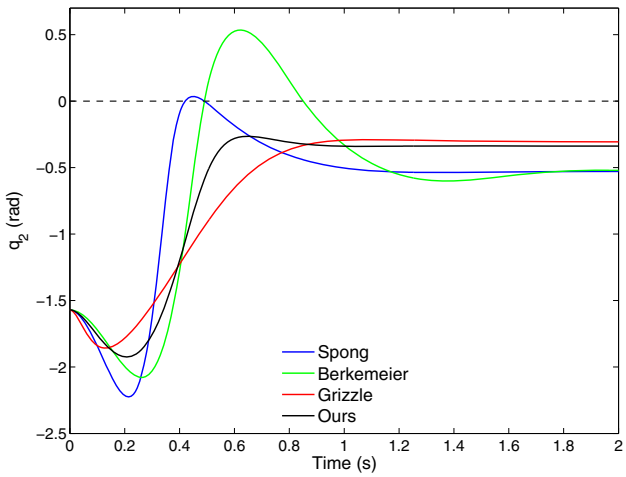

Fig. 14 Robot's straightening motion with 1° bias in the passive joint's encoder

(1999) and Grizzle's nonlinear controller Grizzle et al. (2005). First we compare the performance of all four controllers during the straightening motion of the robot when there are no imperfections in the system. Then we compare the balancing performance of the controllers when there is only one imperfection in the system, which is a 1° bias in the passive joint's encoder. At the end we compare the trajectory-tracking performance of our controller with the one from Grizzle's controller. We only consider Grizzle's controller in the last comparison since Berkemeier's controller only follows a set of special trajectories, and Spong's controller needs a proper objective function (i.e., Q and R) for the whole reference trajectory.

To make the comparisons as fair as possible, we have set the gains as follows. For Spong's LQR controller, we set $R = 100$ and Q to an identity matrix. These values were obtained by searching manually for values that gave good results. Thus, these particular values are probably not the best possible choice. Nevertheless, they are a good choice, and the controller itself is optimal for the given values of R and Q. For Berkemeier and Fearing's controller, we placed all of the poles at -5 which is the value used in their paper. We found by experiment that less negative values resulted in slower convergence, while more negative values resulted in instability. For Grizzle's controller we set the eigenvalues of the error equation to -7; and for our own controller we set the poles also to -7. The role of these two quantities appears to be essentially the same, and we found that both controllers behaved similarly with respect to the placement of poles and eigenvalues (i.e., like Fig. 6), so we decided that the fairest comparison would be to use the same value for both controllers.

There is also one more gain in Grizzle's controller which is denoted by K in Grizzle et al. (2005) and it is claimed that it can be any positive number in order to achieve local stability. However, we experienced in simulations that choosing high values for this gain may lead the controller to hit its singularity during the motions that we consider in this paper (i. e. straightening, crouching and trajectory-tracking). This is due to Eq. (44) in Grizzle et al. (2005) having zero denominator. Therefore, in simulations we set this gain to the highest value that still avoids singularity.

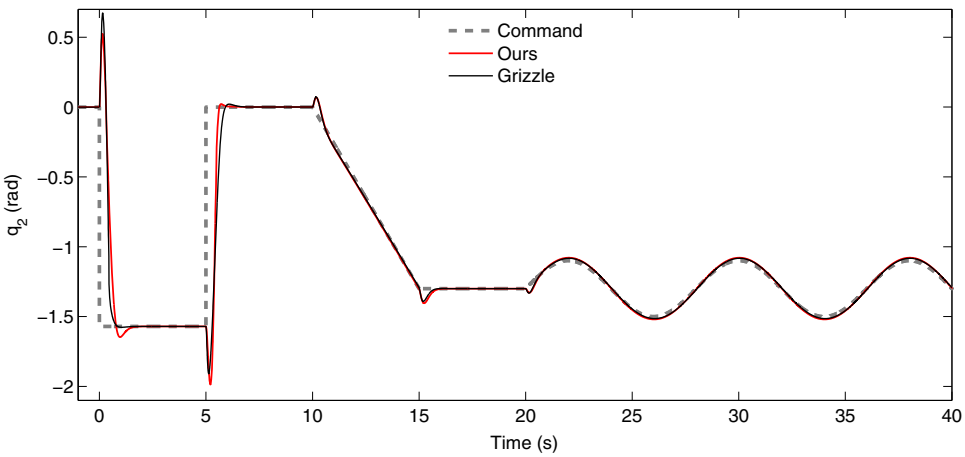

Fig. 15 Trajectory-tracking performance of the robot using our and Grizzle's controllers

Figure 13 shows how the robot converges to its upright balanced configuration starting from an initial crouched position ($q_2 = -\frac{\pi}{2}$) using each of the four control algorithms. It can be seen that, apart from Berkemeier and Fearing's linear controller, which has a large overshoot and a long settling time, the other controllers all perform reasonably well. Figure 14 shows the same balancing motion assuming that there is a one-degree bias in the first joint's encoder. As can be seen, there is a steady-state error with all four controllers. The magnitude of this error is the same in Spong's and Berkemeier's controllers, and it is roughly one third smaller for our and Grizzle's controllers.

