

Atlas Characterization:
Wyatt Newman
Sept 16, 2013

These notes describe an approach to characterizing Atlas's joints for servo control, and ultimately for compliant-motion control.

1) Electrohydraulic actuator modelling:

Control of Atlas's joints is complicated by the dynamics of his electro-hydraulic actuators. Each hydraulic actuator is controlled by a current command, i_{cmd} , which causes a control spool to expose a variable amount of orifice for a fluid-flow path either from the high-pressure fluid source or to the low-pressure exhaust port. An example figure (from Moog), below, shows the flow path from the pressure source (P_s) to port “A” and from port “B” to the low-pressure “tank” (T) when the control spool is displaced to the left. If the control spool is displaced to the right, the flow paths will be reversed (high-pressure fluid will be exposed to port B, and port A will be exposed to the low-pressure tank). The magnitude of displacement of the control spool regulates the flow restrictions (orifice openings).

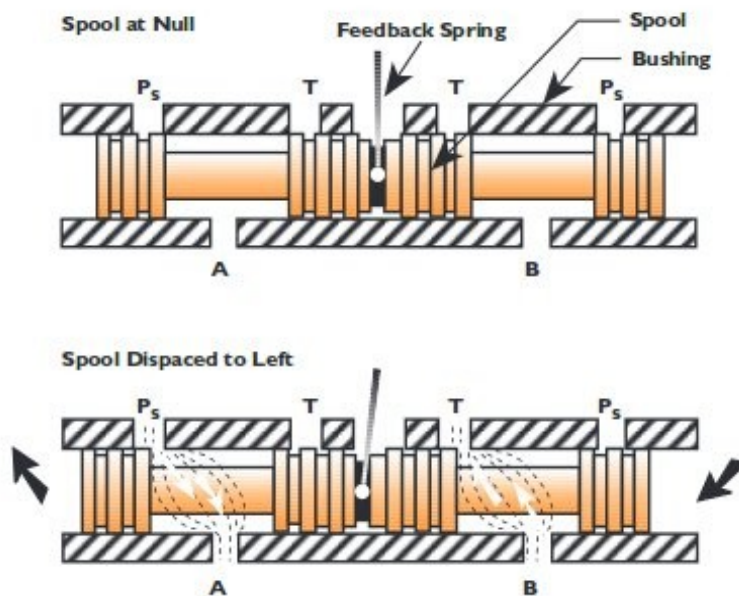


Figure 1: control spool regulating flow orifices

The ports “A” and “B” get connected to opposite sides of an “actuator” piston, as illustrated below:

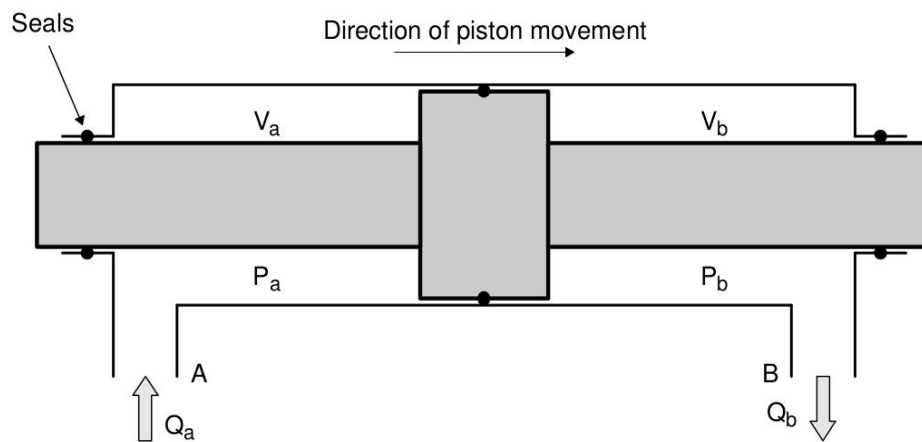



Fig 2: Hydraulic actuator. The output force of this piston depends on the pressures P_a and P_b .

An important source of error is the actuator friction. Per Fig 2, there is a seal on the actuator that prevents fluid leakage from P_a to P_b (or vice versa), in spite of significant pressure differences (e.g. 3,000 PSI). This requires a tight seal (e.g. an O-ring under compression), and this seal introduces significant actuator friction.

2) Setting feedforward gains:

The lowest-level control input to the servo actuator is “i_cmd”, a commanded current that is related to the control-spool displacement. For Atlas, the user does not have direct access to this effort command. Rather, this is exerted by the eBox using a BDI algorithm, described below in Fig 3.



