The ATA Servo Loop

G. R. Harp
11/1/6

Abstract

Here we document some of the details of the antenna tracking software (servo loop). The servo loop consists of two parts – estimation of the current antenna position and evaluation of the antenna velocity setting to optimize response. The antenna position estimation is done with a Kalman filter applied to encoder readings. The control parameter (velocity setting) is evaluated with a PDFF (position, velocity, velocity feed forward, acceleration feed forward) algorithm fed from a low pass prefilter.

Introduction

To achieve optimal ATA performance we require a good pointing model so that we can tell the antennas where to point with high accuracy. But that isn't enough; we also need a high quality feedback mechanism to keep the antenna stably pointed in the correct direction. In cases where we wish to image a large field of view (FOV), we also require the antenna to move nimbly from one pointing to the next, so we can cover a large area while minimizing settling time. This document focuses on the feedback loop presently used in the ATA antenna (note date above). We begin with a discussion of some parameters that constrain the servo loop, such as the encoders and antenna mechanics. We then describe the algorithms for position estimation and calculation of feedback parameters used to drive the antennas in real time.

Constraints

Lowest Antenna Resonant Frequency

It is fairly easy to make a good estimate of the lowest resonant frequency of the antenna mechanics. Mike Davis and Rick Forster drove antennas by hand, feeding back the strongest resonance with vigorous shaking. By counting the number of oscillations over some seconds and dividing by the time, they found the antennas resonate at about 3 Hz in both azimuth and elevation. This is the lowest frequency resonance, and although higher resonances exist we can avoid exciting any resonance by filtering the servo drive commands to avoid all frequencies near 3 Hz or above. This is a low-pass filter.

Incremental Encoders

The ATA antennas employ directly-coupled, incremental encoders with an encoder step size of 9 arc seconds. These encoders are read by dedicated hardware on the control board (designed by Rob Ackermann), and encoder readings are issued from this board at a rate of 10 Hz to the control software.

Antenna Velocity Drives

The antenna is moved with DC motors driven by Copley servo amplifiers. Although the Copley drives contain their own slew of servo controls and software, we don't use this functionality because the servo loops are too "tight" to be useful. We want telescope feedback on the scale of seconds, and indeed wish to avoid driving the 3 Hz resonant mode of the antennas.

The maximum slew speed of the Azimuth drives is nominally 4°/s. Some antennas are observed to stall at this speed, so the software limits the Azimuth velocity to 3°/s. We expect that this mechanical problem will be fixed eventually, and the software limit will be raised at that time. The elevation velocity is presently limited to 1°/s.

The servo drives are controlled by sending step pulses to the Copley amplifier. This is the same interface used by the old, stepper motor drives. We decided to stick with this interface because it provides versatility for the future. The control steps are very small, so there is no limitation introduced by the stepper interface (see table).

Useful constants	
Az Degrees per Servo pulse	1.8803 E-04
El Degrees per Servo pulse	(12.925e-4 * el_degrees + 0.411459) / 5000
Encoder Degrees per encoder step	0.00625

Position Estimation: The Kalman Filter

The control board generates values for the antenna position in azimuth and elevation at a rate of 10 Hz. The 9 arc second encoder resolution provides one limitation on our position estimate. Theoretically, one can beat this resolution limit by interpolating between encoder "ticks" based on recent estimates of the antenna velocity. Velocity measurements are limited not only by the encoder resolution but also by our estimate of when the encoder ticks occur. In practice this is limited to about 0.1 second by the reading rate mentioned above.

To estimate the position, we need a theoretical model of the antenna motion, even if it is very simple. For example, we could model the position as a random walk, where the size and direction of the next step is completely uncorrelated with the size and direction of the last step. The reason our model incorporates randomness or noise, is that A) the antenna is subject to external forces like wind, and B) the antenna is subject to the whims of the astronomer, who may decide to point in a completely different direction at any time. Thus a random walk is the simplest possible nonstationary motion model. However, we expect the antenna has inertia, so we look for a model that incorporates this fact.

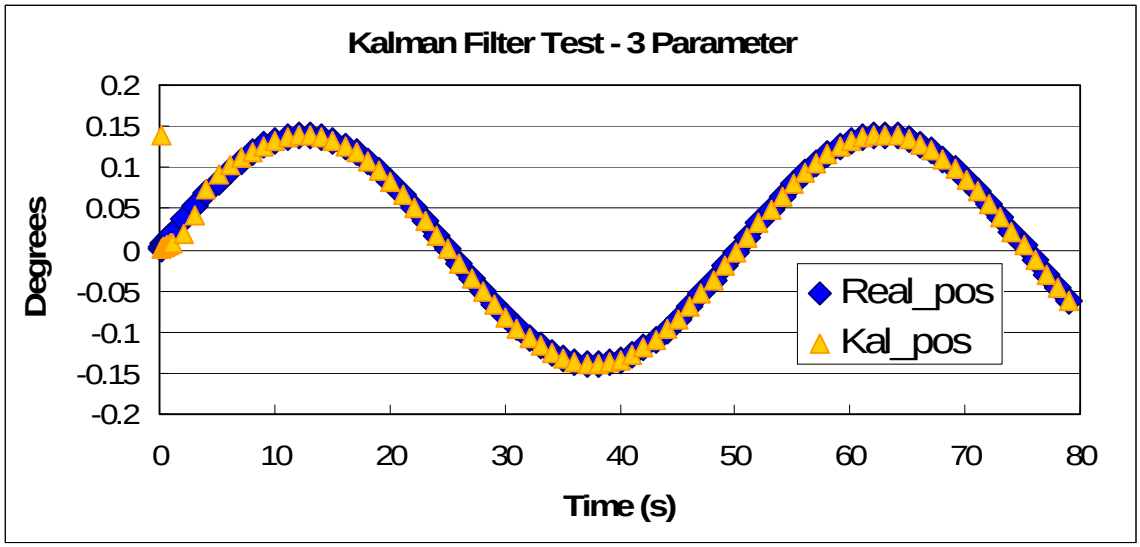
One way to introduce inertia is to integrate a random walk. More specifically, we suppose the antenna has a certain mass and suffers random, Gaussian-distributed momentum

