Atlas Balance Control

Wyatt Newman

March 31, 2014

Introduction: These notes describe theory for Atlas balance control. The present analysis considers a simplified, 2-DOF, planar, underactuated system known as an "acrobot." See, e.g. http://www.youtube.com/watch?v=sMZRnE3q72c [1] for an animation of acrobot control. Some useful background may be found in: http://users.cms.caltech.edu/~murray/preprints/erl-M91-46.pdf [2].

The acrobot is a 2-link, planar robot with a motor at the "elbow" joint, but no motor at the "shoulder" joint. Drawing on figures and equations from reference [2], the acrobot is shown in Fig 1. This figure shows the definitions of angles 1 and 2, measured with respect to (0,0) = (q1,q2) corresponding to standing straight up. This figure simplifies the acrobot to describe it as having concentrated masses at the ends of the links, as well as ignoring rotational inertia. However, this is easily generalized.

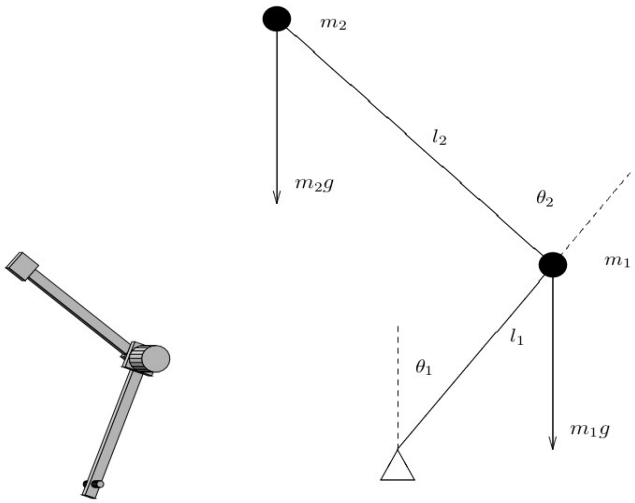

Illustration 1: Acrobot schematic illustration

The relevance to Atlas is that, if the robot is falling sideways (e.g., when attempting to balance on one foot), the ankle torques alone are insufficient to regain balance once the robot's center of pressure lies outside the polygon of the footprint.

Atlas's balance control is most challenging for lateral motion (left/right) while standing on one foot. Leaning by as little as approximately 0.07 rad to the left or right will place the center of mass outside the footprint polygon. Regaining balance then requires involving more joints.

To exaggerate the balance challenge, one may analyze the case for which the ankle torques are zero, and thus balance must be performed using other body joints. In particular, consider only the third torso joint being involved (joint back_bkx, corresponding to bending at the waist left/right about Atlas' pelvis x axis). Assume zero torque on the stance ankle and all other joints held frozen at fixed angles. This situation would be equivalent to the acrobot, in which the system is underactuated (one active actuator only) and the dynamics includes only two links (the lower body and the upper body).

It should be noted that the analogy between Atlas balancing and the acrobot differs in terms of large rotations. The acrobot is capable of continuous rotations of both joints 1 and 2, and this can be used for "swing-up" control, enabling the acrobot to go from a stable pose (pendulum hanging down) to

an upright pose. For Atlas, however, |q1|> pi/2 is impossible, as the robot would be lying sideways at |q1|=pi/2. Further, rotation of the back joint (q2, for the acrobot analogy) is limited to approximately +/- 0.6 rad. Swing-up control is not relevant for Atlas balancing. What is important is to recover balance (q1,q2) = (0,0) after a disturbance without falling over or violating joint limits or torque limits.

Acrobot dynamics: Using the notation of reference [2], the dynamics of the acrobot can be described as:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \begin{pmatrix} 0 \\ \tau \end{pmatrix}$$

Where the terms M, C and G are defined as:

$$M(\theta) = \begin{bmatrix} a + b + 2c\cos\theta_2 & b + c\cos\theta_2 \\ b + c\cos\theta_2 & b \end{bmatrix} \quad (1)$$

$$C(\theta,\dot{\theta}) = \begin{bmatrix} -c\sin\theta_2\,\dot{\theta}_2 & -c\sin\theta_2(\dot{\theta}_1 + \dot{\theta}_2) \\ c\sin\theta_2\,\dot{\theta}_1 & 0 \end{bmatrix} \quad (2)$$

$$G(\theta) = \begin{bmatrix} -d\sin\theta_1 - e\sin(\theta_1 + \theta_2) \\ e\sin(\theta_1 + \theta_2) \end{bmatrix} \quad (3)$$

The terms a through e in the above are constants, comprised of combinations of the following 7 parameters.

- L1: the distance from the ankle joint to the back_bkx joint
- m1: the mass of the lower body, from joint back_bkx to stance ankle joint leg_akx (left or right)
- Lc1: the location of the center of mass of the lower body, measured from the stance ankle
- Ic1: the rotational inertia of the lower body about the lower body's center of mass.
- m2: the mass of the upper body, from joint back_bkx
- Lc2: the location of the center of mass of the upper body, measured from the back_bkx joint
- Ic2: the rotational inertia of the upper body about the upper body's center of mass.

These parameters must be estimated to obtain a dynamic model of Atlas for balance control. With respect to these parameter definitions, the constants a through e in equations (1) through (3) are:

$a = I_{c1} + m_1 * L_{c1}^2 + m_2 * L_1^2$
$b = I_{c2} + m_2 * L_{c2}^2$
$c = m_2 * L_1 * L_{c2}$
$d = -m_1 * g * L_{c1} - m_2 * g * L_1$
$e = -m_2 * g * L_{c2}$

The definitions for *a* through *e* above differ from those in [2]. The definitions above are slightly more general than those in [2], as the above definitions include rotational inertias and allow for centers of masses (Lc) located arbitrarily along the respective links.

An important observation regarding the inertia matrix M is that M is symmetric and M is always invertible. This will be true regardless of the values of the mass property parameters (provided they are physically possible, e.g. m>0).

Again drawing from reference [2], the acrobot has a set of poses corresponding to unstable equilibria, as illustrated in Fig 2. These poses correspond to setting the joint-2 torque, trq2, equal and opposite to the g(2) term, and the combinations of q1 and q2 angles result in g(1)=0.

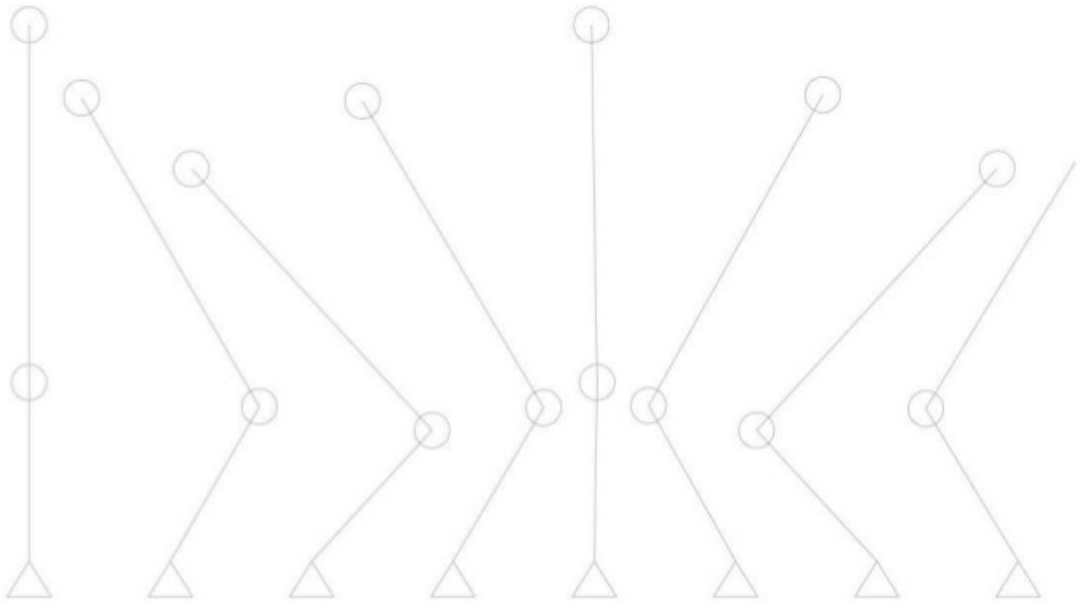

Illustration 2: unstable equilibria poses: g(1) = 0

That is, if q2 is frozen, then there is some angle q1 for which the robot almost balances (although this is an unstable equilibrium). These poses correspond to states for which g(1) =0. With respect to any such pose, if q2 is perturbed to be more positive (leans more to the left), then the robot q1 will accelerate positive (i.e., will fall to the left), and vice versa. The equilibrium curve of q1 vs q2 for g(1)=0 is important to know, since gravity must be used to help rebalance the acrobot.

By analogy, we may refer to the stance ankle joint of Atlas as q1and the back_bkx back joint angle as q2. Fig 3 shows an equilibrium pose for simulated Atlas, analogous to Fig 2. In this example, the angles of the right ankle and of the back_bkx joints are (-0.048, 0.3), which result in the robot's center of gravity directly over the right ankle. From such a pose, Atlas is stable using only PD control of the stance ankle joint.

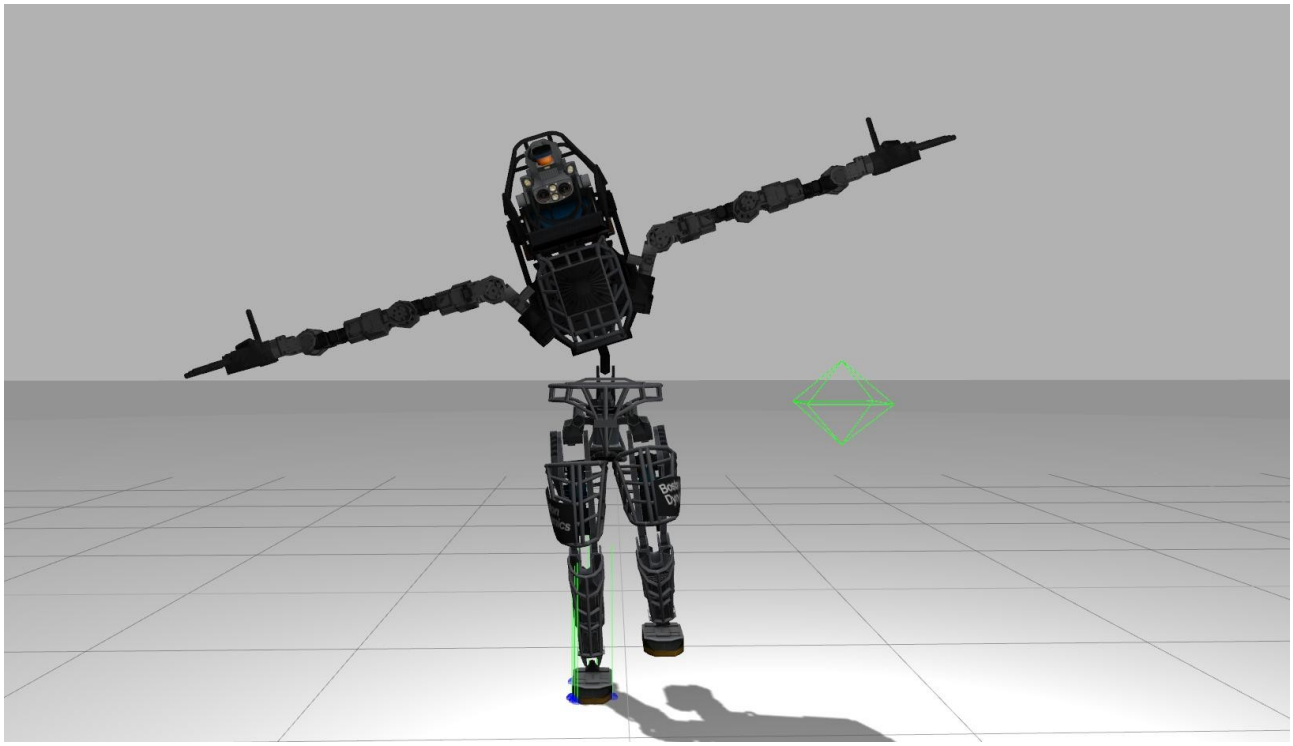

Illustration 3: drcsim balancing on right foot. By bending at the waist, center of gravity is over the right ankle.