Atlas Robot Interface Reference 2.0.1

Overview	Classes	Files
Class List	Class Members	

AtlasJointControlParams Struct Reference Public Member Functions | Public Attributes

Structure for specifying controller parameters. [More...](#)

```
#include <AtlasInterfaceTypes.h>
```

Detailed Description

Structure for specifying controller parameters.

This structure contains the gains to be applied to a joint.

q, \dot{q}, f are sensed position, velocity, torque, from **AtlasJointState** q_d, \dot{q}_d, f_d are desired position, velocity, torque, from **AtlasJointDesired**

The final joint command will be:

$$k_{q_p} * (q_d - q) + k_{q_i} * 1/s * (q_d - q) + k_{\dot{q}_d_p} * (\dot{q}_d - \dot{q}) + k_{f_p} * (f_d - f) + ff_{\dot{q}_d} * \dot{q}_d + ff_{\dot{q}_d_d} * \dot{q}_d + ff_{f_d} * f_d + ff_const$$

Fig 3: Atlas eBox low-level control algorithm

Data collected from an example motion is shown below in Fig 4. For this motion, the right elbow was oriented with its axis vertical, thus allowing elbow motions without gravity loading. The motion of Fig 4 corresponds to a slow, sinusoidal elbow oscillation (0.7 radians amplitude, 0.2Hz frequency).

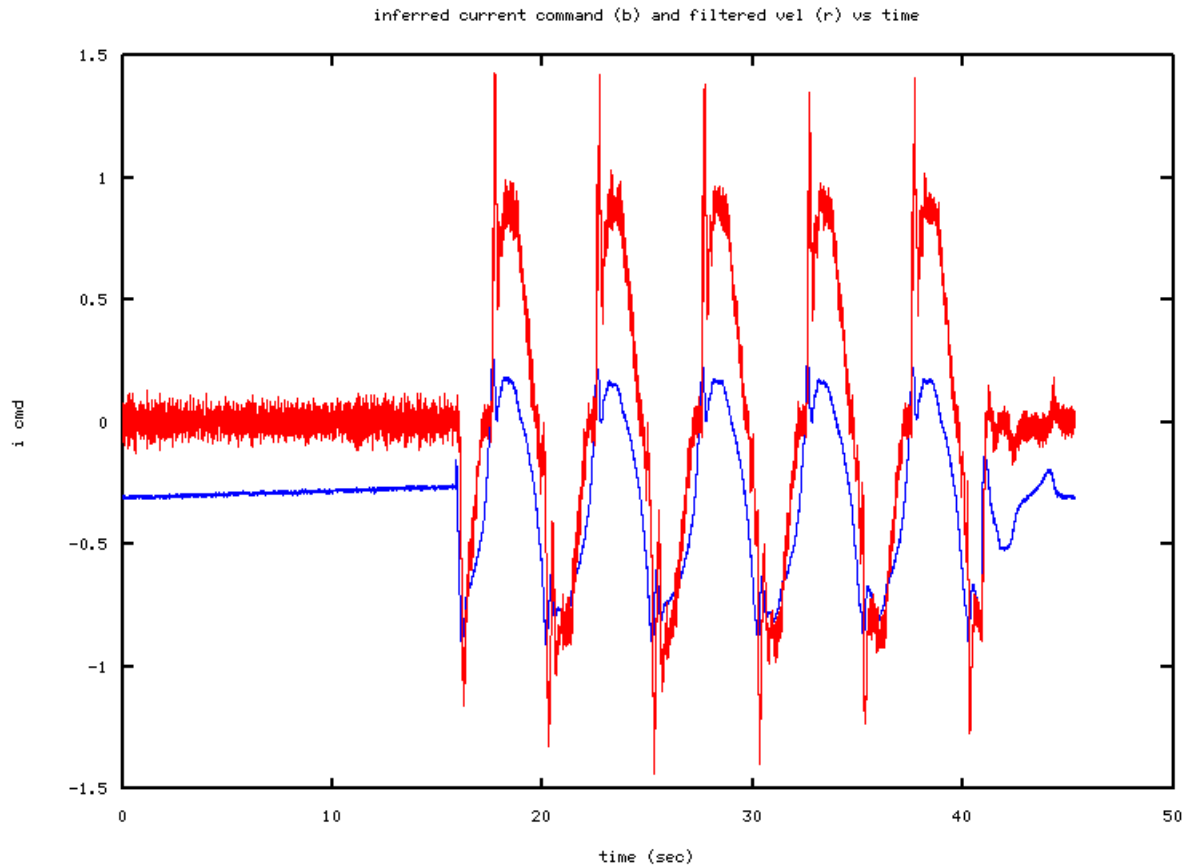


Fig 4: right-elbow velocity (red) and inferred i_cmd (blue) vs time for slow sinusoidal motion

For the example motion of Fig 4, all of the Atlas control gains were set to zero except for $k_q_p=4.0$ and $ff_qd_d=0.5$. Although i_cmd is not a logged quantity, knowing the gains and input commands, we may infer the i_cmd sent by the eBox to the right-elbow actuator as: $i_cmd = k_q_p*(q_command - q_sensed) + ff_qd_d*qdot_command$.