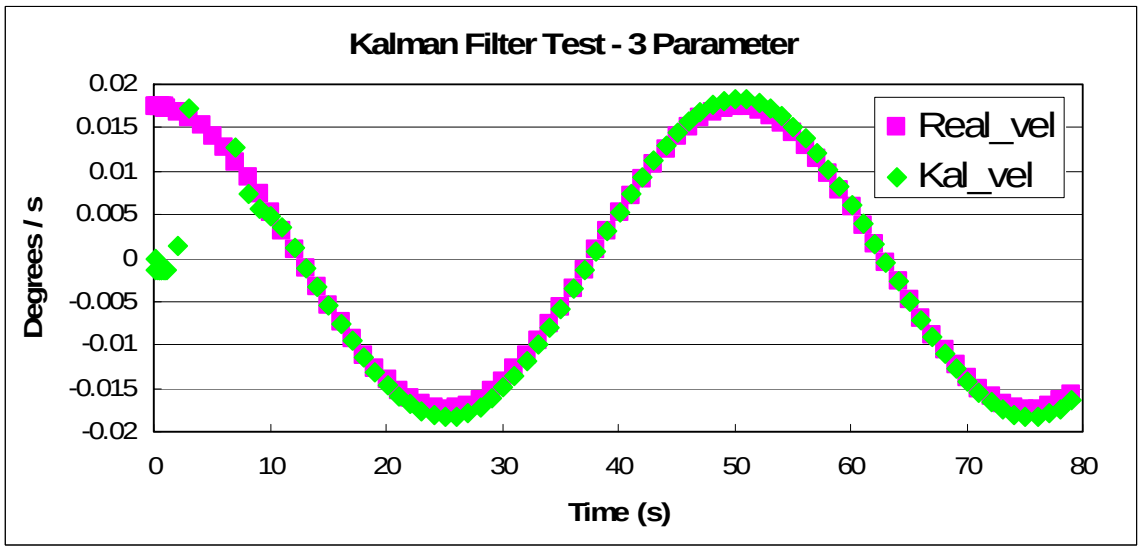
impulses. Or following the physics of random winds, we might suppose the antenna suffers Gaussian-distributed forces (acceleration impulses). Other models are possible.

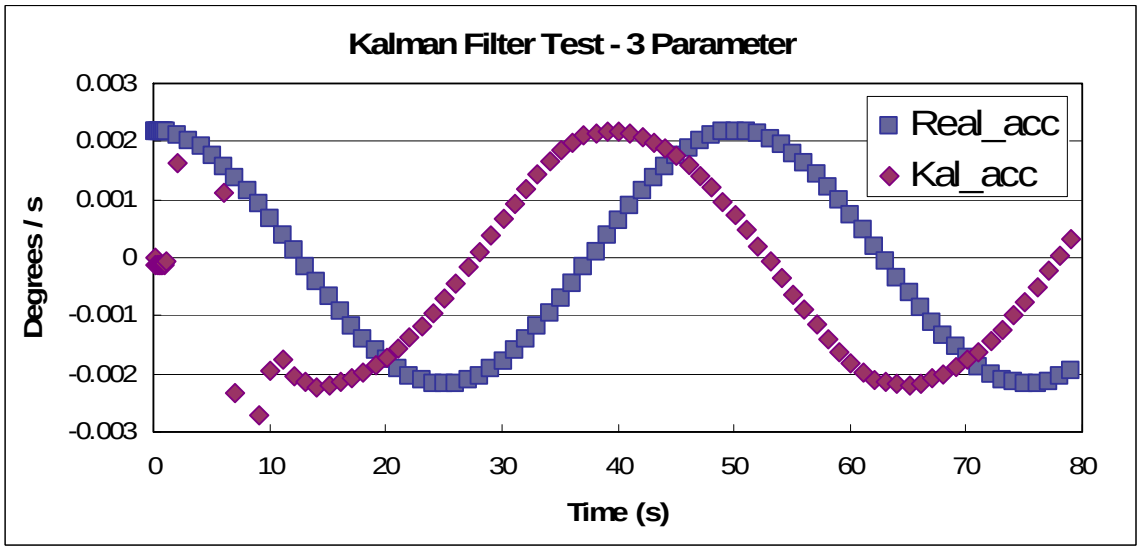
Once we have chosen our model, we may implement the model in a Kalman filter. The brilliance of Kalman's result is that, given a set of time-ordered measurements and a parameterized (linear) model, the Kalman filter gives a least squares best estimate of the model parameters. Many least-squares techniques exist, but Kalman's prescription allows us to apply measurements incrementally as they come in, always maintaining the present best estimate. We won't discuss how it works here, but we have found to be very useful the book, "Kalman Filtering: Theory and Practice", by M. S. Grewal and A. P. Andrews.