Figure 15 compares the trajectory-tracking performance of our controller and Grizzle's controller following the same trajectory as shown in Fig. 5. It can be seen that both controllers behave almost the same in following the commanded trajectory.

9 Conclusion

In this paper a new and simple control algorithm for balancing an under-actuated planar robot is introduced. This controller, which is based on the angular momentum of the robot about its fixed point, is able to stabilize the robot in any unstable balanced configuration and follow an arbitrary trajectory without losing its balance. The balancing and trajectory tracking performance of the controller are shown by the results of the simulations. It is also shown in the simulations that the controller still performs well even in the presence of a significant amount of imperfections in the system. The new controller has been compared with three existing controllers and shown that it equals or outperforms all three. The new controller can be adapted to the problem of balancing a robot in 3D, and we have already implemented such a controller in simulation (Azad 2014; Azad and Featherstone 2014).

Acknowledgments This paper was partly supported by the European Commission, within the CoDyCo project (FP7–ICT–2011–9, No. 600716).

10 Appendix I

This appendix provides the analytical expression of $\dot{\eta} = h(\eta)$ and its linearized equation about $\eta = 0$. To express $\dot{\eta} = (\dot{q}_1, \dot{q}_2, \ddot{q}_1, \ddot{q}_2)$ as a function of η, it is required to compute $\ddot{q}_1$ and $\ddot{q}_2$ in terms of q and $\dot{q}$. Thus, we solve the motion Equation in (1) and (2) for $\ddot{q}_1$ and $\ddot{q}_2$ and obtain

$$\begin{aligned}\ddot{q}_1 = \frac{1}{d}\Big(& c_2c_3\sin(q_2)(\dot{q}_1+\dot{q}_2)^2 + c_3^2\sin(q_2)\cos(q_2)\dot{q}_1^2 \\ & - c_2c_4g\cos(q_1) + c_3c_5g\cos(q_2)\cos(q_1+q_2) \\ & - \tau(c_2+c_3\cos(q_2))\Big), \end{aligned} \tag{28}$$

$$\begin{aligned}\ddot{q}_2 = \frac{1}{d}\Big(& -c_3\sin(q_2)(c_2+c_3\cos(q_2))(\dot{q}_1+\dot{q}_2)^2 \\ & - c_3\sin(q_2)(c_1+c_3\cos(q_2))\dot{q}_1^2 \\ & + c_4g\cos(q_1)(c_2+c_3\cos(q_2)) \\ & - c_5g\cos(q_1+q_2)(c_1+c_3\cos(q_2)) \\ & + \tau(c_1+c_2+2c_3\cos(q_2))\Big), \end{aligned} \tag{29}$$

where $d = c_1c_2 - c_3^2\cos(q_2)^2$ is the determinant of the joint-space inertia matrix. Then, we replace τ from (12) into the above equations and obtain

$$\begin{aligned}\ddot{q}_1 = \frac{1}{d}\Big(& c_2c_3\sin(q_2)(\dot{q}_1+\dot{q}_2)^2 + c_3^2\sin(q_2)\cos(q_2)\dot{q}_1^2 \\ & - c_2c_4g\cos(q_1) - c_2c_5g\cos(q_1+q_2) \\ & - (c_2+c_3\cos(q_2))\Big(k_s(q_2^d - q_2) \\ & + k_p((c_1+c_2+2c_3\cos(q_2))\dot{q}_1 + (c_2+c_3\cos(q_2))\dot{q}_2) \\ & + k_{dd}g(c_4\sin(q_1)\dot{q}_1 + c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ & - k_dg(c_4\cos(q_1) + c_5\cos(q_1+q_2))\Big)\Big), \end{aligned} \tag{30}$$

and

$$\begin{aligned}\ddot{q}_2 = \frac{1}{d}\Big(& -c_3\sin(q_2)(c_2+c_3\cos(q_2))(\dot{q}_1+\dot{q}_2)^2 \\ & - c_3\sin(q_2)(c_1+c_3\cos(q_2))\dot{q}_1^2 \\ & + g(c_2+c_3\cos(q_2))(c_4\cos(q_1) + c_5\cos(q_1+q_2)) \\ & + (c_1+c_2+2c_3\cos(q_2))\Big(k_s(q_2^d - q_2) \\ & + k_p((c_1+c_2+2c_3\cos(q_2))\dot{q}_1 + (c_2+c_3\cos(q_2))\dot{q}_2) \\ & + k_{dd}g(c_4\sin(q_1)\dot{q}_1 + c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ & - k_dg(c_4\cos(q_1) + c_5\cos(q_1+q_2))\Big)\Big). \end{aligned} \tag{31}$$