A plot of the reconstructed i_cmd is shown in Fig 4. This figure reveals two important properties. One is that the current command is highly correlated with the elbow angular velocity, and the other is that the current command itself has a bias offset. The relationship between current command and piston velocity is approximately: $qdot = 2*(i_cmd+0.5)$. Equivalently, to achieve a desired velocity of $qdot$, one should exert:

$$i_cmd = 0.5*qdot - 0.5$$

This can be achieved by setting the BDI feedforward gain to: $ff_qd = 0.5$, and correcting the i_cmd bias with: $ff_const = -0.5$. Some level of control can be achieved using only these feedforward gains. However, better control can be achieved with feedback, in combination with these feedforward contributions.

3) Effort feedback:

In Atlas' joints, the pressures on both sides of each actuator are measured, the pressures are multiplied by the respective piston-head areas (which, unlike Fig 2, may not be symmetric), and the difference between these quantities is reported as the inferred force on the actuator (piston). The net force on the actuator piston, in turn, acts through some transmission ratio (which, for the legs, is a function of angle). The result is reported out by Atlas as the joint's “effort” on the atlas/atlas_state topic. It is unknown if the units reported are calibrated (e.g. N-m of torque).

Figure 5, below, shows the “effort” reported by Atlas while controlling the right elbow to move horizontally in a slow, sinusoidal motion for 5 cycles. For this motion, the arms were pre-positioned such that there was no gravity load on the elbows. Figure 5 reveals two important issues: the reported efforts have a significant bias offset calibration error, and the effort required to overcome Coulomb friction (presumably from the actuator seal) is substantial.