We compare two Kalman filter implementations corresponding to the random momentum impulse (2 parameter filter) or random acceleration impulse models (3 parameter filter). These two implentations are driven with numerically-generated sinusoidal position values (that simulate the kind of trajectory accelerations we intend to track) with realistically simulated encoder noise. Here we're optimizing for the "no wind" case.

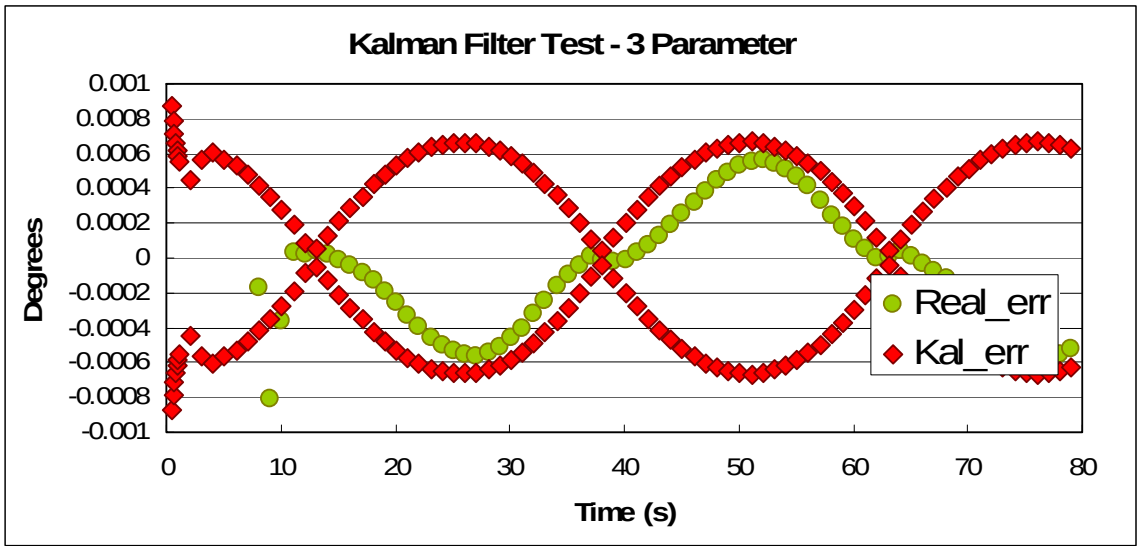
The first graphs show tracking of position, velocity, acceleration, and the tracking error for the 3 parameter filter. The 3 parameter filter does a good job of tracking the antenna position, with position errors of less than 4 arc seconds, after a settling period in the beginning. This substantially beats the encoder resolution limit despite the accelerations and noise. Notice that the error drops almost to zero when the acceleration is zero.





