Digital Detection of Rocket Apogee

Collection Editor:
Edward Luckett

Authors:
Evan Dougal
Julia Kwok
Edward Luckett

Online:
< http://cnx.org/content/col11599/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: December 18, 2013
PDF generated: December 18, 2013
For copyright and attribution information for the modules contained in this collection, see p. 31.

Table of Contents

Chapter 1

Introduction[1]

1.1 Principles of Rocketry

Model rocket flight involves a cocktail of mathematics and physics. The very best model rockets are built on careful calculations from the principles of physics, aerodynamics, thermodynamics, and gas dynamics. For an optimal flight, the builder must consider all of these principles. Physics is used to predict the flight patterns and behavior of a rocket. Aerodynamics must be taken into account to ensure stable flight of the rocket, and to provide for the smoothest and highest flight possible. Finally, thermodynamics and gas dynamics are incorporated into the ignition and motor of the rocket, which actually propel the rocket. In our project, we incorporate the physics of the rocket in order to use digital signal processing to electronically detect rocket apogee. Apogee is the highest point in the rocket's flight and is the best time to deploy a parachute.

1.2 Rocket Build

As seen in Figure 1.1 (Rocket Build), model rockets consist of a number of parts. At the very tip of the rocket is the nosecone, the streamline shaped provides the lowest drag in the air (this is to make the rocket as aerodynamic as possible). The shock cord holds together the parts of the rocket after parachute ejection (the rocket physically splits in half and ejects the parachute, which allows the parts to float safely to the ground). The parachute is one of several recovery methods, most often used. For our purposes, it is both applicable to our larger rocket as well as cost efficient. The launch lug, guides the rocket along the launch rod – part of the tripod-like stand that the rocket is launched from – until the fins start working. The body tube is where devices, such as barometers, altimeters, accelerometers, and other electronics can be situated. Flame-proof recovery wadding is padded in between the area where the parachute and the motor are in order to protect the parachute and other devices from the motors hot ejection gases. The fins of the rocket, located externally on the body tube, provide aerodynamic stability. The motor, finally, accelerates the rocket upwards until it burns out.

[1] This content is available online at <http://cnx.org/content/m48335/1.1/>.

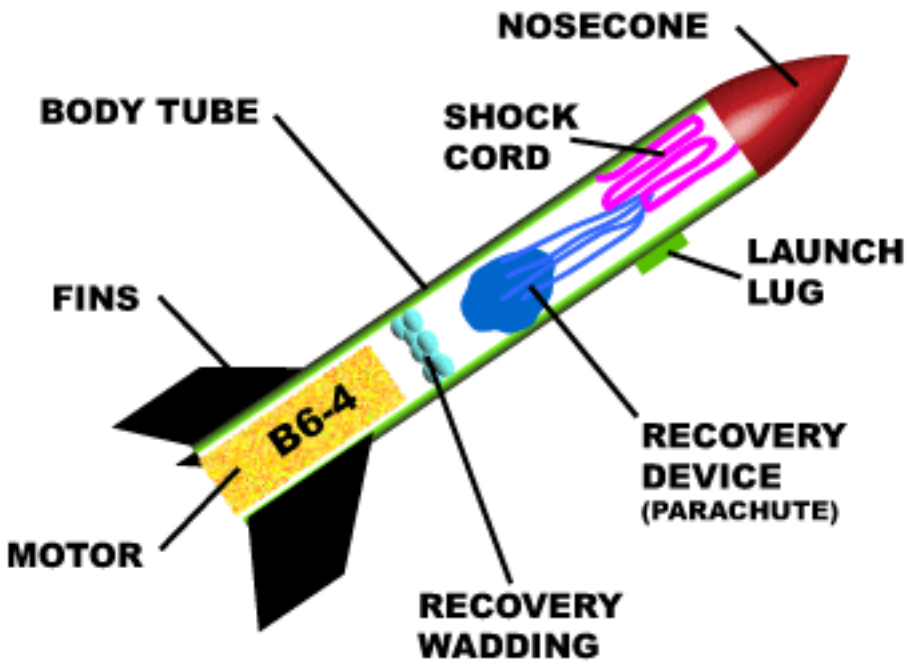

Figure 1.1: Rocket Model Diagram. From: http://www.manitobarocketry.org/AboutModelRocketry.html

Model rockets are usually categorized according to the motors they use. Motor classes are determined by the motor impulse, or the area under the thrust-time curve. The more powerful the motor, the further along the alphabet the class is. Motors start at 1/4A class and increase, to 1/2A, A, B, and so on. “Model Rockets” are officially classed as rockets with “G” motors or under. Rockets which use larger motors are typically called "high-powered rockets." (The rocket which this study centers around flew on an "I" motor). The rocket for this study incorporated an Arduino board paired with a barometer and accelerometer as its payload in order to take data for DSP processing.