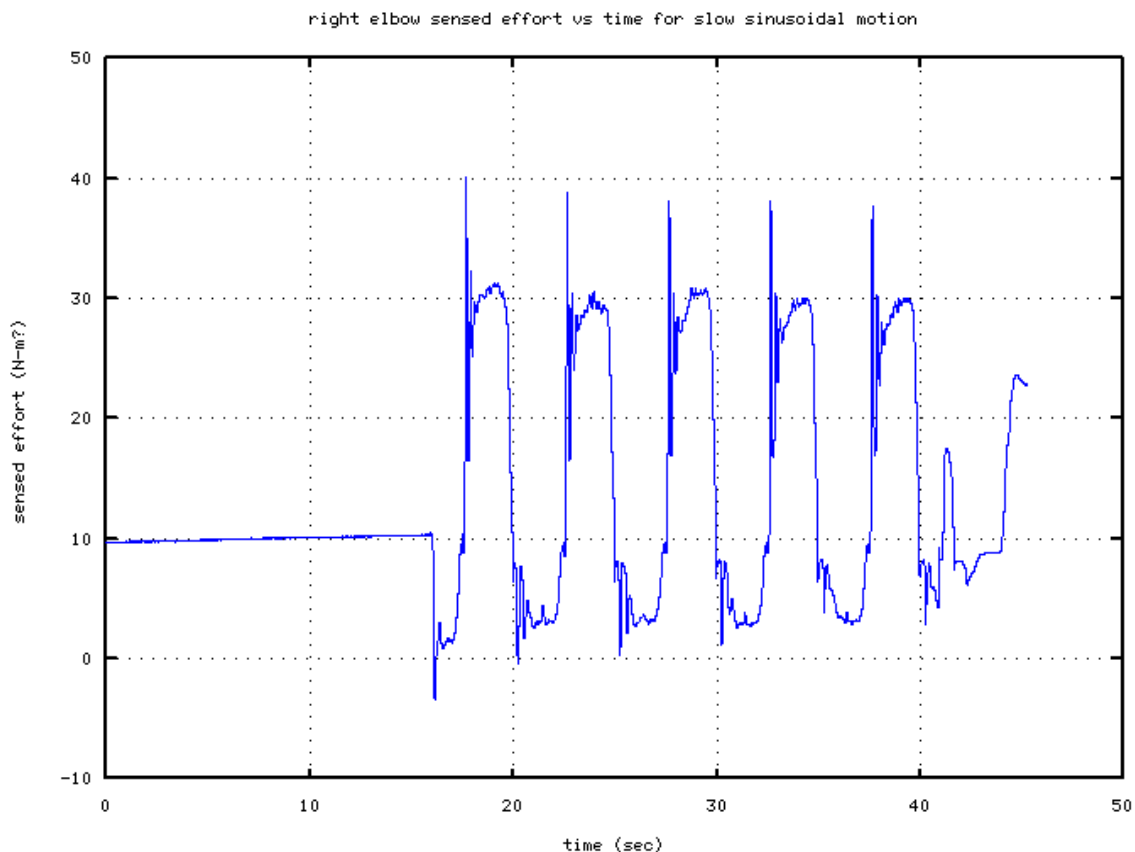


Fig 5: Atlas-reported actuator effort for right elbow, moving horizontally in slow sinusoid

To improve the control of this joint, it is important to model and compensate for the bias in the reported effort and to inject compensation for the substantial Coulomb friction. For Fig 5, the right elbow shows an offset bias of approximately 15.5 N-m and a Coulomb friction of ± 13 N-m.

If one could control the differential pressure (i.e. the “effort”) directly, then the desired, reported effort should be 30 N-m for forward motion and 4 N-m for reverse motion. Unfortunately (unlike a dc motor input), direct control of torque (equivalently, pressure difference) is not available—only flow orifice regulation. With feedback, however, the current command can be adjusted to try to achieve some desirable effort. This is achieved for Atlas through the use of the k_{f_p} gain.

If it is desired to achieve some effort f_d , then the objective is to make the sensed effort, f , converge on the desired effort, f_d . To do so, the current command to the electrohydraulic actuator can be augmented with the quantity: $i_{cmd} += k_{f_p}(f_d - f)$, as per the BDI algorithm of Fig 3.

Experiments were performed on Atlas' left elbow to try to determine a good value for k_{f_p} . Values of 0.25 and above resulted in oscillations, but a value of $k_{f_p} = 0.2$ yielded good results. Figure 6 shows data logged for a sinusoidal left-elbow motion of 1Hz with an amplitude of 0.5 rad.