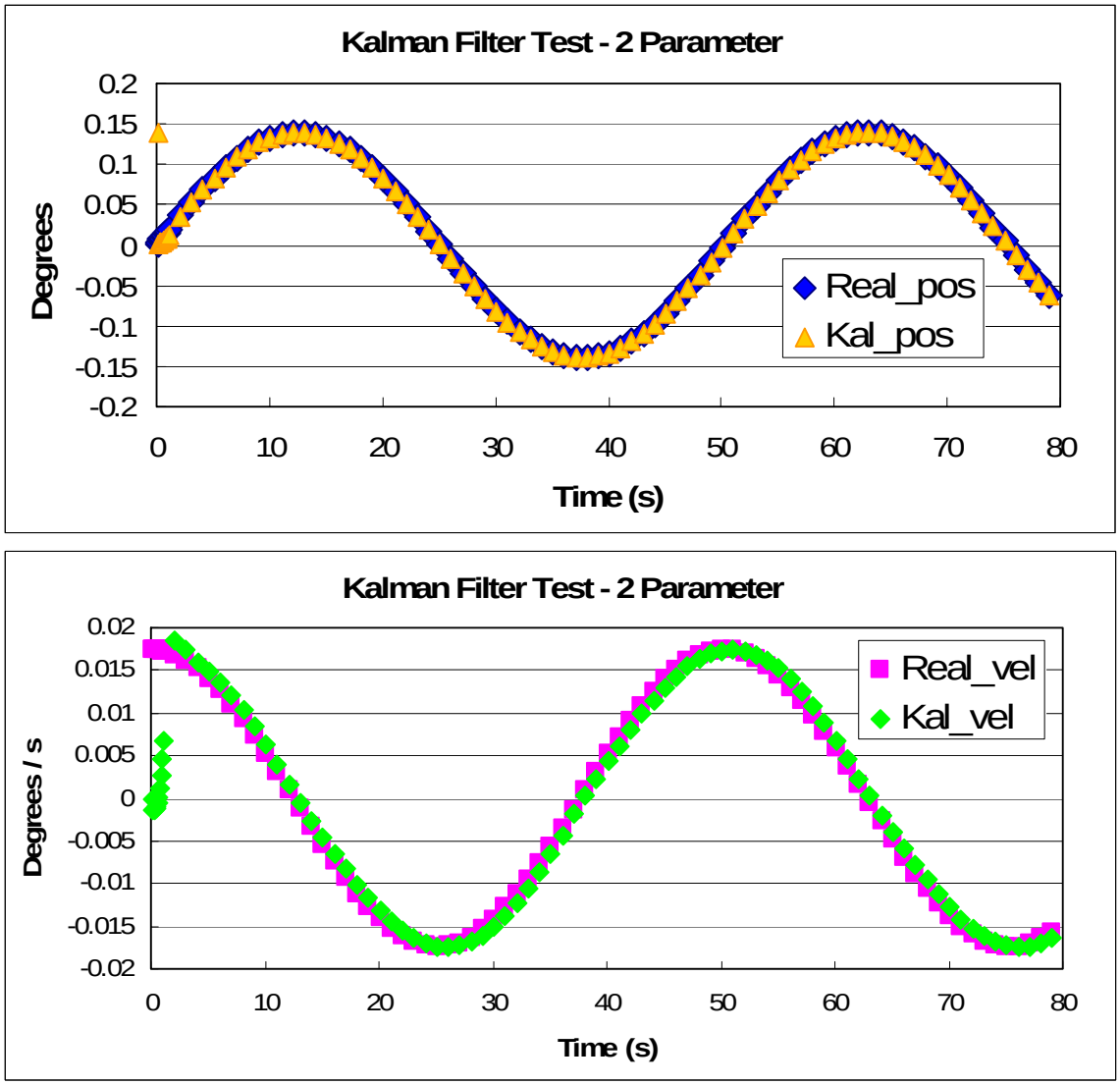


The position, velocity and acceleration estimates generated by the 3-parameter filter.

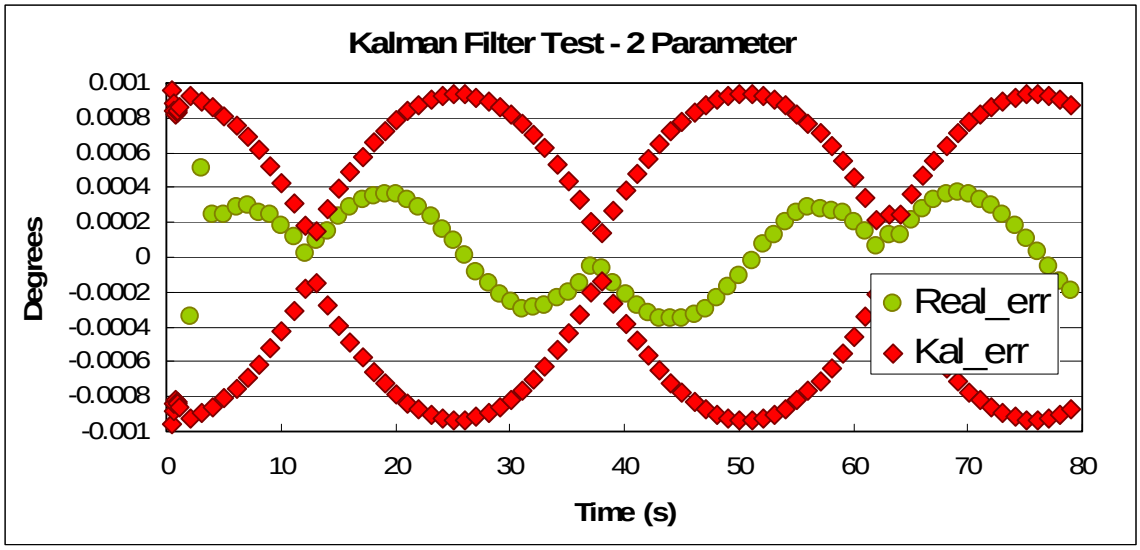


The difference between the filter estimate of position and the "real" position in our simulation (circles). The diamonds show the Kalman filter estimate of the position error, which is similar to and a little larger than the true error in all cases.

The next 3 graphs show the position and velocity tracking, and the tracking error for the 2 parameter filter with the same inputs.



Position and velocity as estimated by the 2 parameter Kalman filter. This filter does not estimate the antenna accelerations.



The difference between the filter position and the "real" position (circles) for the 2 parameter filter. The diamonds show the Kalman filter estimate of the position error.

The performance of the 2 parameter is filter is similar but somewhat better than the 3 parameter filter, as seen by comparing the error plots. The 2 parameter filter settles more quickly (5 s) than the 3 parameter filter (10 s). After settling down, the 2 parameter filter shows smaller error excursions than the 3 parameter filter. Notice that in both cases, the Kalman filter does an adequate job of estimating the bounds of the position error (red diamonds).

Why should the 2 parameter filter perform better? It has a shorter handle on the measurements than the 3 parameter filter. Put another way, the position errors resulting from the encoder resolution and time resolution are more accurately modeled as momentum impulses than as force impulses. If we were optimizing for tracking in wind, we might find the 3 parameter filter has a more appropriate model.

Based on these simulations, we choose the 2 parameter model presently. One might supplement these results with measurements on the real antenna, in the future.

Servo Loop

In software we use a PDFF (position, velocity, velocity feed forward, acceleration feed forward) servo algorithm. The inputs are the desired antenna trajectory (position, velocity, and acceleration), and the current Kalman estimate (position, velocity) of where we are now. We have a discrete time implementation, where time is represented as n for the n^{th} time interval.