Additional combinations of joint angles (ankle, waist)=(-q1,q2) for equilibrium are plotted in Fig 5. Note that the definition of positive ankle rotation is the negative of the definition of positive q1 acrobot rotation. These angles fit a linear approximation of: $q2 = 4*(-q1 + 0.125)$. If Atlas can be brought to be still at any such pose, then he can maintain his balance at that pose, or he can move to any other pose on this equilibrium line and maintain balance.

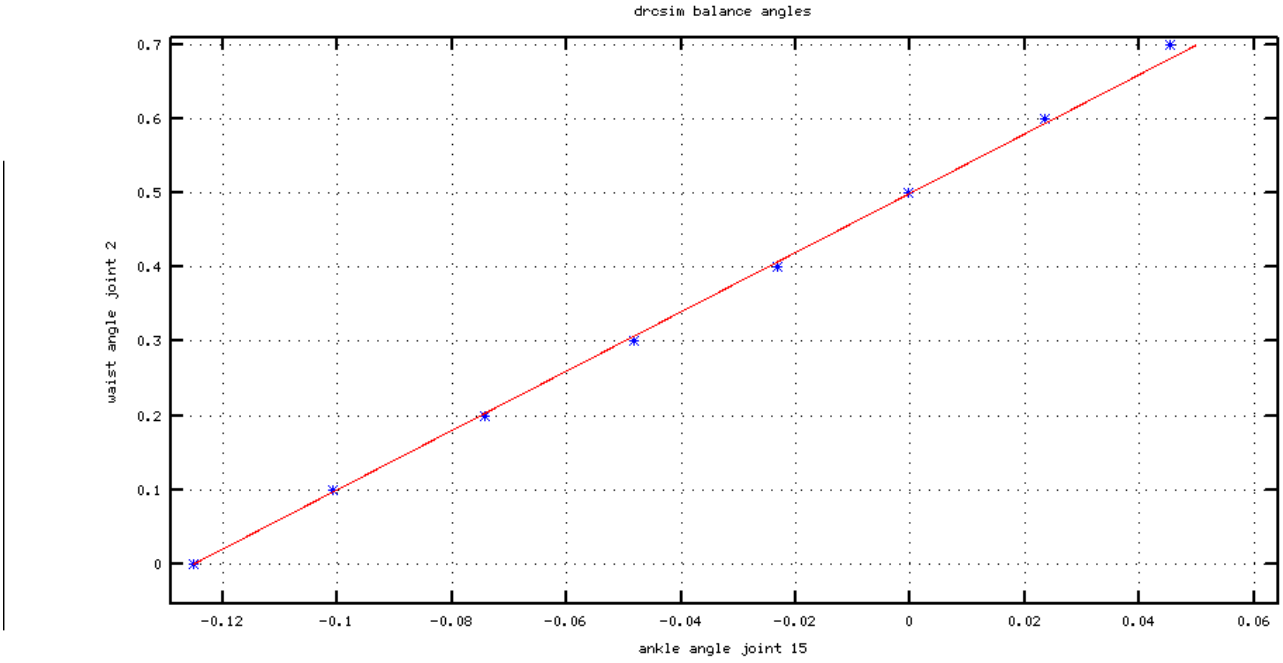

Illustration 4: drcsim stable poses balancing on right foot: ankle angle vs waist angle

Momentum and Displacement Coordinates: Control of the acrobot (and by extension, Atlas) can be thought of in terms of two sub-systems: one that relies on trq2 to eventually straighten the elbow (or the "waist" in the case of Atlas), and another subsystem that depends on gravity to reach a pose as in Fig 2 (or any of the poses in Fig 4). For Atlas control, since one can get at least limited torque from the ankle, the primary objective is to drive Atlas to a state at rest—i.e. (q1dot, q2dot) = (0,0) for which g(1) = 0 . For drcsim, g(1) = 0 at any (q2,q1) such that q2 = -4*(q1 - 0.125). More generally, the gravity term g(1) may be approximated as a linear function of q1 and q2:

(4) $g(1) = K_{g1}*q1 + K_{g2}*q2$

The above approximation is intended to be valid over the expected range of motion of q1 and q2.

An illustrative example will be used here, for which the kinematic parameters are defined as: m1 = m2 = 1; L1 =1; Lc1=Lc2 = 0.5, Ic1 = Ic2 = (1/12)m*L^2 = 1/12.

Figure 5 shows curves (in blue) of (q1,q2) that correspond to constant g(1). In this figure, lines of constant g(1) are shown for increments of 1N-m of g(1). The lines in red are a linear approximation of the function $g1(q1,q2) = K_{g1}*q1 + K_{g2}*q2 = -20*q1 -4.8*q2$

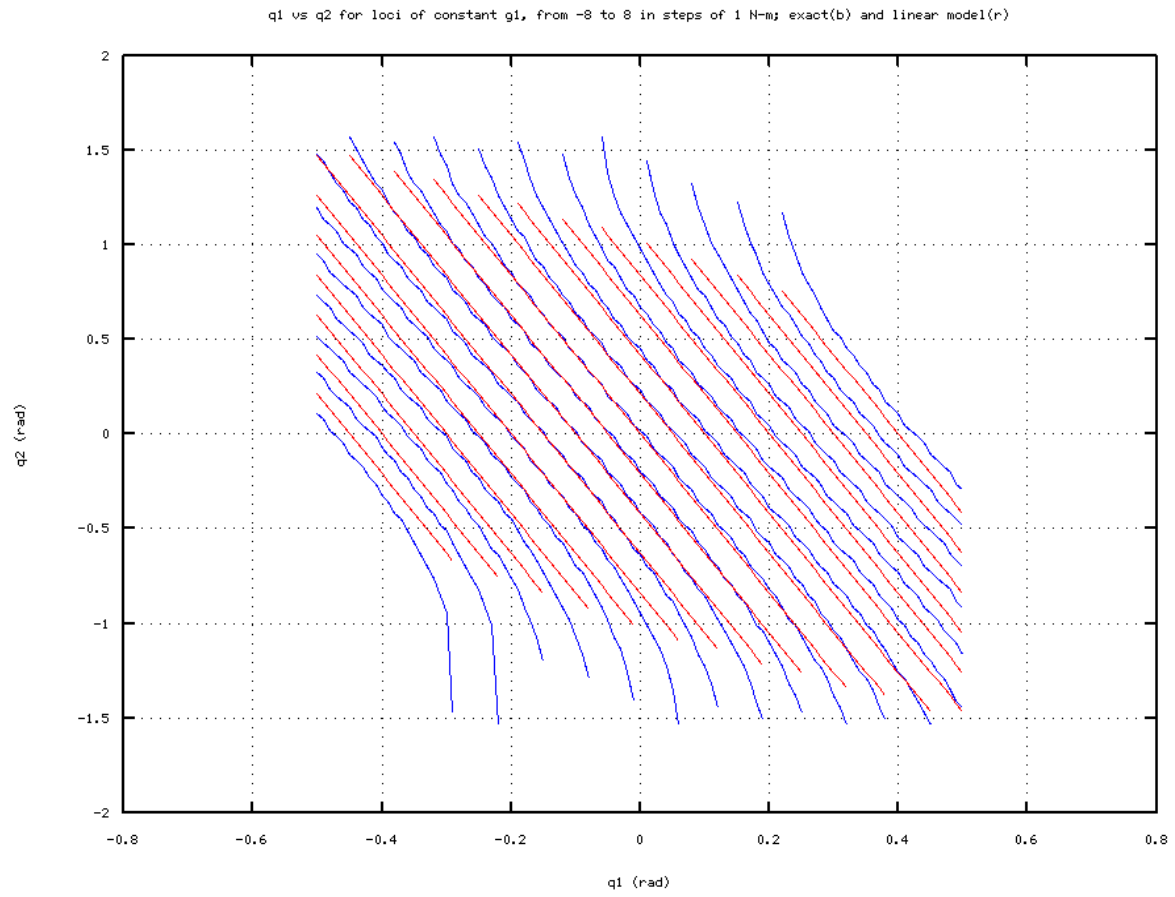

Illustration 5: Acrobot gravity load analysis and linear model fit

The linear approximation is reasonable for values of g1 between approximately +/- 5N-m. This approximation will simplify the controller design significantly.

Another simplification to be realized is that the angular momentum of the system about the origin (joint 1) does not depend on the torque of the actuator. The angular momentum of the system, h,

can be expressed as:

(5) h = M(1,1)*q1dot + M(1,2)*q2dot

Differentiating h with respect to time (after some algebra) yields the following simple relationship:

(6) dh/dt = -g(1)

That is, the total system angular momentum depends only on the gravity loading g(1), not on the actuator torque, trq2. Equation (6) is an exact relationship, not an approximation. This reflects the fact that internal forces and torques cannot influence a system's overall momentum nor angular momentum.

A noteworthy special case is when q2dot=0 and q2ddot=0 (i.e. joint 2 is frozen). In this case, eqn 6 corresponds to M(1,1)*q1ddot = -g(1). With joint-2 frozen, the motion of joint 1 depends only on the gravity load g(1).

For acrobot stability, we need (q1dot, q2dot) = (0,0) to have the system at rest. This corresponds to angular momentum of h=0. But achieving h=0 can only be accomplished by using gravity. Consequently, the control strategy must first use the joint-2 actuator to position the robot such that a desirable gravity term g(1) is achieved. This gravity term must be used to control h and bring h to 0. Note that h=0 does not imply that (q1dot, q2dot) = (0,0), but this is a necessary pre-condition for achieving (q1dot, q2dot) = (0,0).

Yet another simplification is that one can use computed torque to achieve direct control of $\ddot{q}_2$. Since the inertia matrix is always invertible, the general dynamics equation can be expressed as:

(7) $\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\left[-\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}\right]$

where $\ddot{\boldsymbol{q}}$ is a column vector of joint accelerations, the vector $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is defined based on Eqn 2, the G vector (from Eqn 3) is $\boldsymbol{g}(\boldsymbol{q}) = [g_1(\boldsymbol{q}), g_2(\boldsymbol{q})]^T$ and the torque vector is $\boldsymbol{\tau} = [0, \tau_2]^T$.