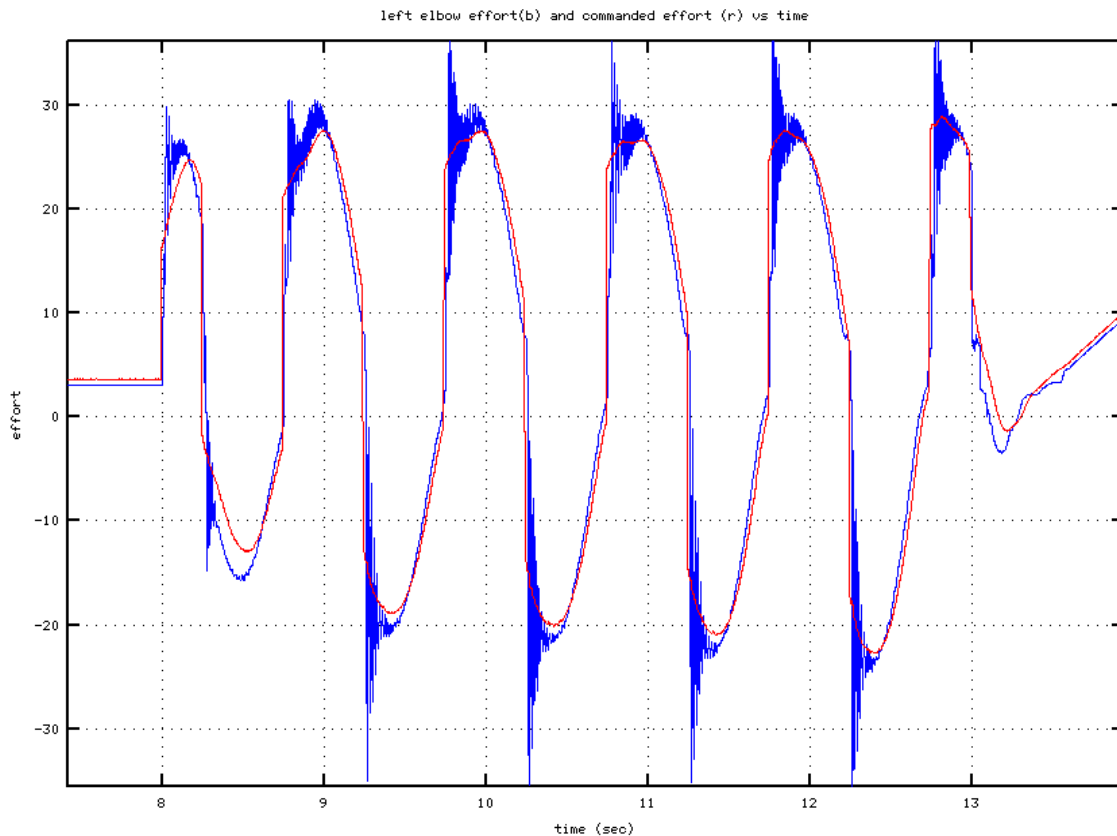


Fig 6: Left-elbow effort tracking with feedback $k_{f_p} = 0.2$. Red = commanded effort; blue=measured effort.

As seen in Fig 6, the measured effort (blue trace) does a good job of following the commanded effort (red trace). The commanded effort included pre-compensation for Coulomb friction, and thus the discontinuous jumps at velocity reversals. The commanded effort was also offset to pre-compensate for the effort sensor bias. The measured effort values slew rapidly in response to step changes in commanded effort, although there is some overshoot and ringing. Nonetheless, this appears to be an attractive response.

Collectively, the the BDI algorithm should be used with tuned values of: i_{cmd} bias offset (e.g. $ff_{const} = -0.5$ for the right elbow—not tested yet), velocity feedforward ($ff_{qd_d} = 0.5$ for both left and right elbows), and effort feedback ($k_{f_p} = 0.2$ for the left elbow—not yet tested for the right elbow). The remaining gains for computation of i_{cmd} do not have a logical mapping. Default elbow gains for user control of Atlas were $k_{q_p} = 4.0$ and $k_{qd_p} = 0.0$, which result in poor performance. Increasing these gains also results in poor performance.

It is recommended that *only* ff_{const} , ff_{qd_d} and k_{f_p} be set to non-zero values, and that

feedback on position and velocity be performed at a higher (field computer) level. Note that the ROS atlas_command does not allow for setting all of the desired values in the BDI control algorithm. Further, once identified, these values should not change (or would change infrequently —e.g. bias drift of the spool offset, or reduced Coulomb friction with seal wear). These values should be set in the

4) Field-computer P-D feedback:

With appropriate gains set for the eBox, Atlas can do a good job of adjusting the control spool such that sensed efforts follow commanded efforts. Additionally, the `i_cmd` bias term, `ff_const`, corrects for the apparent fact that the spool is not precisely centered at `i_cmd=0`. Further, the feedforward gain `ff_qd` help to correct for pressure drops through the flow orifices as function of required flow rate (i.e., equivalent to desired actuator velocity). Collectively, these terms should achieve good tracking of sensed effort following commanded effort.

At a higher level, one should request a sensed effort that is logical. Desirably, one would want to achieve a target joint torque (albeit as one step towards an end). The net joint torque is not the same as the Atlas sensed effort, however, because of pressure-sensor bias and actuator-seal friction. Assuming the effort-sensor bias has been identified as `e_bias` (e.g. `e_bias = 15N-m` for the right elbow) and the Coulomb friction magnitude, `e_Coulomb`, has been identified (e.g., `e_Coulomb = 13N-m` for the right elbow), then one can pre-compensate a commanded effort to achieve a desired net effort.

If the desired net effort is `e_net`, and if the desired angular velocity is `qdot_des`, then one should command:

$$f_d = e_{net} + e_{bias} + e_{Coulomb} * \text{sgn}(\dot{q}_{des})$$

where `sgn(x)` is the “signum” (or “sign”) operator.

Pre-compensation for commanding a combination of effort and velocity is performed on the field computer, an example of which is in: `elbow_test_servo_callback.cpp`. Relevant lines for bias and Coulomb compensation are:

```
//Coulomb compensation:
e_ffwd = 0.0;
if (qdot_des>0.0)
    e_ffwd=Coulomb_effort[i];
if (qdot_des<0.0)
    e_ffwd= -Coulomb_effort[i];
e_cmd+=e_ffwd;

//bias compensation:
e_cmd+=effort_bias[i];
```

The above lines pre-compensate an effort command, `e_cmd`, to achieve some desired net effort.

Assuming the preceding code has been implemented and tuned, the corresponding Atlas joint will approximately exert a net torque on demand. Deciding what torque is appropriate is the job of a higher-level layer. The most common control algorithm is proportional-plus-derivative (PD), expressed as:

$$u = K_p(q_{des} - q) + K_v(\dot{q}_{des} - \dot{q})$$

where “`u`” is the control effort to be exerted (`e_net`, in the above) and `Kp` and `Kv` are position and velocity (or derivative-of-position) gains, respectively.