1.3 Flight Path and Apogee

The flight path of a rocket has several main components. Firstly, we have ignition and liftoff, which are parts of the launch phase. The thrust phase, or the powered flight phase, is the phase in which the motor or engine is burning in order to push the rocket into the air. It is accelerating at this phase. Part three is the burnout, the exact moment in which the engine’s power has been completely expended and the rocket is no longer providing its own thrust. Right after the burnout, the model rocket ascends into the coasting phase, in which it starts slowing down from its top speed. Gravitational pull works against the rocket, and so it decelerates until it reaches Apogee. It in Figure 1.2 (Rocket Flight Path) as “Ejection,” because, ideally, this is where parachute ejection should occur. In model rockets, apogee is the highest point of a rocket’s flight. On a parabolic flight pattern, this is the point where the rocket reaches its maximum height, and where velocity equals zero. Finally, the rocket reaches its recovery phase, where the parachute provides air resistance to counteract the force of gravity pulling on the rocket. It then drifts to the ground, where the

model rocket can be recovered and recorded data logged.

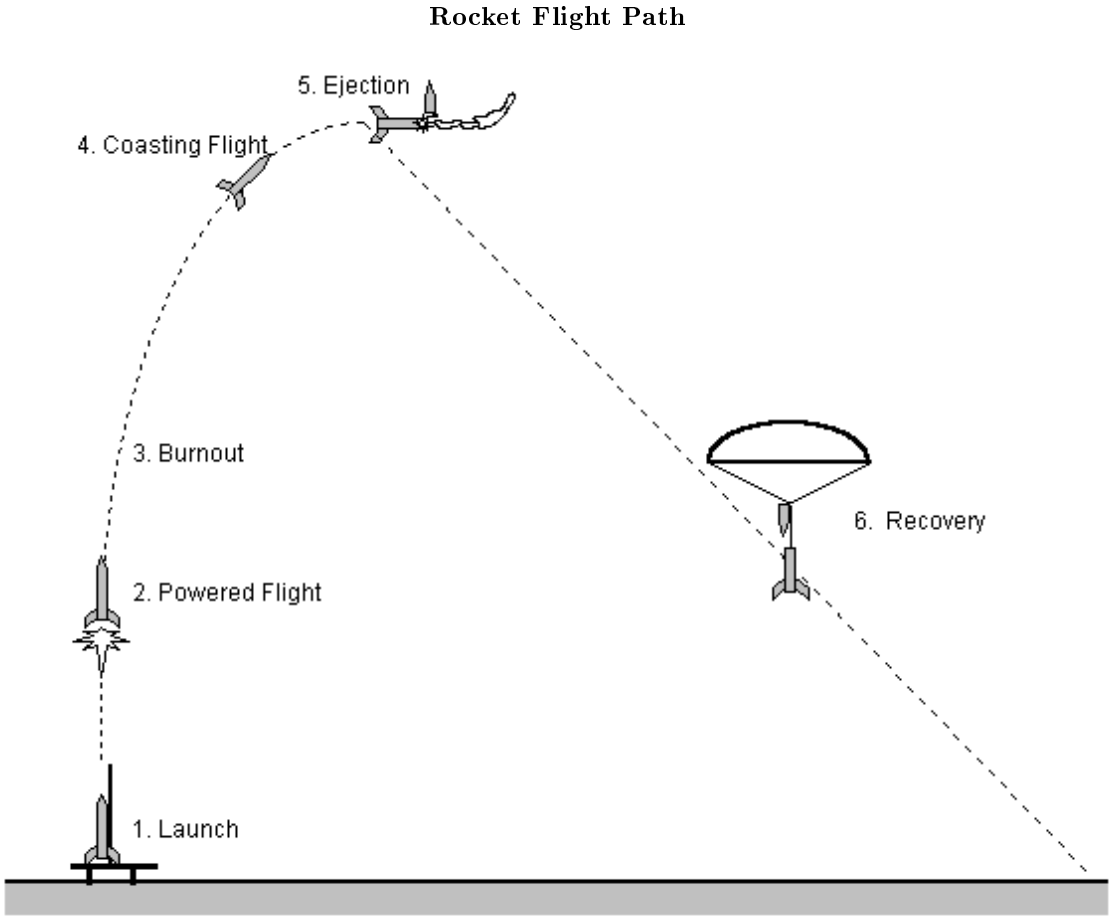

Figure 1.2: Model Rocket Flight Path Diagram. From: http://my.execpc.com/~culp/space/flight.html