To calculate matrix A, we linearize (30) and (31) about the equilibrium point $\eta = 0$. Note that, in the equilibrium point, we have $X = 0$ and therefore,

$$c_4\cos\left(q_1^d\right) + c_5\cos\left(q_1^d + q_2^d\right) = 0\,. \tag{32}$$

Hence, $A = \frac{\partial h}{\partial \eta}|_{\eta=0}$ will be

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\partial \ddot{q}_1}{\partial q_1}|_{\eta=0} & \frac{\partial \ddot{q}_1}{\partial q_2}|_{\eta=0} & \frac{\partial \ddot{q}_1}{\partial \dot{q}_1}|_{\eta=0} & \frac{\partial \ddot{q}_1}{\partial \dot{q}_2}|_{\eta=0} \\ \frac{\partial \ddot{q}_2}{\partial q_1}|_{\eta=0} & \frac{\partial \ddot{q}_2}{\partial q_2}|_{\eta=0} & \frac{\partial \ddot{q}_2}{\partial \dot{q}_1}|_{\eta=0} & \frac{\partial \ddot{q}_2}{\partial \dot{q}_2}|_{\eta=0} \end{bmatrix}, \tag{33}$$

where

$$\frac{\partial \ddot{q}_1}{\partial q_1}|_{\eta=0} = \frac{\gamma_2}{a}\left(k_d\left(c_2 + c_3\cos\left(q_2^d\right)\right) - c_2\right)$$
$$\frac{\partial \ddot{q}_1}{\partial q_2}|_{\eta=0} = \frac{1}{a}\Big(- k_d g\left(c_2 + c_3\cos\left(q_2^d\right)\right)c_5\sin\left(q_1^d + q_2^d\right) + k_s\left(c_2 + c_3\cos\left(q_2^d\right)\right) + c_2 c_5 g\sin\left(q_1^d + q_2^d\right)\Big)$$
$$\frac{\partial \ddot{q}_1}{\partial \dot{q}_1}|_{\eta=0} = \frac{1}{a}\left(k_{dd}\gamma_2 - k_p\gamma_1\right)\left(c_2 + c_3\cos(q_2^d)\right)$$
$$\frac{\partial \ddot{q}_1}{\partial \dot{q}_2}|_{\eta=0} = -\frac{1}{a}\Big(k_p\left(c_2 + c_3\cos\left(q_2^d\right)\right)^2 + k_{dd} g\left(c_2 + c_3\cos\left(q_2^d\right)\right)c_5\sin\left(q_1^d + q_2^d\right)\Big)$$
$$\frac{\partial \ddot{q}_2}{\partial q_1}|_{\eta=0} = \frac{\gamma_2}{a}\left(c_2 + c_3\cos\left(q_2^d\right) - k_d\gamma_1\right)$$
$$\frac{\partial \ddot{q}_2}{\partial q_2}|_{\eta=0} = \frac{1}{a}\Big((k_d g c_5\sin\left(q_1^d + q_2^d\right) - k_s\right)\gamma_1 - \left(c_2 + c_3\cos\left(q_2^d\right)\right)c_5 g\sin\left(q_1^d + q_2^d\right)\Big)$$
$$\frac{\partial \ddot{q}_2}{\partial \dot{q}_1}|_{\eta=0} = \frac{\gamma_1}{a}\left(k_p\gamma_1 - k_{dd}\gamma_2\right)$$
$$\frac{\partial \ddot{q}_2}{\partial \dot{q}_2}|_{\eta=0} = \frac{\gamma_1}{a}\Big(k_p(c_2 + c_3\cos\left(q_2^d\right)\right) + k_{dd} g c_5\sin\left(q_1^d + q_2^d\right)\Big)$$

11 Appendix II

This appendix proves the statement that at any desired unstable balanced configuration of the 2D balancer robot other than a neutral balanced configuration, $K = 0$ if and only if the velocity gain (G_V) is zero. Neutral balanced configurations are excluded because G_V as defined in Featherstone (2012) is not defined at such configurations.