To tune these gains appropriately, one should have a dynamic model of the system to be controlled. A simple model of an Atlas joint assumes the following: “u” can be exerted as a torque on demand; gravity loads are negligible; there are no other external forces on the system. The assumption of exerting “u” on demand is achieved to a good degree with the preceding feedback. Ignoring gravity and external forces is valid for the simple case of horizontal motion of the forearm and no contact with the environment. Inclusion of gravity loads and contact forces require additional computations, but may be ignored for the present. With these assumptions, the dynamic model is:

$$u = m \cdot \ddot{q} \text{ (i.e., } F = m \cdot a \text{)}$$

The term “u” is elbow torque (N-m), “ \ddot{q} ” is elbow angle acceleration ($\text{rad}/(\text{sec}^2)$), and “m” is inertia (Kg-m^2). For this simplified system model, the primary parameter to be identified is the inertia, m.

Figure 7, below, shows logged data from a 1Hz left-elbow oscillation of amplitude 0.5 rad. The greenline is the velocity, as reported by Atlas. The red trace is a post-processed, filtered version of the velocity samples. The blue trace, overlayed on the measured velocity, was computed based on the recorded “effort” values, adjusted for bias and Coulomb friction to infer a net torque, then integrated with a scale factor of $1/m$. That is, it was desired to explain the observed velocity waveform in terms of the measured effort values and an assumed $F=m \cdot a$ model. The value of m was adjusted empirically to try to get a good fit to the data. (Offsets and drift of the blue, computed velocity prediction are not affected by m and may be ignored). A reasonably good fit was obtained at a value of $m=0.67$.

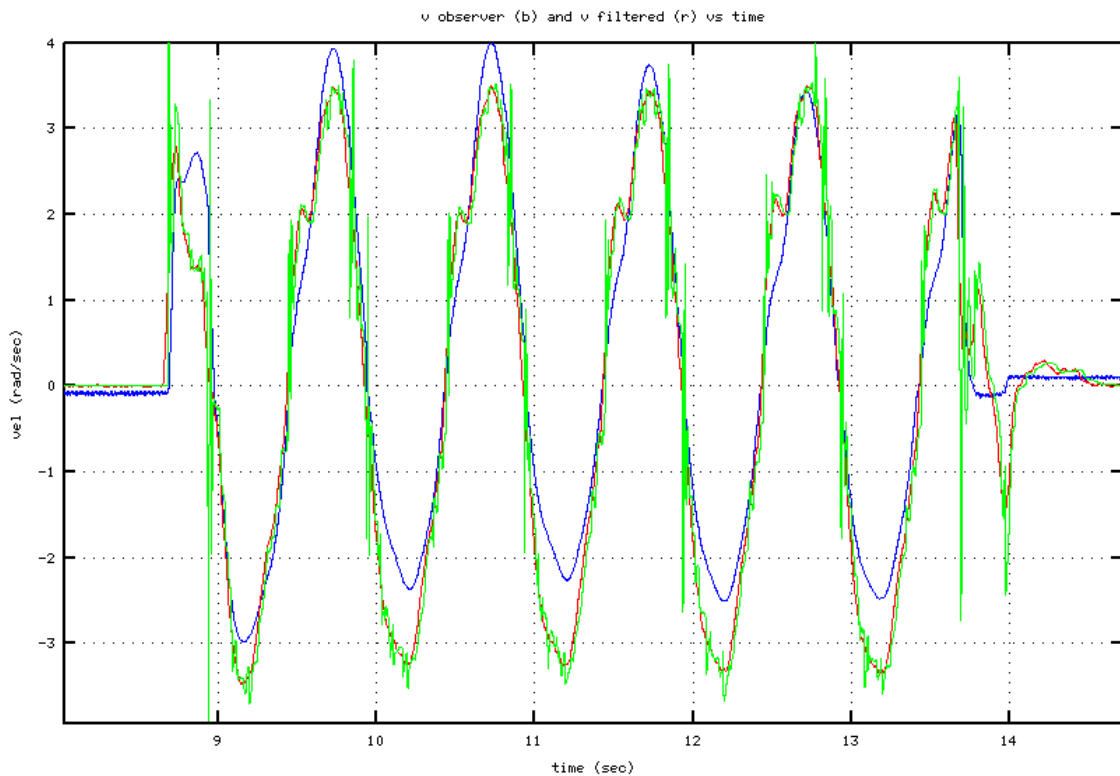


Figure 7: Graphical display of identification of system inertia for left elbow

The units of the reported effort are unknown. They are presumably N-m, but this has not been verified. Nonetheless, the parameter $m=0.67$ relates net effort (in BDI units) to resulting angular acceleration (in $\text{rad}/(\text{sec}^2)$) for the left elbow (and presumably the right elbow). Note: this data was acquired with no hands mounted on the wrists.