One common reason for using electronic deployment is for so called "dual-deployment". To allow rockets to fly higher with less drift, rockets using "dual-deployment" deploy a small "drogue" parachute at apogee which only slows the rocket down a little bit. Closer to the ground the main parachute opens and slows the rocket down to a speed which is safe to land at.

Chapter 2

Background[1]

2.1 Apogee is Important

Why is apogee important in model rockets? Apogee is pivotal in the timing of the deployment of the parachute which allows the rocket to find its way to the ground safely, without damaging the rocket or the devices inside it. Timely parachute deployment from apogee ensures that the parachute will not tear mid-flight, so the rocket will be easier to track by eye on its way to the ground; this way fewer rockets will be lost on their way down from a flight. Late or early parachute deployment results in net forces acting against the parachute (in addition to the force of gravity it already has to act against) in flight, causing a larger strain on the parachute and rocket frame. In relation to the rocket apogee, a one second difference in the deployment of a parachute translates roughly to a change in 32 feet/second in velocity. While it is possible to create rockets and even possibly parachutes that can withstand theses forces, it is ideal to minimize these stresses wherever possible. These extra forces can cause rips and tears to less hardy parachutes or damage to the rocket frame, causing the rocket to plummet downwards rather than parachuting slowly down. Without optimal parachute deployment, the recovery stage of the rocket's flight will fail. Exploring rocket apogee detection methods, and in particular the use of the Kalman filter in rocket apogee detection, helps optimize performance while minimizing costs and disadvantages.

2.2 Current Mechanical Methods

Current methods used to deploy the parachute are completely mechanical and are not completely electronic. The most often used method as well as the most primitive method of deploying the parachute is to make use of a second ignition to push the components of the rocket out and towards the nosecone. The maker of the model rocket will usually drill a hole into a tube, and pack a certain amount of charge powder (the amount depending on the length and area of the hole). After the thrust and consequent burnout of the rocket engine, the last of the burnout sets a second ignition that blasts the charge up the body tube towards the nosecone. The force of the blast splits the rocket and sends the parachute components out into the air to catch wind resistance. The recovery wadding (Figure 1) must be placed in order to protect the parachute from the explosive charge. As seen in Figure 3, the whole system relies on the original ignition to relay through the electrical leads to the ignition charge.

[1] This content is available online at <http://cnx.org/content/m48330/1.1/>.