The servo velocity setting at the n^{th} interval v_s is dependent upon the estimated position $\hat{p}_n$ and velocity $\hat{v}_n$ at the n^{th} interval as

$$v_s = K_p(p_n - \hat{p}_n) - K_D\,\hat{v}_n + K_{\text{vff}}\,v_n + K_{\text{aff}}\,a_n \qquad [1]$$

where p_n, v_n, and a_n are the trajectory position, velocity and acceleration, and K_p, K_D, K_{vff}, and K_{aff} are respectively the gains for the position, derivative, velocity feed forward, and acceleration feed forward. It is possible to write this expression more intuitively (at least for the author) by defining Δt and K_{v} using

$$K_p \, \Delta t \equiv K_D$$

$$K_p \, \Delta t + K_v \equiv K_{\text{vff}}$$

and applying the extra constraint

$$K_v \, \Delta t \equiv K_{\text{aff}} .$$

Substituting these expressions into Eq. 1 gives

$$v_s = K_p (p_{n+1} - \hat{p}_{n+1}) + K_v \, \Delta t \, (v_{n+1} - \hat{v}_{n+1}) \qquad [2]$$

where

$$\hat{p}_{n+1} = \hat{p}_n + \hat{v}_n \, \Delta t$$

and

$$v_{n+1} = v_n + a_n \, \Delta t .$$

Equation 2 takes the perspective of minimizing the future errors rather than the present error, which after all is already history. We choose a velocity based on our prediction of the future position error with gain K_p, and our prediction of future velocity error with gain K_v. In our implementation, the velocity feed forward and acceleration feed forward terms are no longer unrelated and *ad hoc*, but are replaced with a single term associated with the velocity we wish to achieve in the future. The "future" in this case is a time Δt from the present time.

Notice that Δt is not necessarily the same as the time spacing between servo updates Δt_s. It may not be smaller than Δt_s or the system will become unstable. We find that setting Δt = 4/3 Δt_s gives good response time. The ratio K_p / K_v controls the tradeoff between position error and velocity error. Minimizing position error ($K_p >> K_v$) tries to match the encoder positions with the desired trajectory, but the motion can be jerky. Minimizing velocity error emphasizes smooth tracking but increases settling time. We choose

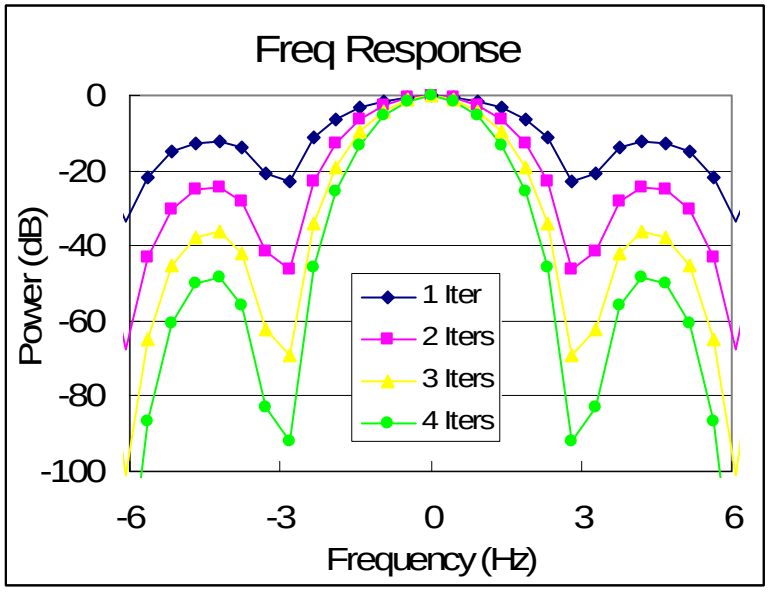
$K_v = K_p$ based on empirical tests to give a happy medium for these qualities. The only remaining factor, tuned at run time, is K_p.

The time between servo updates is Δt_s = 0.3 seconds chosen to be 3 times larger than the time between encoder readings. Thus the time constant of the servo loop is set to 0.4 sec.

Prefiltering the Input Trajectory

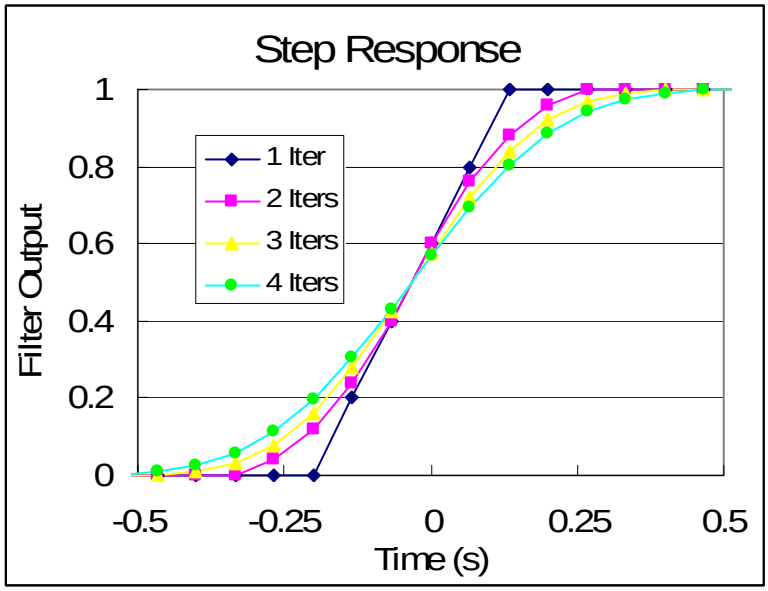
Before the antenna trajectory is sent to the servo loop, we apply a low pass prefilter to remove frequency components at 3 Hz and above. This is done to minimize excitation of resonances in the antenna. We also need to limit the velocity of the trajectory we ask the antenna to follow, so that it is within the antenna capabilities. This situation arises when we change pointings, at which time the theoretical trajectory changes discontinuously. We must not allow these discontinuous changes to pass through to the servo input since they can result in instability of the servo loop.