Given a (now linearized) dynamic model of: $u_{net} = (0.67) \cdot \ddot{q}$, one can choose gains for a predictable response. For a second-order system, the closed-loop (controlled) response to behavior is described by:

$$m \cdot \ddot{q} = K_p(q_{des} - q) + K_v(\dot{q}_{des} - \dot{q})$$

Such a system has a critical frequency, ω_n , (roughly speaking, closed-loop bandwidth) of:

$$\omega_n = \sqrt{K_p/m}$$

and a damping factor, ζ , that follows from:

$$2 \cdot \zeta \cdot \omega_n$$

Choosing values of $K_p = 400 \cdot m$ and $K_v = 40 \cdot m$ translates into $\omega_n = 20$ rad/sec, and $\zeta = 1$, or about a 3-Hz closed-loop position-control bandwidth with critical damping. The system response with these gains for the left elbow, with a commanded 1Hz sinusoidal motion of amplitude 0.5rad is shown in Fig 8.

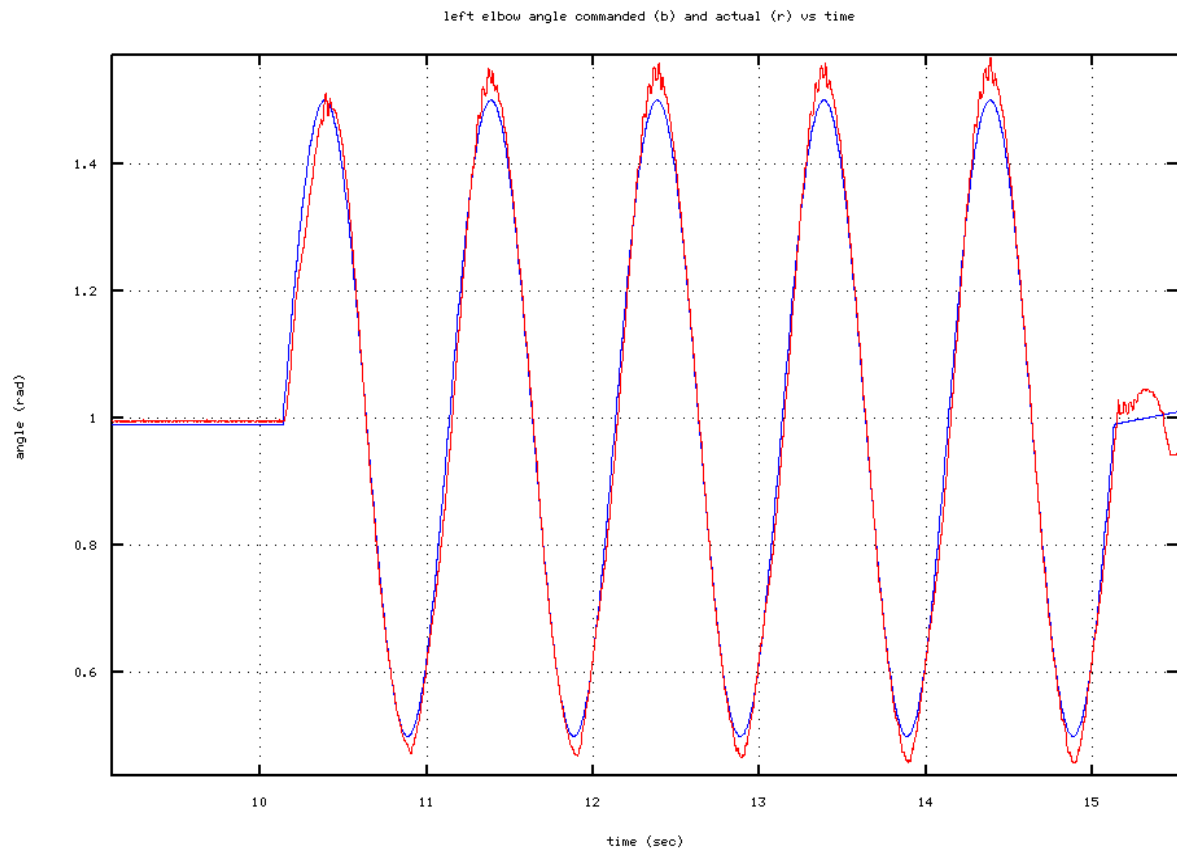


Figure 8: Commanded (blue) and actual (red) motion of left elbow under closed-loop control with $K_p=400 \cdot m$, $K_v=40 \cdot m$

The corresponding velocity command and velocity response are shown in Fig 9.