Components of Parachute Ejection System

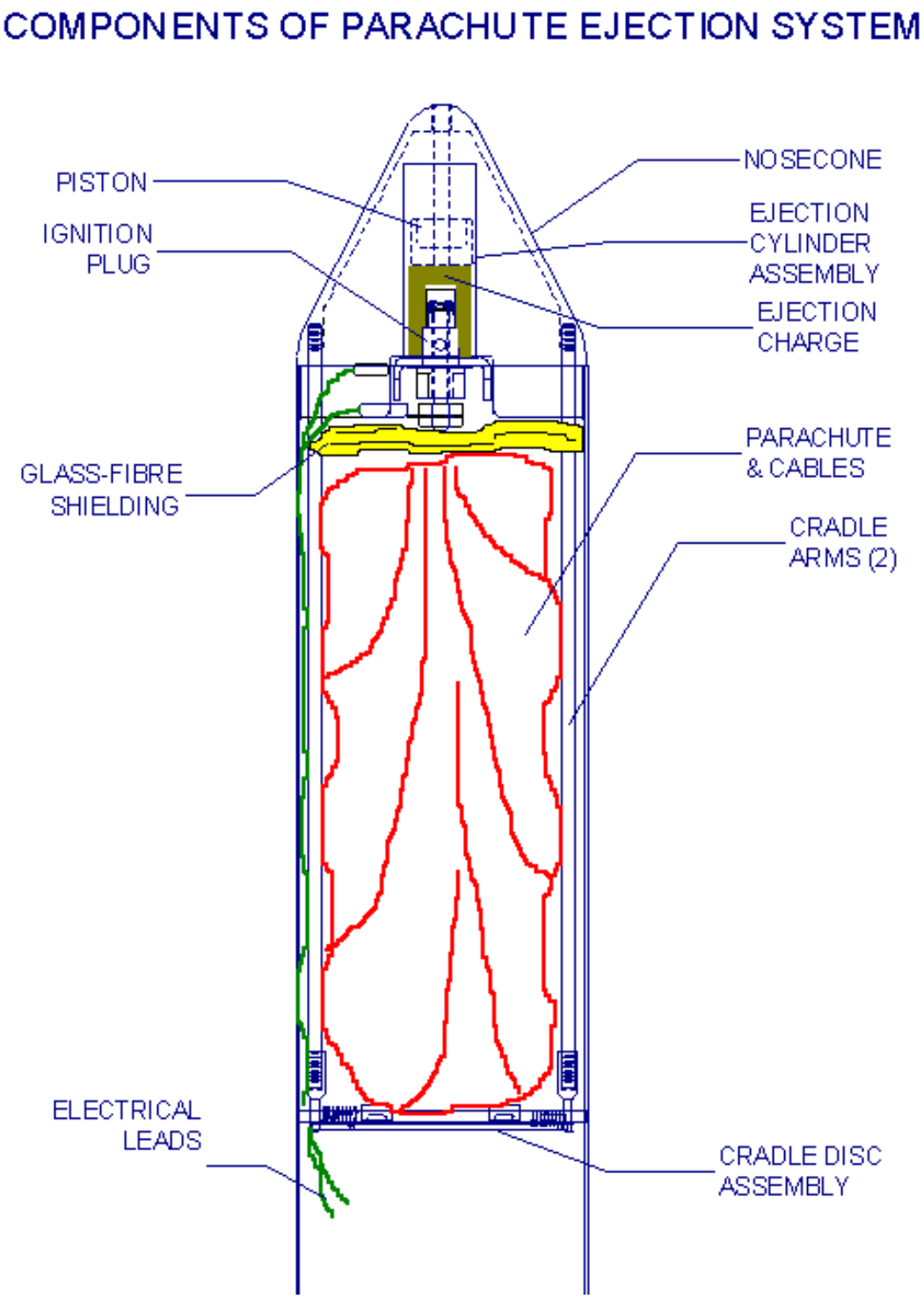

Figure 2.1: A diagram depicting an example of a non-electronic parachute deployment system. From: http://www.nakka-rocketry.net/parasys.html

This method, rather than rely on any sort of data, relies on the pre-computed estimates of apogee and when it happens. The length of the tube that sends the ignition charge can only be adjusted in steps of about two seconds. Those two seconds could end up being the difference between late deployment or early deployment, and on-time deployment. In addition to the large source of error introduced by the lack of precision of the apogee-predicting parachute fuse, it can be difficult to pre-calculate the timing of apogee. Without relying on any real-time rocket data, this non-electronic method of parachute deployment is can be unreliable. Accurate digital detection of apogee, is therefore key to an electronic method of parachute deployment.

2.3 Using the Accelerometer and Barometer

Sensors are used to estimate the rocket's state during flight. In order to take real time data, a microcontroller with both an accelerometer and a barometer are put inside the rocket payload section. A method for apogee detection using only an accelerometer (an "inertial system") is to detect apogee when the velocity, area under the acceleration curve, reaches zero. The acceleration of the rocket steadily increases due to the thrust, reaches a peak, and drops down to a constant -9.8 m/s2 after burn out (since the only force working on the rocket at that point is the downward force of gravity). Therefore, in a perfect world where the acceleration and the accelerometer match, the accelerometer would integrate in real time to find the velocity, and as soon as the velocity reached zero, it would deploy the parachute. However, accelerometers, in general, are far from perfect. The accelerometer data is quantized and noisy. More problematically, the accelerometer contains a bias. This makes integrating the data troublesome, as error from the bias increases constantly over time. This causes enough error to deploy the parachute seconds earlier or seconds later than apogee.