To implement this prefilter, we first calculate the desired, future positions of the antenna on a grid of equispaced points sampled at roughly 10 times the stop frequency (here stop frequency is 3 Hz and sampling is 27 Hz). These calculations take account of the astronomer's desires but, especially in the case of a pointing change, limit the velocity to achievable values (in the example below, to 4°/s). The low pass filter is implemented with an iterated boxcar filter applied to these samples (i.e. a finite impulse response filter applied in the time domain). The envelope of boxcar filter has a slow 1/f frequency roll off, but by iterating the filter we can improve high frequency suppression dramatically as in the graph below. For example, a 3 iteration filter suppresses all frequencies above 3 Hz by more than 35 dB.



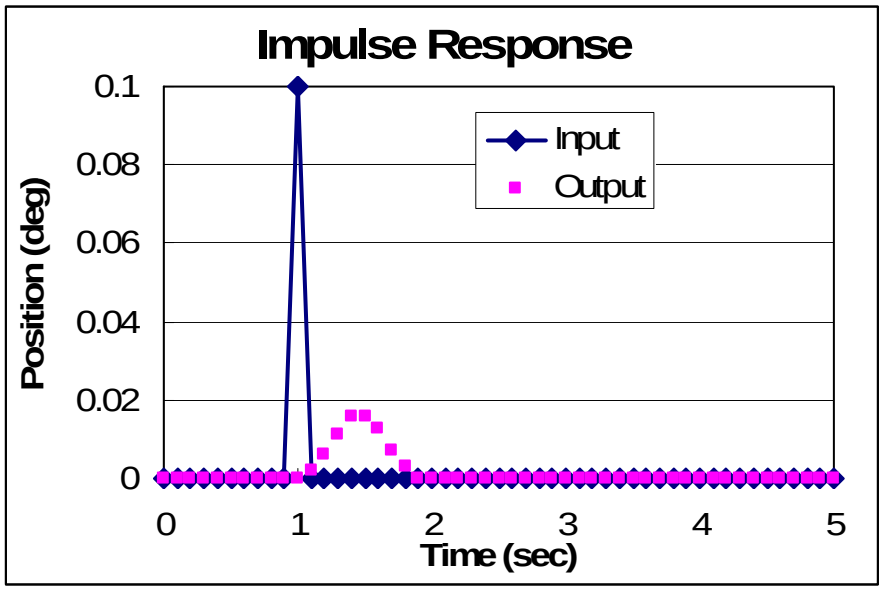
Frequency response of an iterated boxcar filter as a function of the number of iterations. The boxcar length is chosen to have its first null at 3 Hz.

One may ask why we choose anything related to a boxcar filter. Certainly other numerical filters exist that have better frequency response. The reason is that an iterated boxcar is predictably localized in the time-domain. Consider the step function response in the graph below. With a 3-iteration filter, the step response error goes to exactly zero after 0.8 seconds. Therefore, it should be possible to make 1 degree position change in ~1 second, without exciting the 3 Hz resonance at all (< -60 dB).



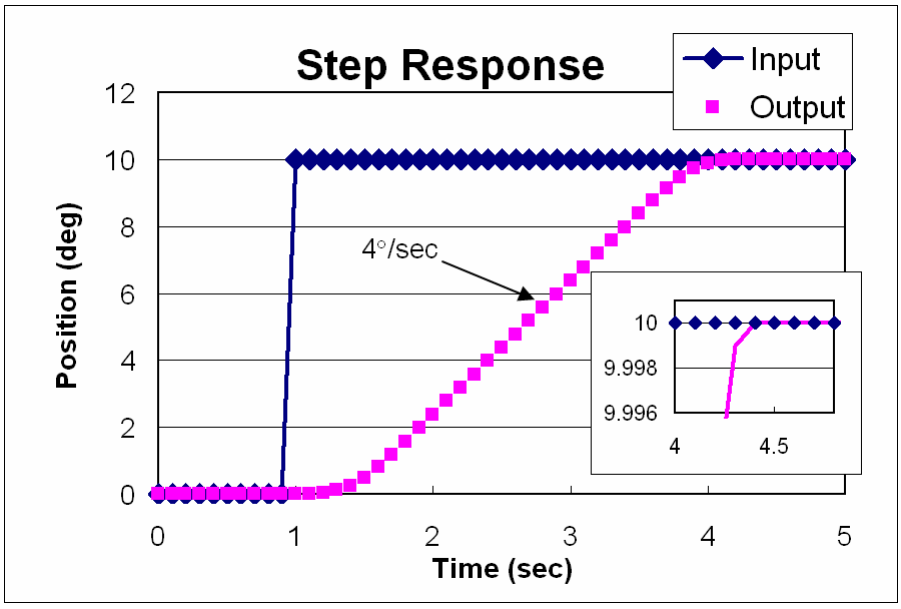
Response of an iterated boxcar filter to a step function at t = 0.

Below we show a couple of results demonstrating our implementation. In the first case, we consider a 0.1° impulse. This is not an ordinary trajectory input for the antenna – it is as if we change pointing for a fraction of a second and then change our mind and go back to the original pointing – but it demonstrates the behavior one can expect. Firstly, the output response is time-shifted from the input. This happens because we don't know in advance that we're going to change our mind. After we get started moving to the new track, we're committed to go at least partway there. This sense of being "committed" comes from the filter which does not allow high frequency components to pass through.



The response of prefilter to a position impulse.