The second row of Eqn 7 is:

(8) $\ddot{q}_2 = [\boldsymbol{M}_{2,1}^{-1}, \boldsymbol{M}_{2,2}^{-1}](-\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q})) + \boldsymbol{M}_{2,2}^{-1}\tau_2$
$= b + \boldsymbol{M}_{2,2}^{-1}\tau_2$

In Eqn (8), the term “b” is an offset bias, with units of angular acceleration, which is computable given the system state (joint angles and joint angular velocities). The term $M^{-1}(2,2)$ is computable given only q_2 . For any desired $\ddot{q}_2$, there is a corresponding value of τ_2 that will achieve the target acceleration $\ddot{q}_2$ (subject to torque saturation limits). As a simplifying substitution, we may assume that $\ddot{q}_2$ may be treated as an equivalent control input (assuming this is achievable with joint-actuator torque constraints).

Consequently, the first row of the dynamics equation yields:

(9) $\boldsymbol{M}_{\mathbf{1,1}}\ddot{q}_1 = -c_1 - g_1 - \boldsymbol{M}_{\mathbf{1,2}}\ddot{q}_2$

At low speeds, the “**c**” term, comprised of centrifugal and Coriolis effects, may be ignored, since this contribution grows as the square of the velocities (and is zero at zero velocities).

During balance recovery, joint velocities hopefully remain low. However, joint accelerations may be high. With respect to Eqn 9, if the joint-2 accelerations are large, such that $|\boldsymbol{M}_{1,2}\ddot{q}_2| \gg |c_1 + g_1|$, then approximately:

$$\boldsymbol{M}_{1,1}\ddot{q}_1 = -\boldsymbol{M}_{1,2}\ddot{q}_2$$

That is, the acceleration of joint 1 is (negatively) proportional to the acceleration of joint 2. Consider an impulse (a torque exerted for a time duration "T") produced by the actuator at joint 2 imposed by enforcing a joint-2 acceleration of $\ddot{q}_{2,cmd}$ for a duration of T followed by an equal and opposite impulse (corresponding to $-\ddot{q}_{2,cmd}$ for the same duration T). This will result in a net joint-2 displacement of $\Delta q_2 = 2\mathrm{T}\ddot{q}_{2,cmd}$. From the approximate acceleration relationship between joint 1 and joint 2, there will be a corresponding net joint-1 displacement of: $\Delta q_1 = -(\boldsymbol{M}_{1,2}/\boldsymbol{M}_{1,1})\Delta q_2$. That is, as one controls q_2 to move to a new angle, q_1 moves in the opposite direction proportionately, with constant of proportionality equal to the inertia ration -M(1,2)/M(1,1).

Figure 6 shows some open-loop dynamics simulated for the example acrobot. This plot was generated with a torque-2 excitation near saturation torque, reversing every 100ms. Proportional feedback was added to the trq2 term to keep joint 2 from drifting too far away from an average of 0 radians. Angle q1 drifts away uncontrolled, due to initial conditions (non-zero angular velocity) and due to the influence of gravity term g(1).

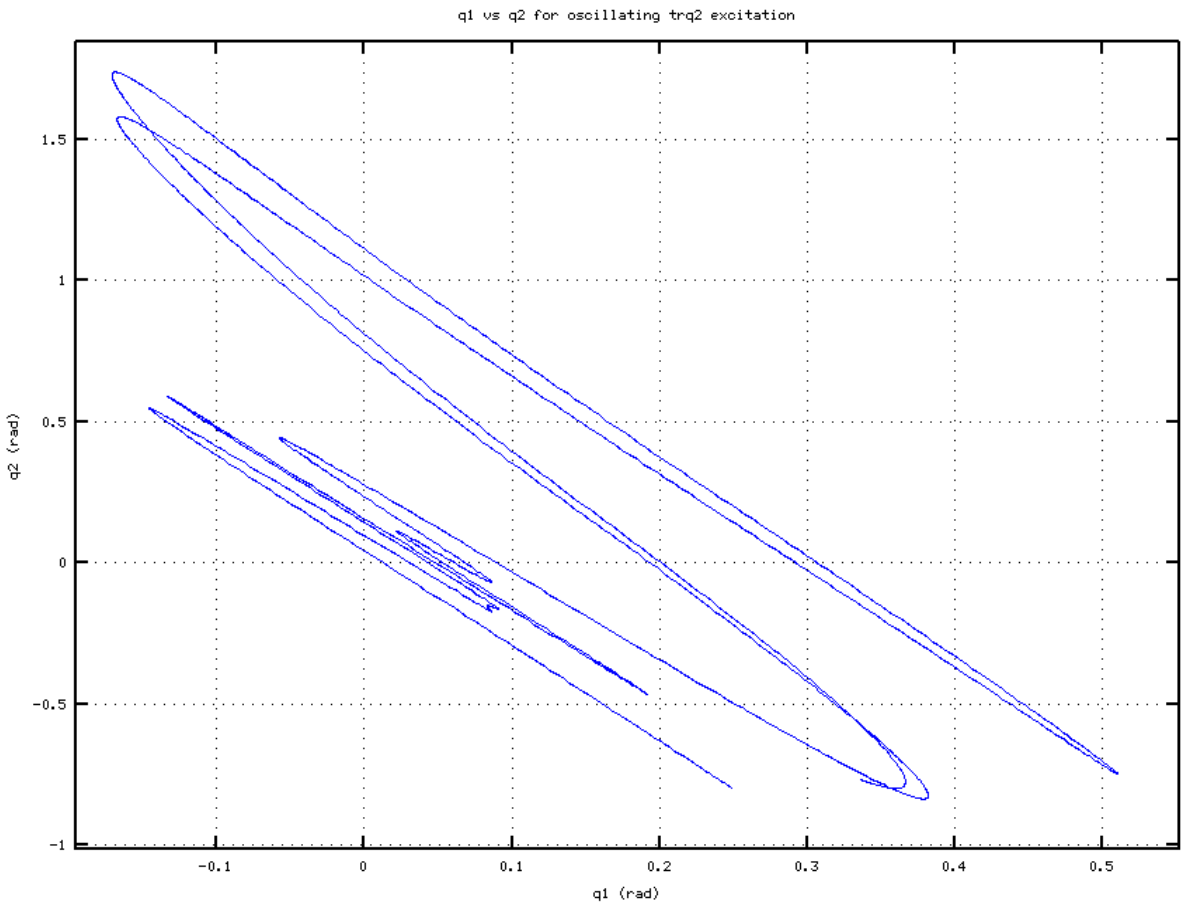

Illustration 6: q1 vs q2 for oscillatory trq2 input of acrobot

Through 12 velocity reversals, the relationship between q_1 and q_2 follows approximately $\Delta q_1 = m \Delta q_2$, with a slope "m" of approximately m = -0.28. The expected ratio, M(1,2)/M(1,1), varies with angle q_2. The maximum ratio is 0.31 at q_2=0. At larger values of q_2, the ratio decreases. At q_2 near 1.5 rad, the ratio is about 0.2. Very approximately, there is an inertial coupling between q_1 and q_2 such fast moves of q_2 result in proportional reactions of q_1 with a constant of proportionality of approximately -0.3.

Fig 7 shows a result of alternating impulses with the dynamics including the centrifugal and Coriolis terms of the acrobot. Even with this additional nonlinear term, rapid displacements of q_2 induce corresponding proportional rapid displacements of q_1.

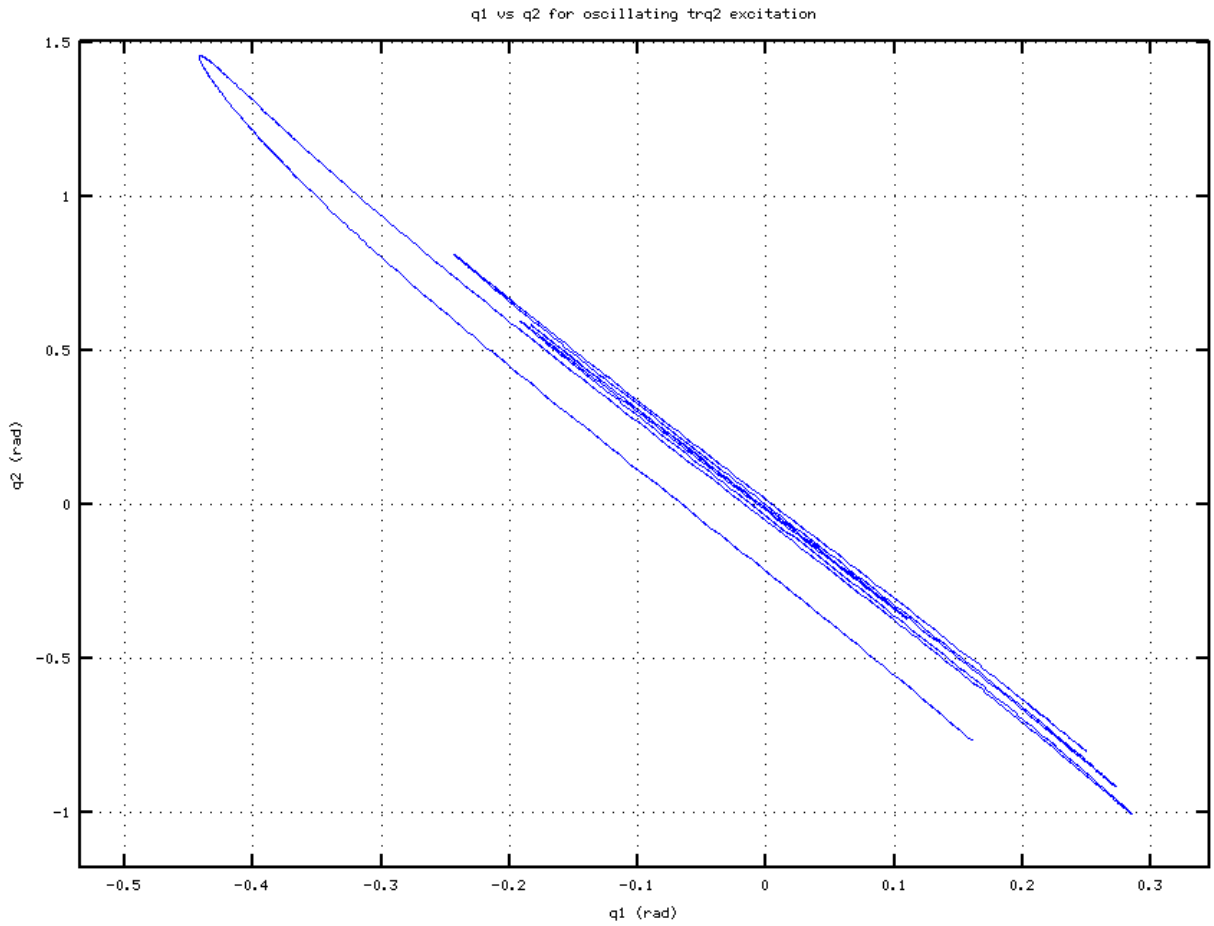

Illustration 7: acrobot q1 vs q2 for fast, oscillatory, open-loop excitation of torque 2. Dynamics includes centrifugal and Coriolis terms.

It will be useful to refer to a new quantity, a momentum "displacement" D, defined as:

(10) D = M(1,1)*q1 + M(1,2)*q2

The quantity "D" is related to the angular momentum "h" through a derivative:

(11)
$$dD/dt = \boldsymbol{M}_{1,1}\dot{q}_1 + \boldsymbol{M}_{1,2}\dot{q}_2 - 2c\sin(q_2)\dot{q}_2 q_1 - c\sin(q_2)\dot{q}_2 q_2$$
$$= h - c\sin(q_2)(2q_1 + q_2)\dot{q}_2$$

where "h" is the angular momentum and "c" is the constant defined earlier: c = m_2*L_1*L_{c2} (not to be confused with the vector **c**, corresponding to centrifugal and Coriolis torques).