Another approach to electronic apogee detection comes from a barometer. The barometer measures air pressure and detects changes in air pressure in order to determine the height of the rocket. Apogee is detected when the air pressure stops decreasing, and begins to increase. With the barometer is that the data will be quantized. How quantized the data is depends on the sensitivity of the barometer, but are in general significant because the barometer detects changes in air pressure which are small for a comparatively large change in altitude. However, the presence of a bias term is irrelevant when computing apogee from a barometer because it is the change in altitude that determines apogee, not the absolute altitude. The barometer, in theory, can detect when apogee is by looking at the past terms and determining if the altitude is higher, lower, or about the same as the current height. Consider the half parabolic flight pattern in Figure 2. When the rocket is in thrust phase, the barometer will read that the current height is higher than the previous heights. Near the very end of the coasting, as the barometer starts to read values that are incredibly close to each other, the program may use a decision rule to deploy the parachute here. Alternatively, it can deploy as soon as it starts seeing a decrease in altitude. Of course, without digital filtering, apogee is impossible to detect; there is too much noise and quantization error. In addition, because barometers measure air pressure, the pressure wave produced when the rocket nears the speed of sound may be registered as a drop in altitude and therefore apogee. Barometers, like accelerometers are easily fooled. The noise caused in the data alone could cause the parachute to deploy anytime the data dipped below a past value, even if the model rocket is in thrust phase.

2.4 Conventional Filters are Inadequate

After taking in raw flight data, the next attempt to accurately extract the apogee is to filter it in order to obtain a signal that can be used to best approximate apogee. First is the naïve low pass filter approach. As seen in Figure 4, the boxcar filter was able to smooth out the noisy raw data to create a semi-parabolic flight pattern similar to the expected flight pattern. However, the fatal flaw of the boxcar filter is that, while it filters the signal reasonably, it creates a time delay. In this graph, the apogee occurs at around 10-11 seconds into the flight. However, the output to the boxcar filter puts apogee at around 11-12 seconds into the flight. Knowing how crucial a few seconds is in real time flight, the boxcar tragically falls short of the

expectations.