Next we consider the response 10° step change in pointing. This step is large enough that the velocity limiter part of the filter has to kick in. The antenna achieves its maximum velocity of 4°/sec in less than one second, and the "error" goes exactly to zero in less than 4.5 seconds.

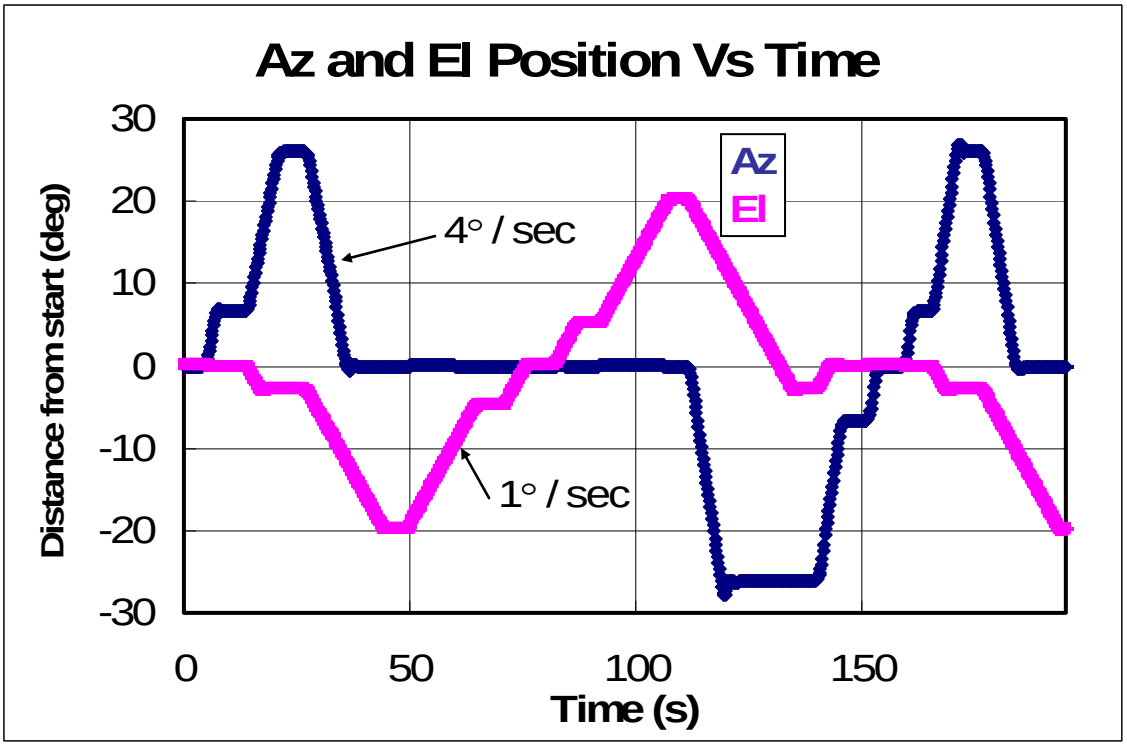


The response of prefilter to a 10° step function.

This example is an idealization to the response of the real antenna to a 10° pointing change. There is an extra latency introduced by the PDFF servo loop, another latency from the Kalman filter and yet another small latency from the encoder read back.