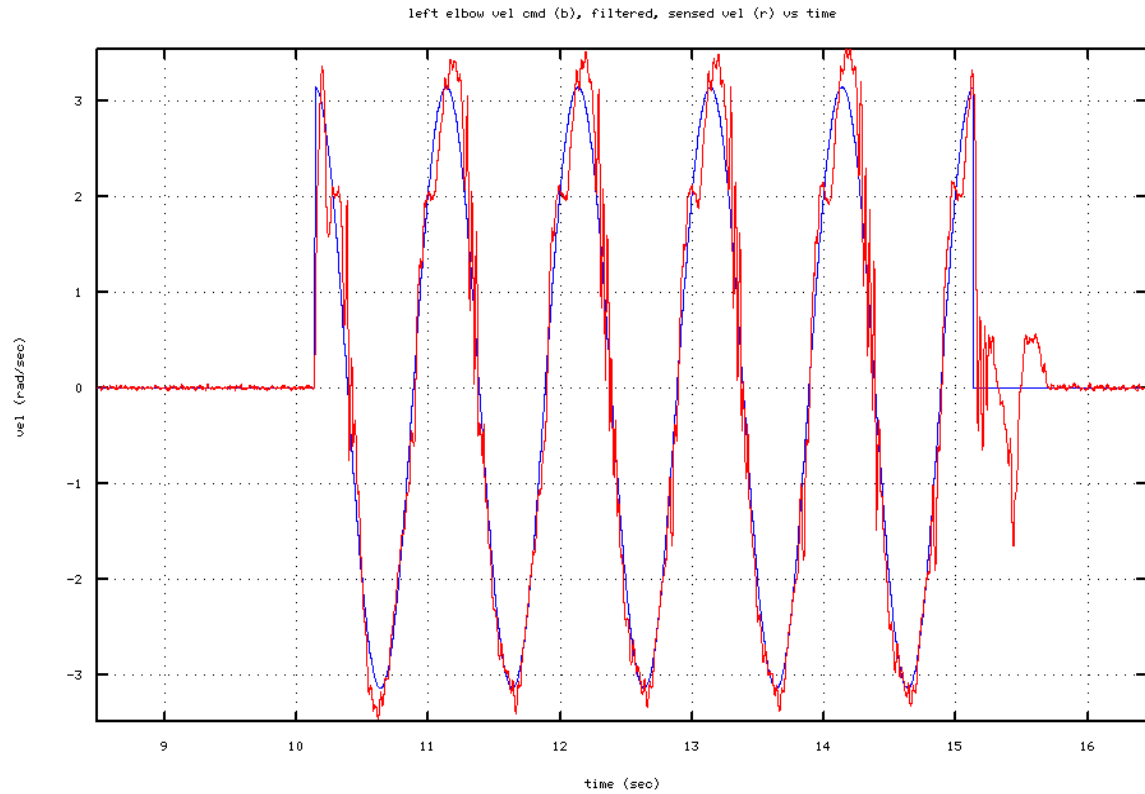


Figure 9: Commanded (blue) and measured (red) velocities for 1Hz sine wave, left elbow, with $K_p=400 \cdot m$, $K_v=40 \cdot m$

5) Next steps:

The identification methods described here should be repeated for all Atlas joints that will be under the control of the user. The i_cmd bias correction has not yet been tested for any joint. It may be that the effort feedback gain of 0.2 is applicable to multiple joints, but this is not yet known. Bias terms, velocity feedforward terms and inertias should be identified for all joints of interest. It may be that $K_p=400 \cdot m$ and $K_v=40 \cdot m$ are also applicable gains for all joints, although this is not known yet.

A gravity model should be constructed to account for gravity loads in control. An “observer” has been constructed, but it has not been demonstrated if this is necessary, or if the reported velocity measurements would be adequate.

The “effort” outputs should be calibrated in SI units, and the identified inertias should be recomputed accordingly.

Ultimately, it is desired to create a compliant-control mode for the arms. This will require first identifying the above quantities, finding the maximum practical feedback gains K_v , then testing a “Natural Admittance Control” law.