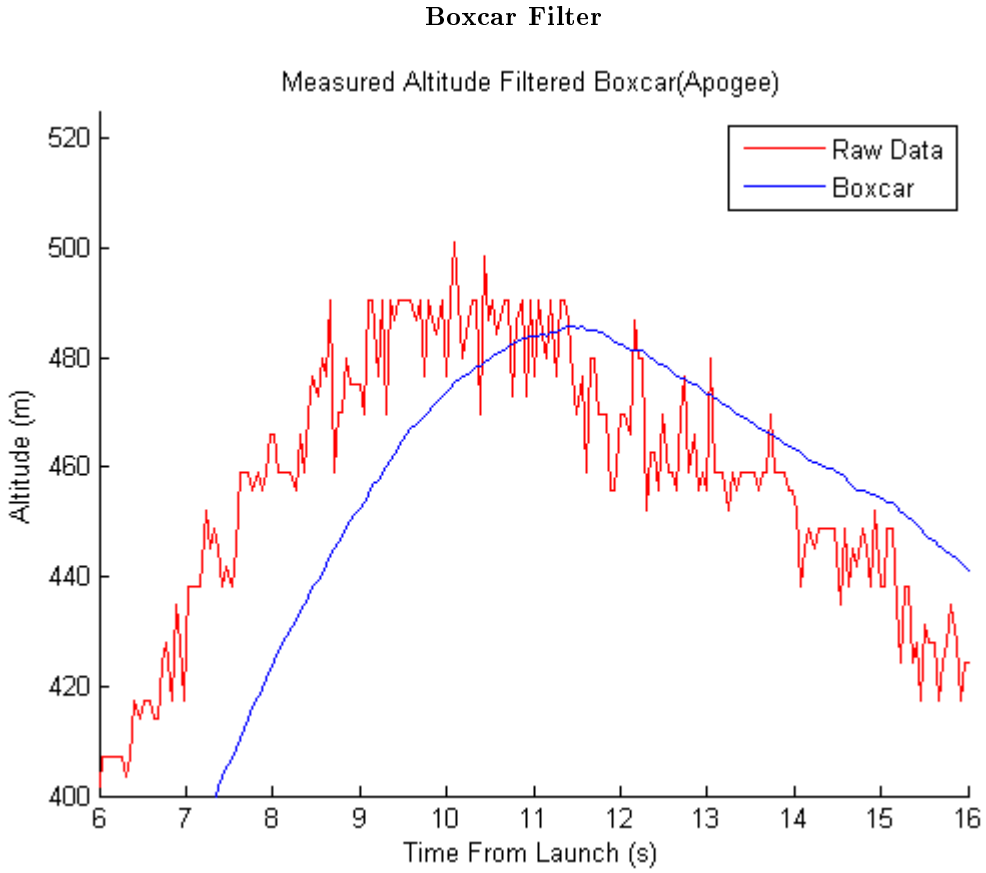

Figure 2.2: Graph of Raw Barometer Data vs. Filtered Data – Boxcar Filter

The first conclusion is that a simple boxcar filter has not worked for the purpose of accurately detecting the time of apogee. Perhaps other filters may work. Figure 5 and Figure 6 depict the same set of raw altitude data filtered by the butterworth and elliptical filters, respectively. On further inspection of the graph of the butterworth filtered data, it might be noted that the Butterworth filter has also created a time delay. Not only that, but this filter has created peaks and troughs (waves, ripples) in the filtered data. Although the time delay has been decreased slightly from the boxcar filter, it fails to smooth out the noisy signal. Unfortunately, if this filter were to be implemented, the system would detect apogee at the first decrease in altitude, which occurs around the 8.5 second mark.

Butterworth Filter

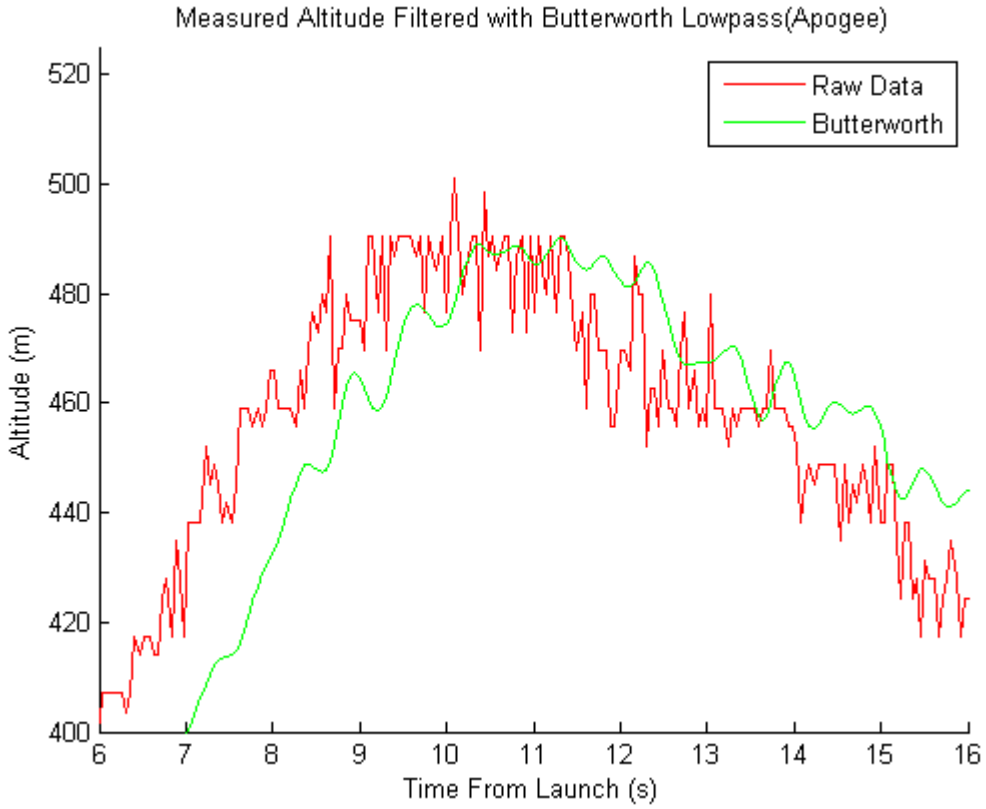

Figure 2.3: Graph of Raw Barometer Data vs. Filtered Data – Butterworth Filter

Unfortunately, the butterworth filter did not meet the criteria for accurate rocket apogee detection. One last conventional filter, the lowpass elliptic filter, might work. Figure 6 below is a graph of the raw and elliptically filtered altitude data. The elliptic lowpass filter, based on its rough outline, lines up well with the data. The global maximum of the elliptic filter does indeed seem to coincide with the apogee of the rocket (as extrapolated roughly by eye from the raw data). It seems to have overcome the time delay problem. However, the elliptic filter shares its flaw with the butterworth filter; it contains peaks and troughs that the system could easily confuse as apogee. The filtered data has very distinct local maxima and minima, resulting in the same problem as the butterworth filter; the system will deploy the parachute much too early, because it makes a decision rule at the very first decrease in altitude to release the parachute. In addition, the filtered data is attenuated, and while – in this case – bears no consequence to detecting the time of apogee, causes flaws in the actual altitude data.