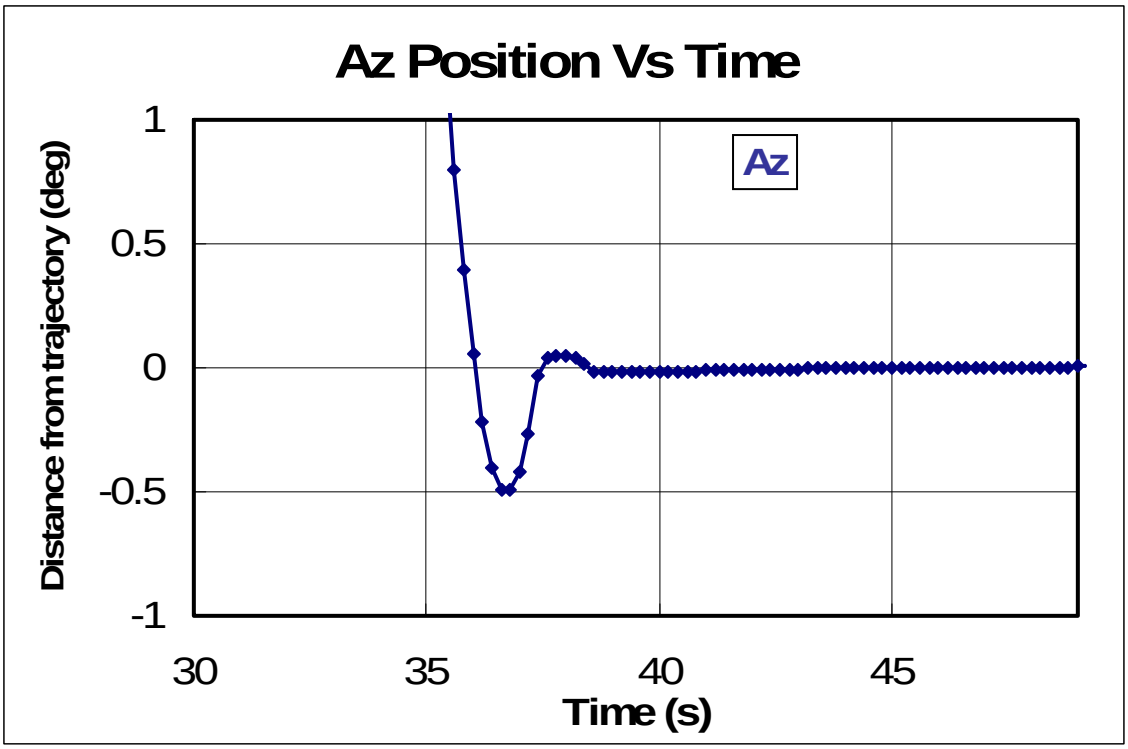
Real Antenna Response

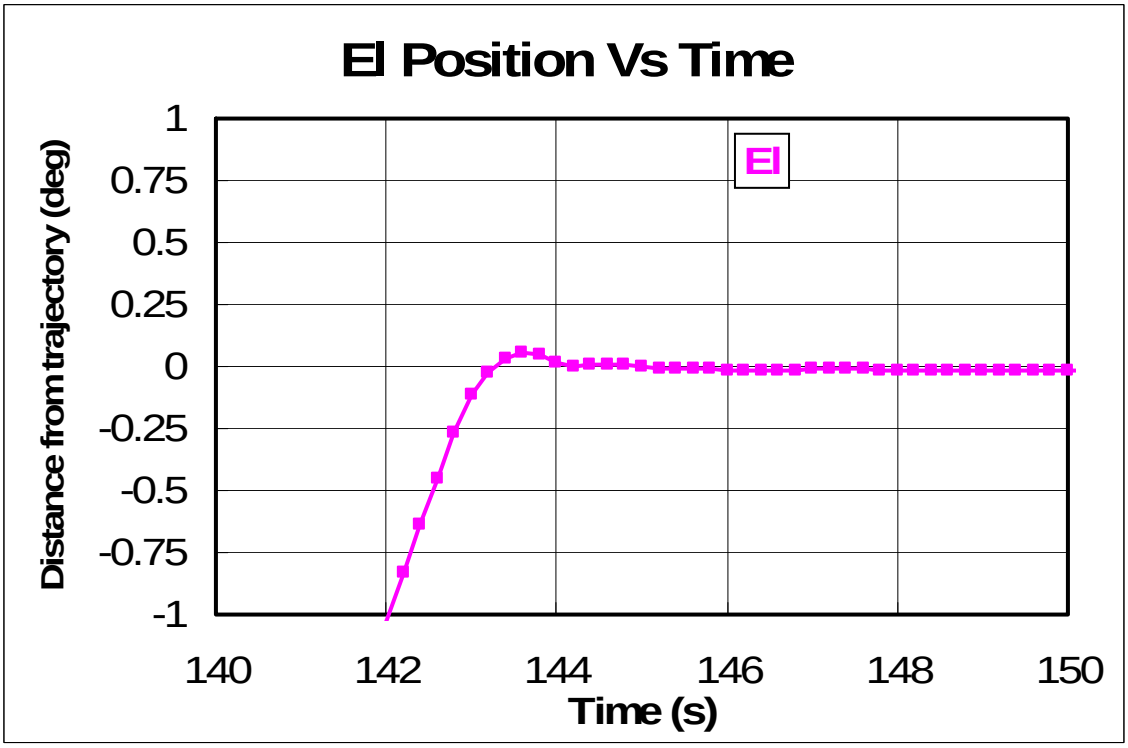
To optimize the pointing on each real antenna, we execute a regime of multiple pointings surrounding bright sources. The description of the pointing model and 10-point regime is a topic for another memo. Here we display some results from repeatedly cycling through a 10-point regime surrounding a GPS satellite.



Careful study of the graph above shows that there are 10 distinct pointing directions between 0-160 seconds of time, and then the cycle repeats. This graph demonstrates the rapidity with which our antenna goes from nearly zero velocity (tracking GPS) to maximum velocity (slew rate = 4°/sec on azimuth and 1°/sec on elevation).

To give an idea of the settling time, we blow up two regions of the above graph, when the azimuth or elevation are returning to the GPS satellite position after a large excursion. This is shown in the graphs below.





We observe that the azimuth and elevation positions both overshoot the correct position. The amount of overshoot is related to the maximum speed of the motor. The settling time is in the range of a few seconds, consistent with our predictions above.

Discussion and Conclusion

The informed reader might ask, “Where is the I?” The most common servo algorithm is PID – position, integral, differential. If you don’t feed back the integrated error, the servo may never converge on the true pointing direction. In our system the “I” term is hidden in the Copley servo amplifiers. Although we treat the amplifiers as if they were velocity drives, they are servo loops in their own right. When we issue a velocity command, fundamentally we are commanding the Copley servo with a filtered position. The Copley drive makes sure that we achieve the indicated position exactly, to within a fraction of an arc second. While we don’t specify the integral term directly, it is in there.

This raises another issue: There is another parameter space out there associated with the Copley amplifiers. When we issue a velocity command, the amplifiers respond with alacrity that subverts our attempt to limit 3 Hz oscillations of the antenna. Tuning the Copley amplifiers is unfinished business, and we believe further improvements in the antenna response will result when this tuning is undertaken.

Putting the Copley aside, the goal of this memo is to document the current implementation of the servo loop in the antenna control software. Further analysis and tuning is in order to optimize this servo loop to give the best response to pointing changes on the real antennas. For example, we anticipate that the settling time might be reduced even further. The tools to do this (i.e. to measure the pointing) are now in place, both by reading back the encoders and with an optical pointing system (described elsewhere by Rick Forster).