For D=0, it is (at least approximately) possible to perform a rapid (bang-bang torque-2 excitation) move to achieve the position (q1,q2) = (0,0). No other value of D will allow for reaching the origin through action of trq2 alone. Additionally, if h = 0, then it is possible to bring the velocities $(\dot{q}_1, \dot{q}_2)$ to (0,0). No other value of h will allow for reaching $(\dot{q}_1, \dot{q}_2)=(0,0)$ through action of torque-2 alone. Thus, a subgoal for balancing is to achieve D=0 and h=0.

A Switching Control Strategy for Balance Recovery: Putting these pieces together, we can construct a heuristic strategy for balance-recovery control. This strategy depends on the fact that slope of q1 vs q2 for constant g1 (e.g. Fig 4 or 5) is different from the slope of q1 vs q2 for constant D (e.g. Fig 6 or 7). Figure 8 illustrates an example balance recovery trajectory.

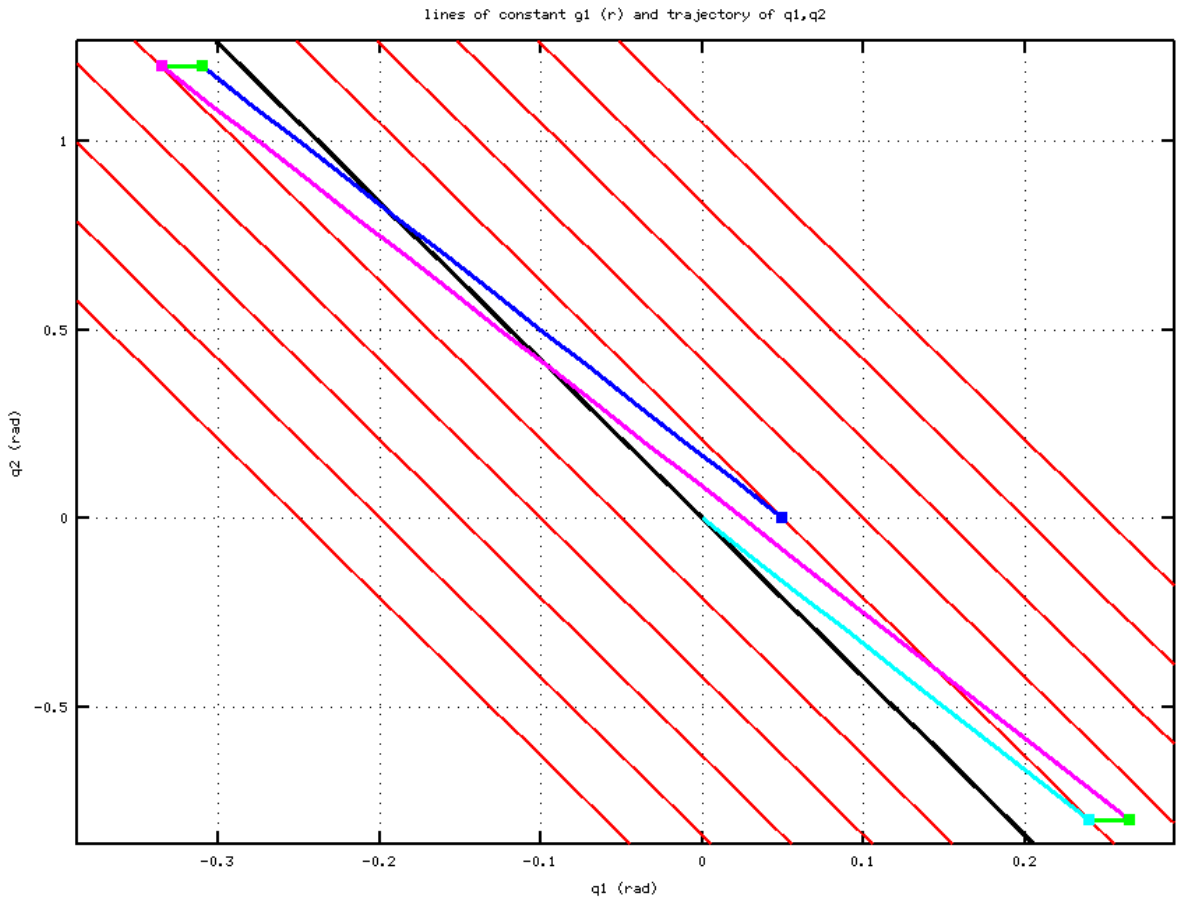

Illustration 8: Illustration of balance recovery. Trajectory starts at blue square, follows blue line, then green, then magenta, then green, then cyan to origin.

In Fig 8, the red lines correspond to (q1,q2) pairs of constant gravity torque about joint 1. The black line corresponds to zero gravity torque, e.g. corresponding to the poses of Fig 2.

Figure 8 only shows 2 of the four states variables: (q_1, q_2) , but not $(\dot{q}_1, \dot{q}_2)$. The actual trajectory of the robot in state space occurs in 4-D. In this illustrative example, the acrobot starts from the state (q_1, q_2) = (0.05,0.0), indicated by the blue square. The initial joint velocities are at rest: $(\dot{q}_1, \dot{q}_2)$ = (0,0). This will be labelled state "0". At state 0 the robot is leaning to its left (positive q1) and is at risk of falling over towards its left (increasing q1). State 0 lies on the red constant-gravity line corresponding to 1N-m of torque. At state 0, the initial angular momentum, h(0), is zero, since the joint velocities are both zero. However, the displacement, D, is positive: $D(0)= M(1,1)*q_1(0) >0$.

The only way to correct the displacement error is to reposition to a pose in which the torque about joint 1 due to gravity is negative. This can be done with a fast move of q2. This is accomplished by exerting a positive impulse via torque-2, then exerting an equal and opposite impulse. These impulses create an approximately triangular velocity profile of joint 2, starting from rest and ending at rest. If this is accomplished quickly (such that the impulse due to g(1) is negligible compared to the impulses of trq2), then the angular momentum, h, will remain at zero. With equal and opposite impulses from the actuator, the joint velocities starting from (0,0) will end up at (0,0). However, both q1 and q2 will be at new angles. These angles will lie along constant D.

For the example, the impulses result in motion along the blue line in Fig 8, ending up at the green square at which (q_1, q_2) ~= (-0.32,1.25) at rest: $(\dot{q}_1, \dot{q}_2)$ = (0,0). This will be called state "1". Although the robot ends up in a dramatically different pose at the end of this bang-bang move, the values of h and D are unchanged: h(1) = h(0) and D(1)=D(0). This is characteristic of all fast q2 moves: h and D will both be unchanged. Only gravity torques can alter h and D.

Although h and D are unchanged by moving to state 1, this new pose corresponds to a different gravity load. State 1 lies on the red line of constant gravity torque corresponding to -1N-m. Thus, the first fast move successfully achieves the subgoal of inducing negative torque due to gravity. In this pose, the robot will tend to fall to its right (decreasing values of q1).

The first move may be surprising. The robot starts off leaning to its left, and the first move of the waist joint is to have the torso lean *into* the fall. If the robot leans in the opposite direction of its fall, matters will only get worse. By leaning into the fall, the gravity torque can be negated.

It should be noted that it is only possible to alter the gravity load through a move of q2 because the slope of lines of constant D, $D = M(1,1)*q_1 + M(1,2)*q_2$, is different from the slope of lines of constant g, where $g = K_{g1}*q_1 + K_{g2}*q_2$. To the extent that this difference in slopes can be exaggerated, the balancing strategy will become easier. This may explain why tight-rope walkers carry a long pole (increasing Ic2 while hardly increasing m2*Lc2).

From state 1 (the green square), the controller freezes the value of q_2 and allows the robot to begin to fall to its right, due to the influence of gravity. During this period, q_1 decreases while q_2 is constant, following the green line of Fig 8 from state 1 (green square) to state 2 (magenta square). State 2 corresponds approximately to (q_1, q_2) = (-0.34,1.25), $(\dot{q}_1, \dot{q}_2) = (\dot{q}_1(2), 0)$ with $\dot{q}_1(2) < 0$. This results in a change in both h and D: $h(2) = M(1,1)*\dot{q}_1(2)$ and $D(2) = H(1,1)*q_1(2) - H(1,2)*q_2(2)$. The angular momentum at state 2 is negative, and the value of the displacement has also decreased: D(2)<D(1). However, the displacement at state 2 is still (intentionally) positive: D(2)>0.

From state 2, another fast move is invoked, this time with a negative displacement of q2. This move is along the magenta line, ending up at state 3, corresponding to the green marker at approximately (q_1, q_2) = (0.27,-0.8) and with velocities $(\dot{q}_1(3), \dot{q}_2(3)) = (\dot{q}_1(2), 0)$. Again, if the joint-2 torques are large compared to the gravity torque (where the gravity torque starts at approximately -1N-m), then the fast move will result in approximately unchanged h and D: h(3)=h(2) and D(3)=D(2). However, the new configuration at state 3 now has a positive torque due to gravity g > 0. Importantly, h and D have opposite signs.

At state 3, joint 2 is frozen at q_2= -0.8. At this pose, the torque due to gravity is positive. However, the system's angular momentum, h, is still negative, and thus the robot will continue to rotate towards more negative values of q_1. Since the gravity torque is positive, this velocity is being brought to rest as the robot moves from state 3 to state 4, along the green line from the green square

marker (state 3) to the cyan square marker (state 4). During this move, both h and D are decreasing. By choosing the target state 3 carefully, the motion from state 3 to state 4 can be designed to result in h(4)=0, since $(\dot{q}_1, \dot{q}_2)$ =(0,0), and D(4)=0. This is a precondition for recovering balance. However, the robot cannot remain in state 4, since gravity will make it start to fall to the left (increasing q_1).

The final step is a fast move of q_2 from $q_2(4)$=-0.8 to $q_2(5)$=0. This again will be done with equal and opposite impulses, thus starting from $(\dot{q}_1(4), \dot{q}_2(4))=(0,0)$ and ending with $(\dot{q}_1(5), \dot{q}_2(5))=(0,0)$, preserving h=0 and ending up at rest. Additionally, since D(4)=0, then D(5)=0. By moving to $q_2(5)$=0, it follows that $q_1(5)$=0 (consistent with D(5)=0). Thus, the robot ends up at (q_1, q_2) = (0,0) (with D=0) and $(\dot{q}_1, \dot{q}_2)$ =(0,0) (with h=0), and this pose corresponds to 0 gravity torque (g=0), and thus this pose is stable.

To recap, the control sequence is as follows. Starting from state 1 at D>0, h=0, a fast q_2 move brings the system to state 2 at D>0, h=0, but with negative gravity torque. Under frozen q_2, the robot is allowed to begin to fall towards state 3, for which h<0, D>0. A fast q_2 move in the opposite direction then brings the system to state 4, still with h<0 and D>0, but with a positive torque due to gravity. Again, q_2 is frozen, allowing gravity to slow down the rotation of q_1, bringing the system to rest (instantaneously) at state 5, for which D=0 and h=0. However, this pose does not have g=0. A final fast move of q_2 brings the system to the final state for which D=0, h=0 and g=0, which is stable.