The balance condition in (10) can be written in the form

$$r\cos(\phi)\cos\left(q_1^d\right) - r\sin(\phi)\sin\left(q_1^d\right) = 0, \tag{34}$$

where

$$r\cos(\phi) = c_4 + c_5\cos\left(q_2^d\right)$$

and

$$r\sin(\phi) = c_5\sin\left(q_2^d\right),$$

which implies

$$r = \left(c_4^2 + c_5^2 + 2c_4c_5\cos\left(q_2^d\right)\right)^{1/2}.$$

Note that $r = 0$ only if $c_4 = c_5$ and $\cos(q_2^d) = -1$, which is the condition for neutral balance, so we can exclude this possibility from consideration. Given that $r \neq 0$, the balance condition in (34) simplifies to

$$\cos\left(q_1^d + \phi\right) = 0.$$

This equation has two solutions, but the correct solution for an unstable balanced configuration is $q_1^d = \pi/2 - \phi$. From this solution we get

$$\sin\left(q_1^d\right) = \cos(\phi) = \left(c_4 + c_5\cos\left(q_2^d\right)\right)/r \tag{35}$$

and

$$\sin\left(q_1^d + q_2^d\right) = \cos(\phi)\cos\left(q_2^d\right) + \sin(\phi)\sin\left(q_2^d\right) = \left(c_5 + c_4\cos\left(q_2^d\right)\right)/r. \tag{36}$$

Substituting (35) and (36) into the expression for K in (14), and simplifying the result, gives

$$rK = c_2c_4^2 - c_1c_5^2 + \left(c_3\left(c_4^2 - c_5^2\right) + \left(c_2 - c_1\right)c_4c_5\right)\cos\left(q_2^d\right). \tag{37}$$

According to Eq. 5 in Featherstone (2012), the velocity gain for a 2D balancer is

$$G_V(q_2) = \frac{c_y^2 + \frac{m_2 c_x c_{2x}}{m_1 + m_2}}{c_x^2 + c_y^2} - \frac{H_{12}}{H_{11}}, \tag{38}$$

where

$$H_{11} = c_1 + c_2 + 2c_3\cos(q_2),$$
$$H_{12} = c_2 + c_3\cos(q_2),$$
$$c_x = (c_4 + c_5\cos(q_2))/c,$$
$$c_y = c_5\sin(q_2)/c,$$
$$c_{2x} = l_{c2}\cos(q_2) \quad \text{and} \quad c_{2y} = l_{c2}\sin(q_2).$$

So G_V at a desired configuration can be written as

$$G_V(q_2^d) = \frac{c_5^2 + c_4c_5\cos\left(q_2^d\right)}{c_4^2 + c_5^2 + 2c_4c_5\cos\left(q_2^d\right)} - \frac{c_2 + c_3\cos\left(q_2^d\right)}{c_1 + c_2 + 2c_3\cos\left(q_2^d\right)}. \tag{39}$$

Note that the first denominator is nonzero because it is equal to r^2, and the second denominator is nonzero because it is the

first element of the robot's joint-space inertia matrix, which is a positive-definite matrix. If we multiply both sides of (39) by both denominators then we get, after some simplification,

$$r^2 H_{11} G_V(q_2^d) = c_1 c_5^2 - c_2 c_4^2 + \cos\left(q_2^d\right)\left(c_3\left(c_5^2 - c_4^2\right) + \left(c_1 - c_2\right)c_4 c_5\right). \quad (40)$$

On comparing (40) with (37) it can be seen that $K = -r H_{11} G_V$, which proves that $K = 0$ if and only if $G_V = 0$.

References

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., & Featherstone, R. (2012). Angular momentum based controller for balancing an inverted double pendulum, RoManSy 19-Robot Design, Dynamics and Control, pp. 251–258, Paris, France, June 12–15.

Azad, M., & Featherstone, R. (2013). Balancing and hopping motion of a planar hopper with one actuator. *IEEE International Conference Robotics and Automation* (pp. 2027–2032). Karlsruhe, Germany, May 6–10.

Azad, M., & Featherstone, R. (2014). Balancing control algorithm for a 3D under-actuated robot. *Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems* (pp. 3233–3238). Chicago, IL, September 14–18.

Berkemeier, M. D., & Fearing, R. S. (1999). Tracking fast inverted trajectories of the underactuated acrobot. *IEEE Transactions Robotics and Automation*, *15*(4), 740–750.

Featherstone, R. (2012). Analysis and design of planar self-balancing double-pendulum robots, RoManSy 19-Robot Design, Dynamics and Control, pp. 259–266, Paris, France, June 12–15.