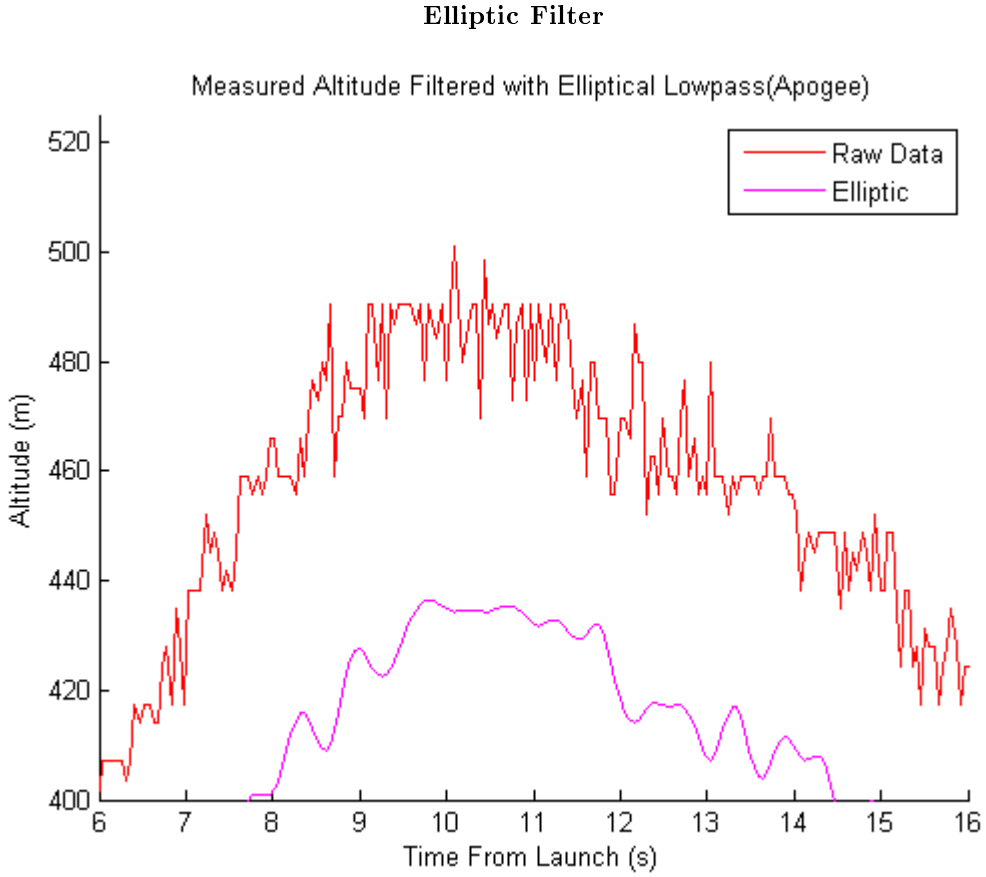

Figure 2.4: Graph of Raw Barometer Data vs. Filtered Data – Elliptic Filter

2.5 Why The Kalman Filter?

The Kalman filter is a filter that takes into consideration several factors that the other filters fail to. The Kalman filter is ideal when the situation involves a physical system along with real-time information gathering from noisy sensors. Model rocket flight can approximated from a fairly simple physical model which makes this problem well suited for a Kalman filter. While conventional filters do take into consideration past events in order to process current events, the Kalman filter utilizes the physical model of the rocket (which can be extrapolated from basic physics equations) as well as the previous state in order to process current events. One major advantage of the Kalman filter is that it only uses the previous state and the sensor data to predict the next state, unlike a running average which use many previous samples. Finally, the Kalman filter takes into account the reliability of the sensors (as determined by their variances and covariances) when making the state estimate, and is provably mathematically optimal when the model is accurate, and the sensor data is being corrupted by white noise.