The above strategy has the correct logic (lean left, then lean right, then straighten up). However, it depends on switching at precise states. A more robust feedback controller is desired. Further, this controller should be achievable through an available interface: commands to a joint PD position controller. These extensions are presented next.

State-space dynamics in transformed coordinates: Although the switching-control strategy helps to clarify the theory of balance recovery, a more robust controller is needed. To develop the controller, a convenient state-space representation of system dynamics is needed.

Recognizing the value of the displacement and momentum, D and h, these two quantities are chosen to be two of the state variables. The importance of controlling pose to achieve a desired gravity influence is also clear, and thus the torque about joint 1 due to gravity is chosen as a state variable. The scaler symbol “g” will be used. In the general dynamics formula, the vector **g** was defined (Eqn 3). The state variable “g” will be defined as -**g**(1). (The minus sign is for convenience, so that positive “g” results in positively increasing h). This value is the torque about joint 1 due to gravity, in units of N-m.

For the switching controller analysis, it was assumed as an approximation that q_2 could be moved quickly, resulting in almost instantaneous changes in configuration and new values of g. More realistically, the present analysis will assume that g cannot be changed instantaneously, and thus a state variable “ $\dot{g}$ ” will be introduced, where $\dot{g}=\frac{dg}{dt}$. The proposed state vector, **x**, is thus:

$$\text{(12)} \quad \boldsymbol{x}=\begin{bmatrix} D \\ h \\ g \\ \dot{g} \end{bmatrix}$$

The time derivative of state can then be expressed as:

(13) $$\dot{\boldsymbol{x}}=\frac{d}{dt}\begin{bmatrix}D\\h\\g\\\dot{g}\end{bmatrix}=\begin{bmatrix}0100\\0010\\0001\\0000\end{bmatrix}\begin{bmatrix}D\\h\\g\\\dot{g}\end{bmatrix}+\begin{bmatrix}d_1(q_2,\dot{q}_1,\dot{q}_2)\\0\\0\\K_{g1}\ddot{q}_1+K_{g2}\ddot{q}_2\end{bmatrix}+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\text{u}$$

where the last vector is Eqn 13 is a nonlinear function. From Eqn 11, $d_1(q_2,\dot{q}_1,\dot{q}_2)=-c\sin(q_2)(2q_1+q_2)\dot{q}_2$. The term $K_{g1}\ddot{q}_1+K_{g2}\ddot{q}_2$ is a computation of $\ddot{g}$, which according to Eqn (4) can be expressed as:

(14) $$\ddot{g}=\begin{bmatrix}K_{g1}K_{g2}\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\\ddot{q}_2\end{bmatrix}$$

Equation 14 can be further interpreted in terms of assuming direct control of $\ddot{q}_2$, i.e. $\ddot{q}_2=\ddot{q}_{2,cmd}$. It was shown by Eqn (8) that this is possible, since for any $\ddot{q}_{2,cmd}$, there is a corresponding torque input, τ_2 .

Substituting for $\ddot{q}_1$ in Eqn 14 per Eqn (9) yields:

(15) $$\begin{aligned}\ddot{g}&=(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})(-c_1+g-\boldsymbol{M}_{\mathbf{1,2}}\ddot{q}_2)+K_{g2}\ddot{q}_2\\&=-(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})c_1+(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})g+(K_{g2}-K_{g1}\boldsymbol{M}_{\mathbf{1,2}}/\boldsymbol{M}_{\mathbf{1,1}})\ddot{q}_2\\&=-(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})c_1+a_{4,3}g+b_4\ddot{q}_2\end{aligned}$$

Recognizing that "g" in Eqn 15 is a state variable and that $\ddot{q}_2$ will be treated as an input, and using the definitions of a_{34} and b_4 from Eqn 15, we can re-write Eqn 13 as:

(16) $$\dot{\boldsymbol{x}}=\frac{d}{dt}\begin{bmatrix}D\\h\\g\\\dot{g}\end{bmatrix}=\begin{bmatrix}0100\\0010\\0001\\00a_{4,3}0\end{bmatrix}\begin{bmatrix}D\\h\\g\\\dot{g}\end{bmatrix}+\begin{bmatrix}d_1(q_2,\dot{q}_1,\dot{q}_2)\\0\\0\\d_4(q_2,\dot{q}_1,\dot{q}_2)\end{bmatrix}+\begin{bmatrix}0\\0\\0\\b_4\end{bmatrix}\ddot{q}_{2,cmd}$$

where $d_4=-(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})c_1=(K_{g1}/\boldsymbol{M}_{\mathbf{1,1}})m_2L_1L_{c2}\sin(q_2)(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)$ originates from the Coriolis/centrifugal virtual torque on link 1.

Since there is a computable mapping between the actual control input, τ_2 , and the assumed control input, $\ddot{q}_{2,cmd}$, the control policy can be expressed in terms of $\ddot{q}_{2,cmd}$, and it can be implemented in terms of τ_2 (or, alternatively, via a PD controller, as shown subsequently).

Having shown that it is feasible to define and use an equivalent control input u= $\ddot{q}_{2,cmd}$, we can re-write the dynamics of our system in our chosen state space, Eqn 16, as:

(17) $$\dot{\boldsymbol{x}}=\boldsymbol{A}\,\boldsymbol{x}+\boldsymbol{d}+\boldsymbol{b}\,u$$

The system, in this representation, is nearly linear, except for the nonlinear term **d**(**x**), defined in Eqn 16.

Given the re-expressed dynamics of Eqn 17, we can derive a control law, u(**x**), which is presented next.

A sliding-mode balance-recovery controller: For the state dynamics of Eqn 17, it is desired to bring the state **x** to **0**. At this state, the system will be upright, at rest and in an unstable equlibrium. The controller must achieve this state and maintain this state to keep the robot upright. An approach to such control is sliding-mode control. In this technique, one chooses a constraint equation,

(18) $\sigma(\boldsymbol{x})=0$

such that: 1) it is possible to drive the system to satisfy this constraint (and to maintain satisfaction of this constraint), and 2) when satisfying this constraint, the ensuing dynamics are desirable.

A common approach is to define: $\sigma(\boldsymbol{x})\equiv[a_1 a_2 a_3 a_4]\boldsymbol{x}$

The values of a may be chosen, e.g., to be:

(19) $\sigma(\boldsymbol{x})\equiv[\lambda^3, 3\lambda^2, 3\lambda, 1]\boldsymbol{x}$

These values were chosen based on an approximation, in which the "**d**" term is ignored in Eqn 16. Because this is an approximation, convergence proofs for sliding-mode control do not hold, and thus it is uncertain if this controller will be successful. The above choice for constraint surface results in:

(20) $\sigma(\boldsymbol{x})=\lambda^3 D+3\lambda^2 h+3\lambda g+\dot{g}$

If the **d** vector of Eqn 16 can be ignored, Eqn 16 becomes linear, and the terms of **x** can be expressed as derivatives of D. In addition to ignoring the nonlinear d vector, the derivation also assumes use of the approximate (linearized) gravity model as a function of joint angles. For large angles, this will be invaled. However, large angles will also inevitably result in the robot falling, so the assumption may be valid over the range of interest.

For the approximate linear system, the derivatives can be expressed in the Laplace domain with operator "s". Further, assuming the sliding-mode controller is successful in reaching the sliding surface, then the constraint will be enforced that $\sigma(\boldsymbol{x})=0$. Combining these assumptions results in:

(21) $0=\lambda^3 D+3\lambda^2 s D+3\lambda s^2 D+s^3 D$

Which factors to:

(22) $0=(s+\lambda)^3 D$

The above linear system corresponds to three coincident eigenvalues (poles) all equal to $-\lambda$. To the extent that the linear approximately holds, any positive choice of lambda will result in the system being stable, converging to **x** = **0** with a time constant of $1/\lambda$.

To enforce approach to the constraint surface $\sigma(\boldsymbol{x})=0$, a sliding-mode control law can be used. In general, if $\sigma(\boldsymbol{x})>0$, the controller should enforce that $\dot{\sigma}(\boldsymbol{x})<0$, and if $\sigma(\boldsymbol{x})<0$, the controller should enforce $\dot{\sigma}(\boldsymbol{x})>0$. Equivalently, the controller should enforce that $sgn(\dot{\sigma}(\boldsymbol{x}))=-sgn(\sigma(\boldsymbol{x}))$, where "sgn()" is the "signum" (sign) operator returning +1, -1 or 0,

depending on the sign of the argument.

To enforce this condition, the controller has only the value of "u" available. The derivative $\dot{\sigma}(\boldsymbol{x})$ is obtained by differentiating Eqn 20 and using relationships from Eqn 16:

(23) $\dot{\sigma}(\boldsymbol{x}) = \lambda^3(h+d_1) + 3\lambda^2 g + 3\lambda \dot{g} + a_{4,3} g + d_4 + b_4 \ddot{q}_{2,cmd}$

From Eqn 23, one can propose a controller of the form:

(24) $u = -\lambda^3(h+d_1)/b_4 - (a_{4,3} + 3\lambda^2) g/b_4 - 3\lambda \dot{g}/b_4 - d_4/b_4 - \hat{u}\, sgn(\sigma)$
$= u_{nom} - \hat{u}\, sgn(\sigma)$

The contribution u_{nom} cancels out most of the terms of $\dot{\sigma}(\boldsymbol{x})$, such that if *only* this term were used for control, one would expect a close approximation of achieving $\dot{\sigma}(\boldsymbol{x}) = 0$.

Given the approximate cancelling of terms via u_{nom}, the proposed controller of Eqn 24 (ideally) results in:

(25) $\dot{\sigma}(\boldsymbol{x}) = -\hat{u}\, sgn(\sigma)$

For any positive choice of the parameter $\hat{u}$, Eqn 25 achieves the desired effect of $sgn(\dot{\sigma}(\boldsymbol{x})) = -sgn(\sigma(\boldsymbol{x}))$, and thus this controller would enforce convergence to the constraint $\sigma(\boldsymbol{x}) = 0$. Note that Eqn 24 *does* include the nonlinear term "**d**", and thus proof of convergence to $\sigma(\boldsymbol{x}) = 0$ is valid. (The uncertainty due to the approximation of ignoring **d** is the behavior of the system on the constraint surface, once $\sigma(\boldsymbol{x}) = 0$ has been achieved).

A problem with switching controllers of the type proposed in Eqn 24 is that the harsh switching term typically results in unacceptable "chatter" about the constraint surface. An approximation to smooth out this chatter is to substitute the sgn() function with the sat() function. The sat() function has the property that it is identical to the sgn(x) function for |x|>1, but it is linear (sat(x)=x) for |x| <1. The control law then becomes:

(26) $u_{nom} - \hat{u}\, sat(\sigma/\sigma_{lin})$

The parameter σ_{lin} is introduced as a tuning adjustment for the slope of the linear region. Collectively, tuning the proposed controller requires setting values for λ, $\hat{u}$ and σ_{lin}. Making λ larger will imply attempting to make the system converge to **x**=0 faster along the constraint surface. Larger λ is desirable, but if it is too large, the controller will fail due to torque saturation. The value of σ_{lin} is ideally small. As σ_{lin} approaches 0, the sat() function approaches the sgn() function. Introducing the sat() function is an approximation that does not conform to the rigorous proof of convergence to $\sigma(\boldsymbol{x}) = 0$. However, if σ_{lin} is too small, the controller will chatter. Finally, the value of $\hat{u}$ must be at least as large as the uncertainty in Eqn 23. If this term dominates the uncertainty, then convergence to $\sigma(\boldsymbol{x}) = 0$ will be achieved. It may be desirable to make $\hat{u}$ as large as possible, since convergence to $\sigma(\boldsymbol{x}) = 0$ must be achieved rapidly, before the robot exceeds practical joint limits.