Formal'skii, A. M. (2006). On stabilization of an inverted double pendulum with one control torque. *International Journal Computer and Systems Sciences*, *45*(3), 337–344.

Goswami, A., & Kallem, V. (2004). Rate of change of angular momentum and balance maintenance of biped robots. *IEEE Interantional Conference Robotics and Automation* (pp. 3785–3790). New Orleans, LA.

Grizzle, J. W., Moog, C. H., & Chevallereau, C. (2005). Nonlinear control of mechanical systems with an unactuated cyclic variable. *IEEE Transactions Automatic Control*, *50*(5), 559–576.

Harnefors, L., & Nee, H. P. (2000). A general algorithm for speed and position estimation of AC motors. *IEEE Transactions Industrial Electronics*, *47*(1), 77–83.

Hauser, J., & Murray, R. M. (1990). Nonlinear controllers for nonintegrable systems: the acrobot example. *American Control Conference* (pp. 669–671). SanDiego, CA, May 23–25.

Inoue, A., Deng, M., Hara, S., & Henmi, T. (2007). Swing-up and stabilizing control system design for an acrobot. *Proceeding IEEE International Conference Networking, Sensing and Control* (pp. 559–561), London, UK, April 15–17.

Lai, X., Wu, Y., She, J., & Wu, M. (2005). Control design and comprehensive stability analysis of acrobots based on Lyapunov functions. *Journal of Central South University of Technology*, *12*(1), 210–216.

Lauwers, T. B., Kantor, G. A., & Hollis, R. L. (2006). A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. *Proceeding IEEE International Conference Robotics and Automation* (pp. 2884–2889). Orlando, FL, May 15–19.

Lee, S. H., & Goswami, A. (2012). A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, *33*(4), 399–414. Springer.

Macchietto, A., Zordan, V., & Shelton, C. R. (2009). Momentum control for balance. *ACM Transactions Graphics*, *28*(3), 80–87.

Olfati-saber, R. (2000). Control of underactuated mechanical systems with two degrees of freedom and symmetry. *Proceeding American Control Conference*. pp. 4092–4096.

Raibert, M. H. (1986). *Legged robots that balance*. Cambridge, MA: The MIT Press.

Segway Inc. (2015). Personal Transporter. http://www.segway.com, Accessed January 2015.

Spong, M. W. (1995). The swing up control problem for the acrobot. *IEEE Control Systems*, *15*(1), 49–55.

Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., & Morris, B. (2007). *Feedback control of dynamic bipedal robot locomotion*. Boca Raton, FL: CRC Press.

Xin, X., & Yamasaki, T. (2012). energy-based swing-up control for a remotely driven acrobot: theoretical and experimental results. *IEEE Transactions Control Systems Technology*, *20*(4), 1048–1056.

Yamakita, M., Yonemura, T., Michitsuji, Y., & Luo, Z. (2002). Stabilization of acrobot robot in upright position on a horizontal bar. *IEEE Internatioanl Conference Robotics and Automation*. (pp. 3093–3098), Washington, DC, May 11–15.

Yonemura, T., & Yamakita, M. (2004). Swing up control problem of acrobot based on switched output functions. *Proceeding SICE Annual Conference*. (pp. 1909–1914). Sapporo, Japan, August 4–6.

Morteza Azad Dr. Azad received his Bachelor's degree from Amirkabir University of Technology, Iran and his Master's degrees from Sharif University of Technology, Iran, in 2004 and 2006, respectively, both in Mechanical Engineering. He obtained his Ph.D. in Robotics from the Australian National University in 2014. Currently, he is a research fellow at the school of Computer Science, University of Birmingham, UK. His research interests include design and control of under-actuated robots, humanoid robotics, dynamics simulation and modeling compliant contacts.

Roy Featherstone Dr. Featherstone obtained his Bachelor's degree in Mathematics with Electronics from Southampton University in 1979, and his Ph.D. in Artificial Intelligence from Edinburgh University in 1984. He spent more than 6 years working in industry, first for a small start-up company in the UK, and then for Philips Electronics in the USA. In 1992 he moved to Oxford University to take up an EPSRC Advanced Research Fellowship. From 1998 to 2001 he was a Lecturer in Computer Science at the University of Wales, Aberystwyth, and from 2001 to 2011 he worked for the Australian National University, holding positions in both Computer Science and Engineering. He is currently a Visiting Professor at the Italian Institute of Technology. He is also a Fellow of the IEEE. Dr. Featherstone is the inventor of the articulated-body algorithm, as well as several other algorithms for rigid-body dynamics, and he has written two books on the subject. His current research interests include the design and control of highly athletic robots that can hop high, balance on a point, and perform a variety of other difficult manoeuvres.