Simulation of Sliding-Mode Balance Control: The acrobot system was simulated using the control law of Eqn 26, then converting the resulting u= $\ddot{q}_{2,cmd}$ into an equivalent joint torque via Eqn (8). Results are shown below, for an initial condition of (q_1, q_2) = (0.05,0.0) and $(\dot{q}_1, \dot{q}_2)$ = (0,0). Fig 9 shows the 4 state variables vs time, which all converge to zero. Fig 10 shows the

joint angles (q_1, q_2) vs time. Figure 11 shows $\ddot{q}_{2,cmd}$ computed by the sliding-mode control law, and Fig 12 shows the corresponding joint-2 torque vs time. Convergence to a stable, upright pose is well behaved. Note that the response is heuristically similar to the control strategy described previously. The system begins by having link 2 rapidly lean *into* the fall (i.e., rotate in the same direction as the q1 offset). This causes a rapid opposite motion of q1, although the displacement (D) remains positive immediately after this move, while the gravity load on joint 1 reverses sign. This causes the angular momentum, h, to decrease until D and h have opposite signs. Subsequently, both D and h decrease to zero as the robot achievs a stable, upright position.

Figure 13 shows the sliding-surface constraint value. The constraint $\sigma(\boldsymbol{x})=0$ is nearly satisfied within about 0.3 seconds. This is the "reaching" phase during which the the system is driven towards the constraint surface. Subsequently, it takes approximately 3.5 sec to converge on the stable, upright position. The dynamics during this period correspond to satisfying the constraint $\sigma(\boldsymbol{x})=0$, during which the displacement approximately behaves according to Eqn 22.

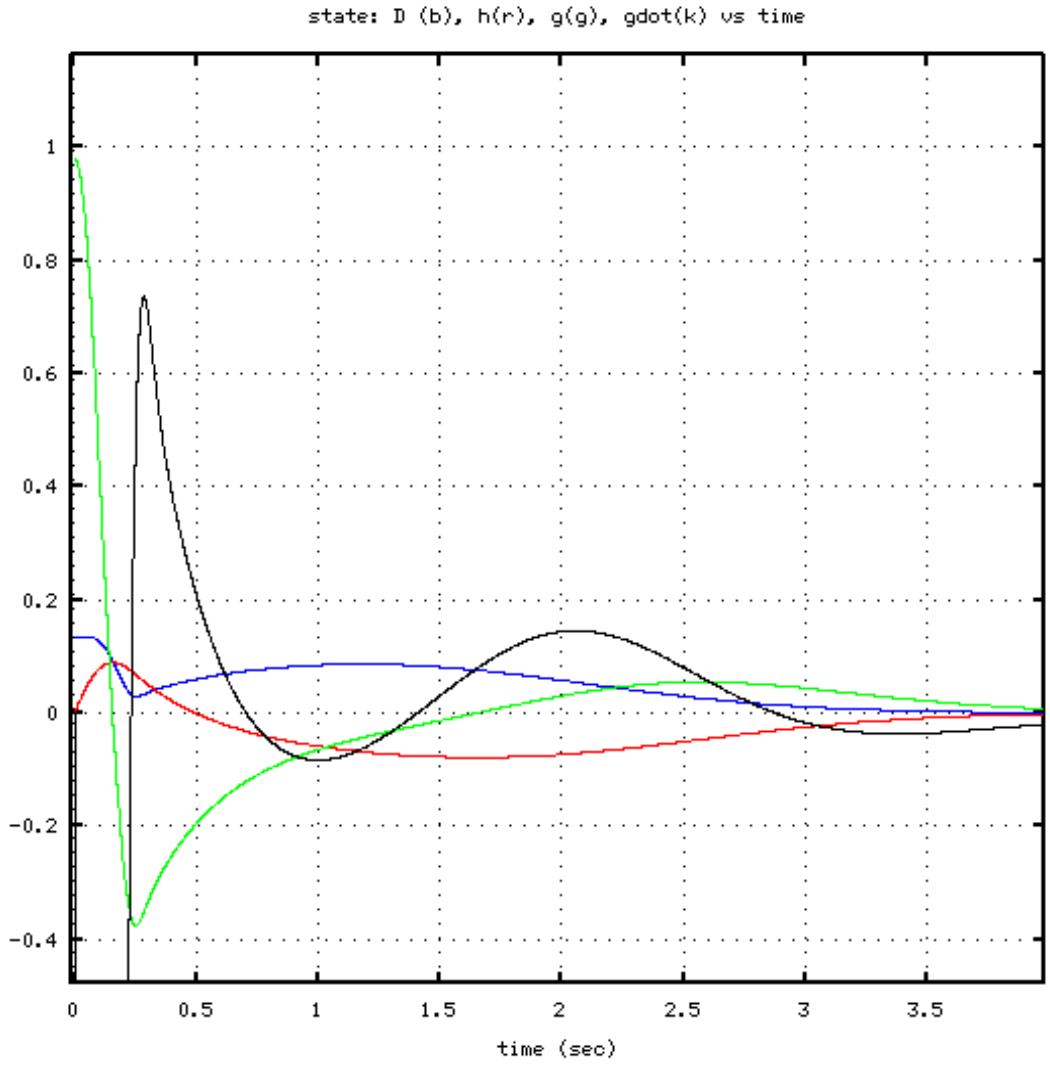

Illustration 9: Sliding-mode torque control: state variables vs time

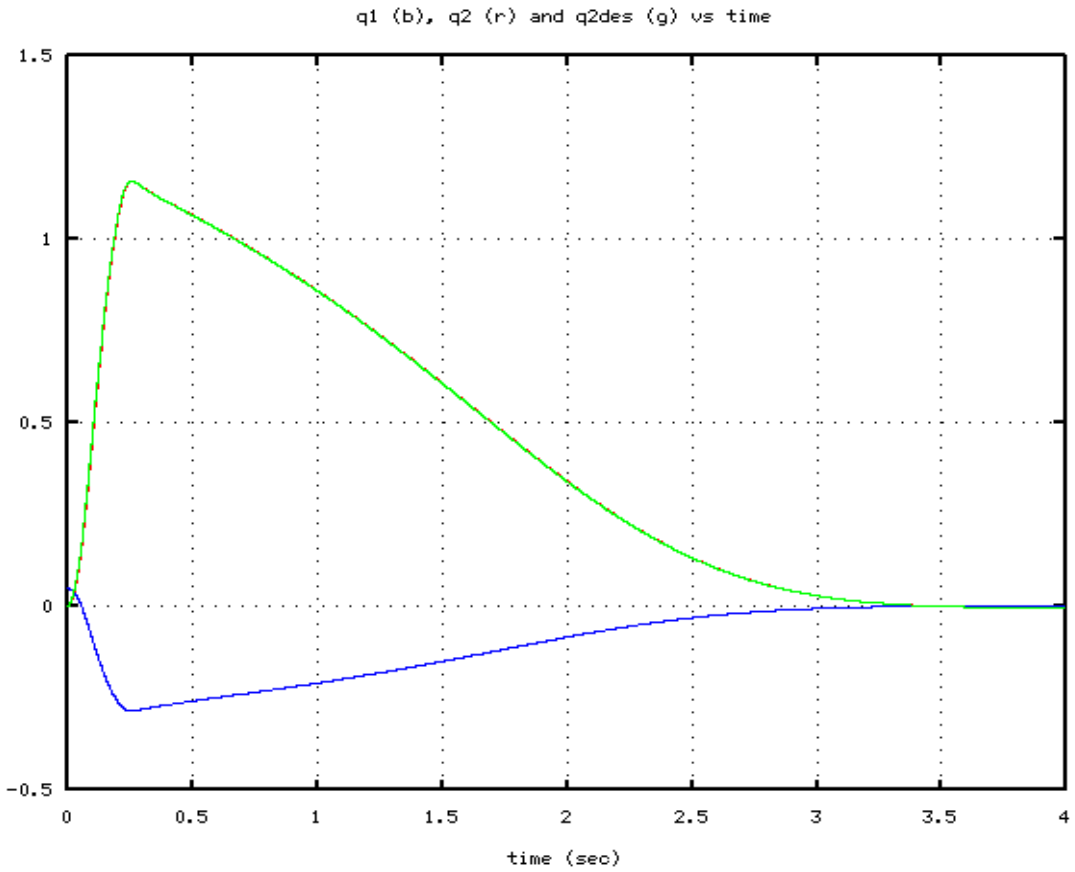

Illustration 10: Sliding-mode torque controller: joint angles vs time

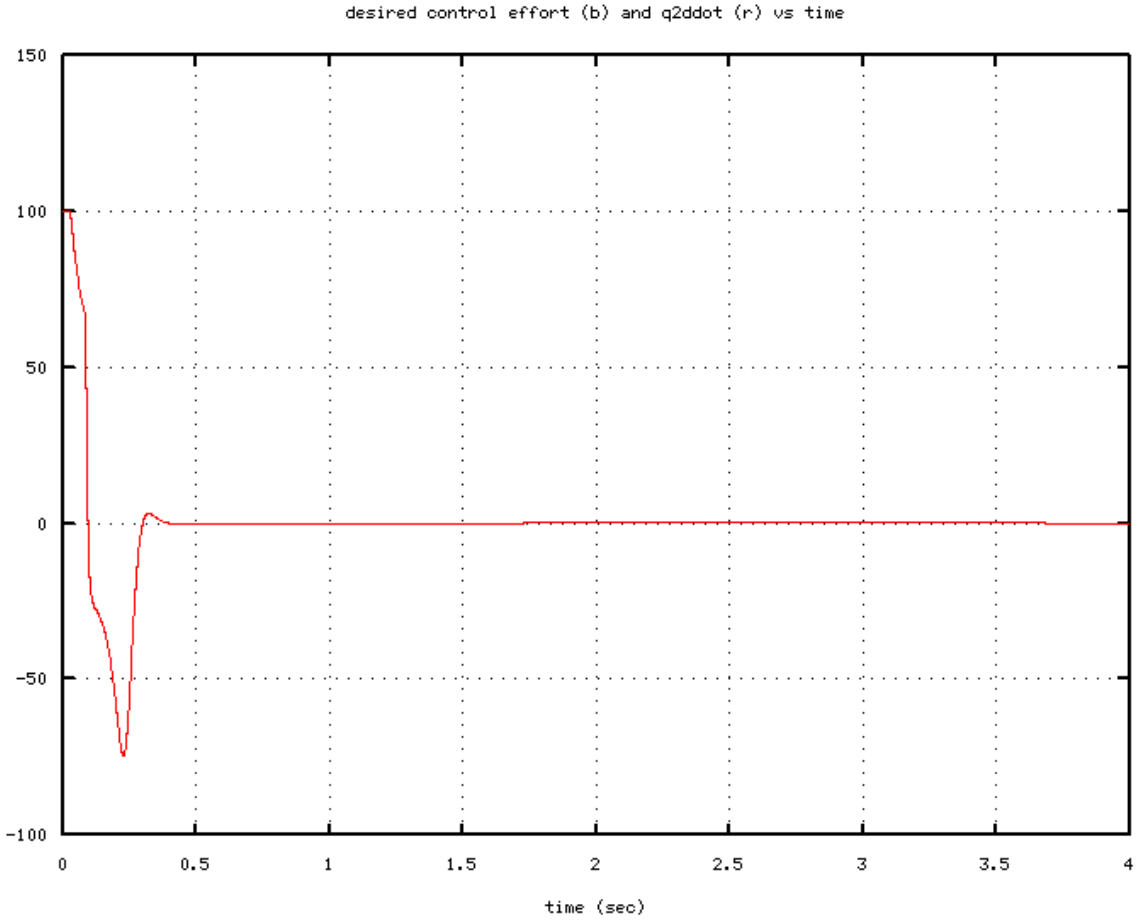

Illustration 11: Sliding-mode torque controller: desired jnt2 accel vs time

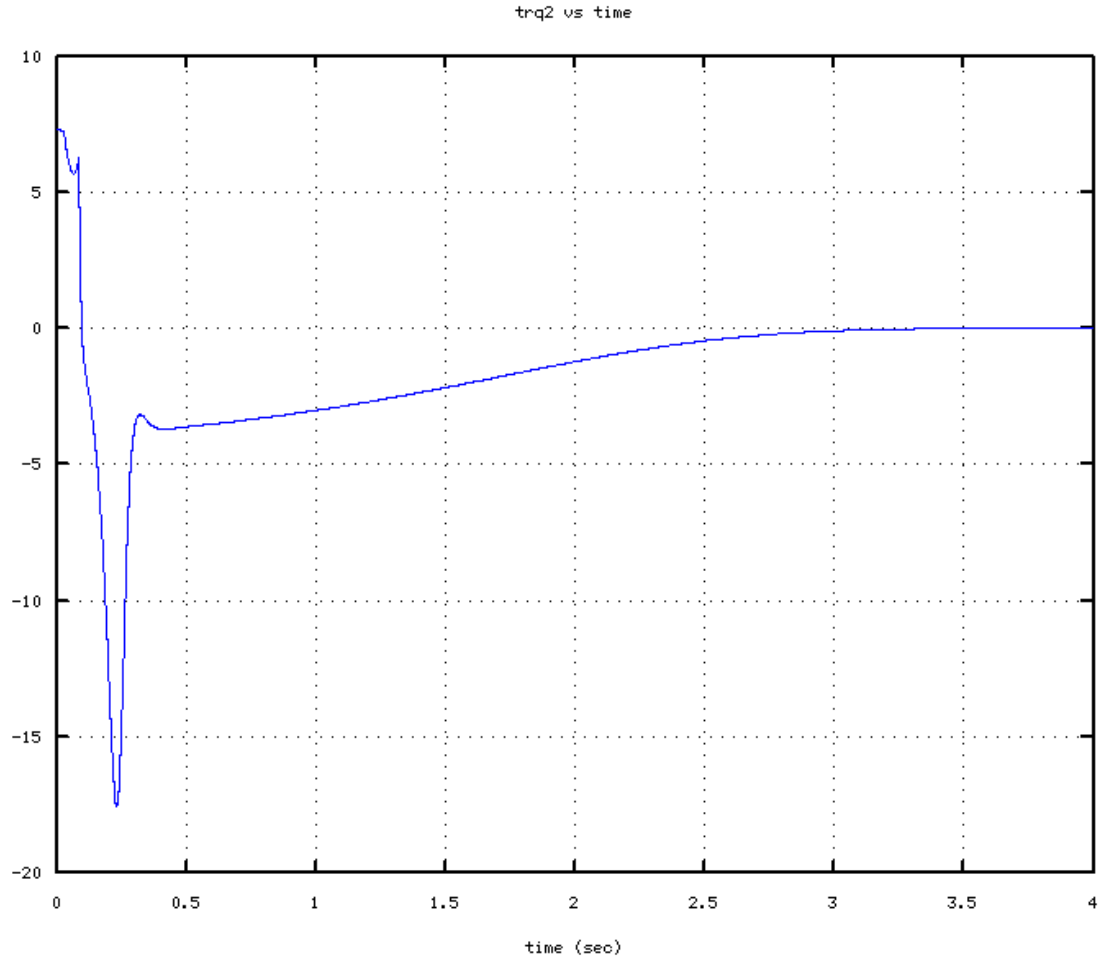

Illustration 12: Sliding-mode torque controller: joint-2 torque vs time

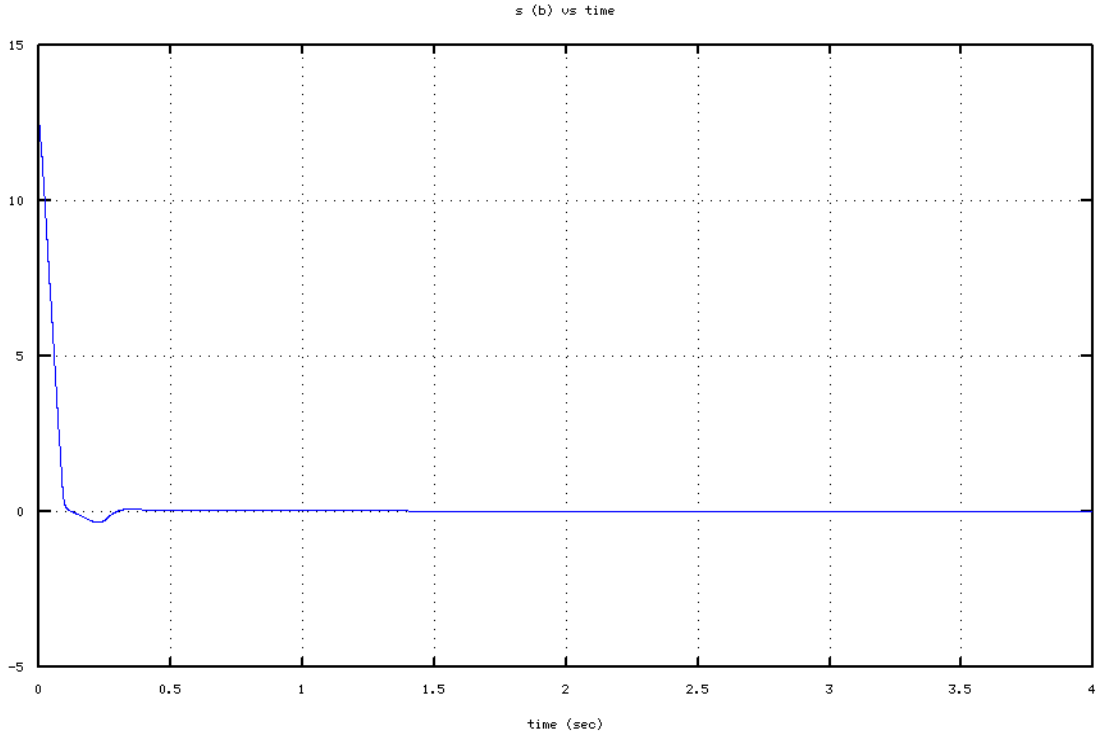

Illustration 13: Sliding-mode torque controller: constraint s(x) vs time

Balance recovery using PD joint control: The above derivation assumed use of $\ddot{q}_{2,cmd}$ as a

control input, from which the corresponding joint-2 torque was computed and imposed. However, direct joint torque control for balancing Atlas is impractical. The current interface accepts joint position and joint velocity commands, and an algorithm running on the eBox performs feedback with respect to measured joint positions and velocities to send commands to an electro-hydraulic servo valve. Direct torque control is difficult, due to large Coulomb friction. Further, direct torque control is incompatible with existing software that assumes position-command inputs. Having a stable position controller in place is also prudent for safety of the robot.

It is therefore desirable to achieve the benefits of a sliding-mode controller for balance recovery, but do so in terms of position and velocity commands. This can be done (approximately) by using the control law of Eqn 26 to derive consistent position and velocity commands. This may be done as follows.

First, invoke Eqn 26 to compute a desired control effort, u= $\ddot{q}_{2,cmd}$.

Consistent velocity and position commands can be derived as:

(27) $\dot{q}_{2,cmd} = \dot{q}_0 + \int_{t_0}^{t} u\, dt$

(28) $q_{2,cmd} = q_0 + \int_{t_0}^{t} \dot{q}_{2,cmd}\, dt$

The constants q_0 and $\dot{q}_0$ are set to the respective measured values of these quantities at the start time, t_0 .

Even though the value of "u" from the control algorithm may undergo some rapid switching, the resulting joint position and velocity commands will be smooth. In tuning the controller, saturation values of acceleration and velocity will impose constraints. These saturation values were not taken into account in the proof of convergence to the sliding surface, nor in maintaining the sliding-surface constraint. Thus, some trial-and-error tuning of the sliding-mode parameters λ , $\hat{u}$ and σ_{lin} is required.

Simulation of Sliding-Mode Balance Control via Position and Velocity Commands: Simulation of the acrobot was performed using the position and velocity command interface with a PD controller implying joint torques. The initial conditions were identical to those of the torque-based sliding-mode controller: (q_1, q_2) = (0.05,0.0) and $(\dot{q}_1, \dot{q}_2)$ = (0,0). The response is shown in Figures 14 through 16. The response is virtually indistinguishable from the torque-based sliding-mode controller. This suggests that using position and velocity commands can be effective in performing sliding-mode balance recovery control.

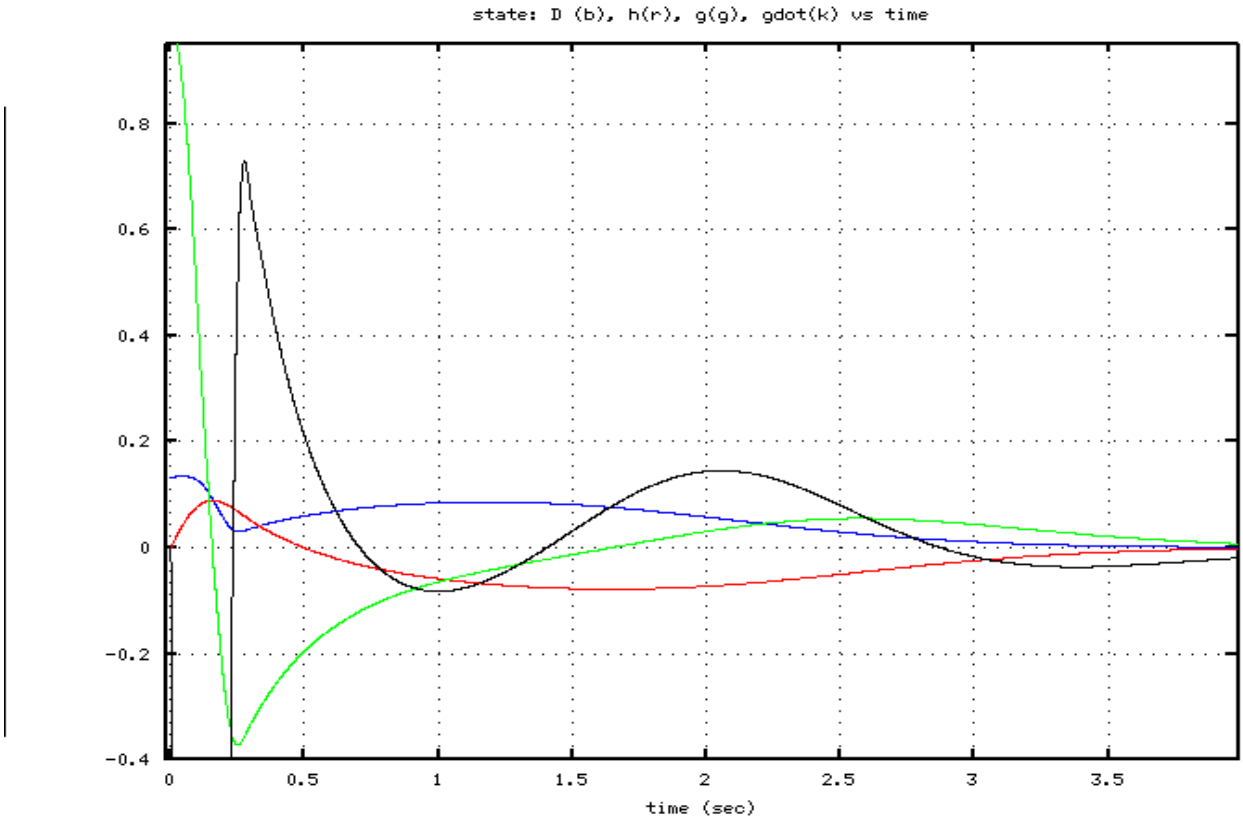

Illustration 14: Sliding mode with PD controller: states

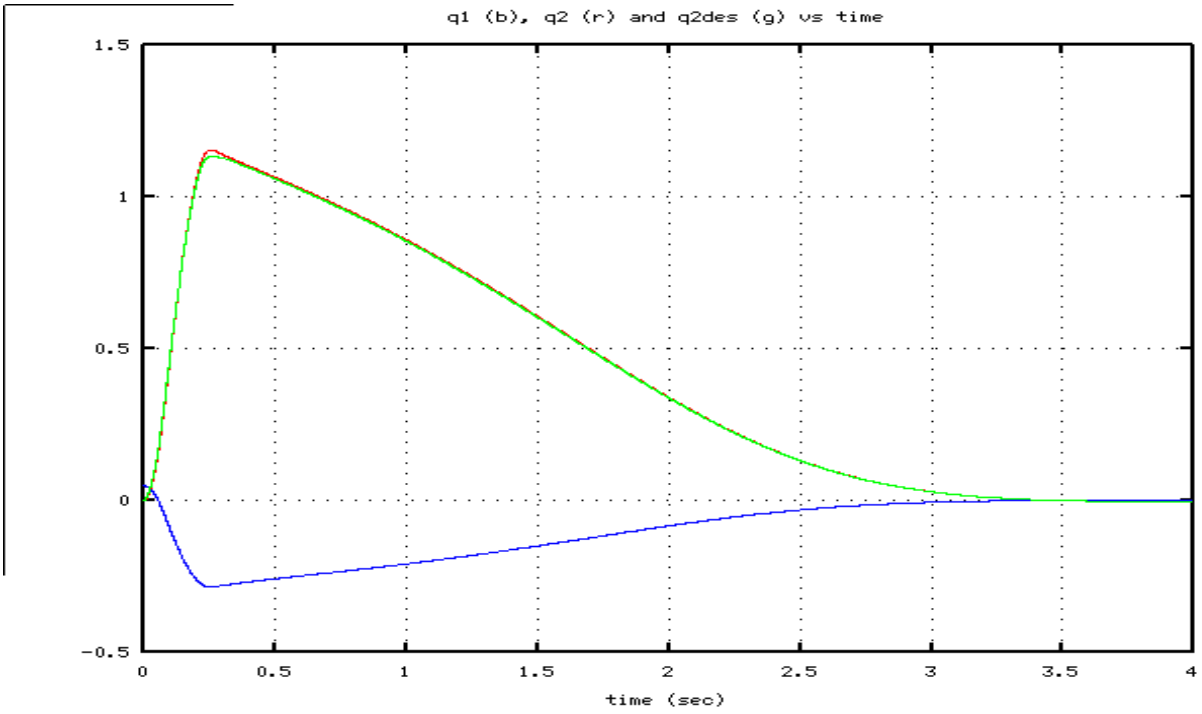

Illustration 15: Sliding mode with PD controller: q1 and q2(t)

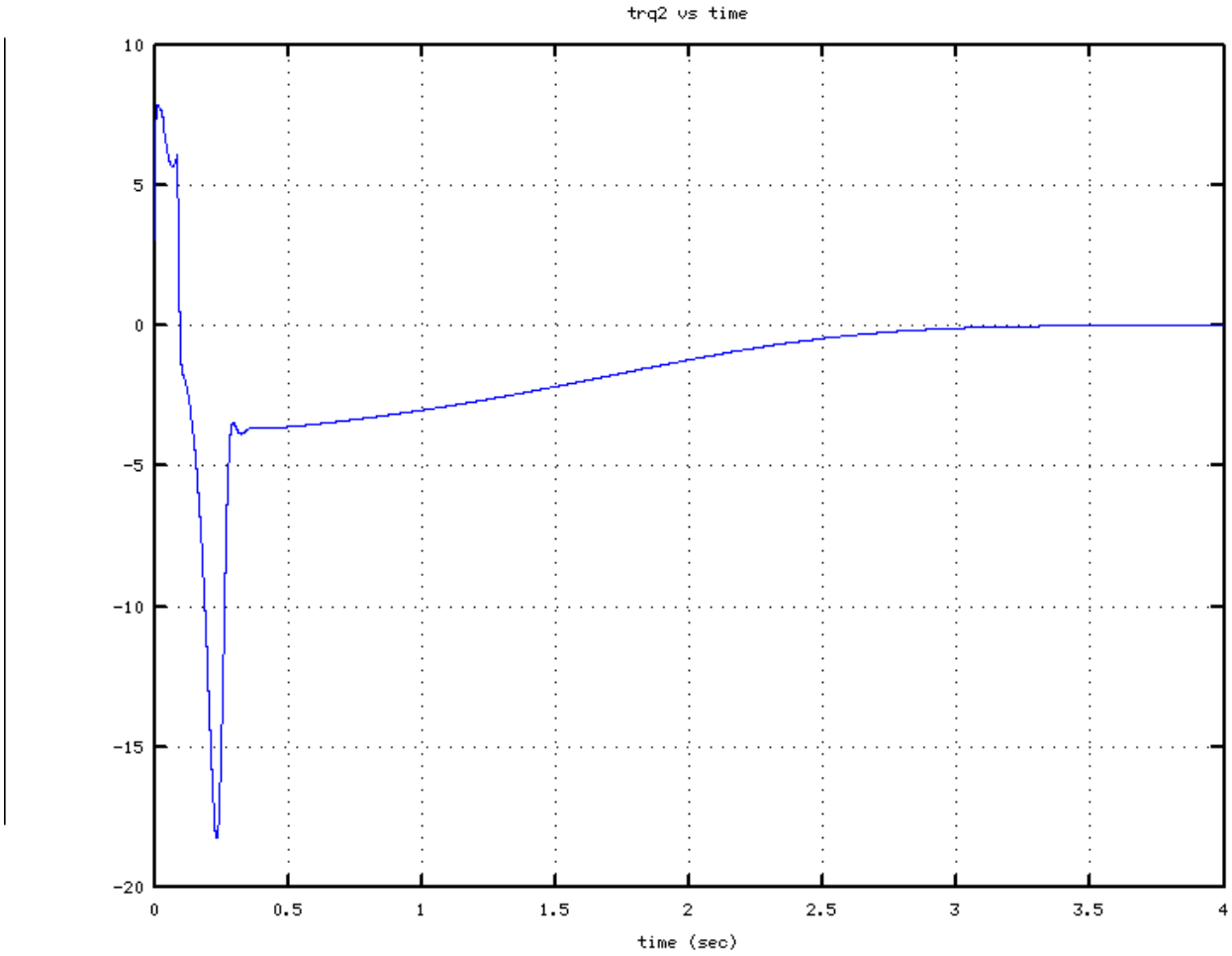

Illustration 16: Sliding mode with PD controller: torque-2 vs time

Controller code: Octave code for sliding-mode control is contained in the file "slide_mode_trq_ctl.m". This function accepts a state, (**q,qdot**) and returns the desired acceleration of joint 2, q2ddot. This function also returns values for the trq2 consistent with the desired q2ddot, as well as the values of internal computations for s (the constraint equation imbalance) and x (the 4x1 state vector of displacement, angular momentum, gravity load and gravity load derivative).

Key lines are:

```
 % construct the state vector:
x = [D;h;g1;gdot];
s = Svec*x; %want to drive this scalar function, s(x) to zero
a43 = Kg1_lin_model/H(1,1);
b4 = Kg2_lin_model-Kg1_lin_model*H(1,2)/H(1,1);
A = [0 1 0 0;
0 0 1 0;
0 0 0 1;
0 0 a43 0];
B = [0;0;0;b4];
% unom for sdot = 0:
unom = -Svec*A*x/(Svec*B); %terms from linear state-space eqns only:
% add more terms to cancel nonlinearities:
s2 = sin(q(2));
```

```
d1 = -c_core*s2*(2*q(1)+q(2))*qdot(2);
d4 = -(Kg1_lin_model/H(1,1))*c_vec(1); %c_core*s2*(2*qdot(1)*qdot(2) + qdot(2)*qdot(2));
d_vec = [d1;0;0;d4];
unom = unom - Svec*d_vec/(Svec*B);


q2ddot = unom -sat(Ks*s)*sign(Svec*B)*uhat; %this is the desired q2ddot
```

Conversion of q2ddot control inputs into position and velocity inputs is performed in the main program, “acrobot_simu_main.m”. Key lines are:

```
%use q2ddot_des to define q2des, q2dotdes:

q2dotdes = q2dotdes + q2ddot_des*dt;
q2des = q2des + q2dotdes*dt;

%compute torque due to a PD controller on link 2:

trq_ctlr = Kp*(q2des - q(2)) + Kv*(q2dotdes - qdot(2));
```

Computation of torque from q2des and q2dotdes would normally take place in the “eBox” of Atlas via the PD feedback controller.

Future Work: In next steps, this controller should be evaluated in terms of the range of initial conditions that can be tolerated. This might fit a model of $|D(x)-K_h\, h(x)| < \text{tolerance}$.

The controller should be extended to simulation of balance control in “drcsim”. To do so, it will be necessary to extend the gravity model of Fig 4 to a more general (linearized) gravity model, as was done in Fig 5 for the acrobot.

In addition, it will be necessary to obtain reasonable estimates of the inertia parameters M(i,j). Notably, slopes of q1 vs q2 for fast moves, equivalent to Fig 6 and 7, must be obtained for drcsim. More generally, estimates of the parameters “a” through “e” used in the acrobot model should be estimated for drcsim.

With these model parameters, the PD-based sliding-mode controller should be tested for balance control of Atlas in drcsim. This will require converting the Octave code into ROS-compatible C++ code. The range of initial conditions for which balance can be recovered should be examined.

Subject to success in balance control in drcsim, the modeling and control should be extended to implementation on the real robot.