Chapter 3

Implementation of the Kalman Filter[1]

Kalman Filter

The Kalman filter is a time domain method of incorporating knowledge of the physical model of the system and of the reliability of the sensors to accurately estimate the state of the system. Implementation of the Kalman filter first requires the creation of an accurate physical model of the system. The two equations which are used to determine the estimate of the current state from that of the previous state are:

$$x_k = \mathrm{A}x_{k-1} \tag{3.1}$$

$$x_k = x_k + K\left(m - Hx_k\right) \tag{3.2}$$

$$x = \begin{pmatrix} s \\ v \\ a \end{pmatrix} \tag{3.3}$$

Near apogee, the physical equations governing the rocket's flight are simple, which makes A simple. The only force acting on the rocket is gravity only (because drag forces vary with the square of the velocity they can be neglected near apogee, where the velocity is close to zero).

$$s = vt + \frac{1}{2}at^2 \tag{3.4}$$

$$v = at \tag{3.5}$$

$$a = -g \tag{3.6}$$

$$\begin{pmatrix} 1 & \Delta\mathrm{t} & \frac{\Delta\mathrm{t}^2}{2} \\ 0 & 1 & \Delta\mathrm{t} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Where Δt is the time between $\mathrm{x_k}$ and $\mathrm{x_{k+1}}$.

[1] This content is available online at <http://cnx.org/content/m48325/1.1/>.

m is a vector of the measured values from the sensors. Position is measured with the barometer and acceleration by the accelerometer.

$$\begin{pmatrix} s_m \\ a_m \end{pmatrix} \tag{3.8}$$

H is a matrix which maps x_k to m:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

Finally, K, the Kalman gain matrix, weights the difference between the measured values and the estimated values. K is typically computed in real time as the system changes. However, the formula for K is rather complicated and therefore difficult to implement on a microcontroller in real time. Luckily, because the rocket's flight can be approximated over the whole flight by the system and because the sensor variances do not change, K can be precomputed via the following recursive process:

$$K = PH^T\left(HPH^T + R\right)^{-1} \tag{3.10}$$

$$P = (I - KP)\,P \tag{3.11}$$

$$P = APA^T + Q \tag{3.12}$$

In a small number of repetitions, K will converge. In these equations, R is the measurement noise covariance matrix which holds the variances for each sensor:

$$\begin{pmatrix} {\sigma_p}^2 & 0 \\ 0 & {\sigma_a}^2 \end{pmatrix} \tag{3.13}$$

P is called the error covariance matrix, and it is first approximated with a guess, and then recursively defined like the K matrix. Finally, Q is the process noise covariance matrix, and is associated with the amount of noise added to the estimate in each time step. The code for calculating the K matrix is shown below:

```
    .% Calculates the Kalman gain

H = [1 0 0; 0 0 1];         % maps x (state variables) to z (sensor data)
R = [35.8229 0; 0 .0012];   % measurement noise covariance
Q = [0 0 0; 0 0 0; 0 0 1];  % process noise covariance matrix

T = .05;                    % time step

A = [1 T 1/2 * T^2; 0 1 T; 0 0 1]; % maps previous state to next state


% these three equations recursively define k (matrix of kalman gains)
% and P (error covariance matrix)

P = eye(3); % initial guess for p

for i = 1:20
```

```
    K = P*H'/(H*P*H' + R);   % Kalman gains
    P = (eye(3) - K *H)*P;
    P = A*P*A' + Q;
end

display(K)
display(H)
display(P)
```

The last piece of code demonstrates the actual implementation of the Kalman filter in Matlab.

```
    .% implements Kalman filter on altitude and accelerometer data. Required vectors are alt and accel,

t = .05:.05:15;

estimate = zeros(3,length(t));
estimate(:,1) = [alt(1); 0; accel(1)];
for i = 2:length(t)

        estimate(:,i) = A*estimate(:,i-1);
        estimate(:,i) = estimate(:,i) + K*([alt(i);accel(i)] - H *estimate(:,i));

end
```

Chapter 4

Rocket Flight Simulation[1]

4.1 Introduction

In order to ensure that the apogee detecting filter works, we needed to test the filter with flight data. While we were able to get altimeter and barometer data from one rocket launch, we quickly realized the infeasibility of launching a model rocket multiple times in order to gather data. Our solution to this problem was to use the measured data from the one rocket launch to create an accurate simulation in MATLAB that would save both time and money.

4.2 Simulation Synthesis

Because we knew the specifications of the rocket engine and the A/D quantization error of our sensors, we were able to easily create a simulation of rocket flight in MATLAB. The default measured values from the rocket data are as follows:

- The total length of rocket flight is 15s
- Acceleration of the rocket while in flight rises linearly from g = -9.81m/s^2 to a peak value 160 m/s^2 in the span of 0.05s
- This peak value is maintained for 0.35s
- Acceleration decreases linearly from 160 m/s^2 to g in 0.5s
- An acceleration of g is maintained for the rest of the rocket flight

Although the default parameters are listed above, the code below shows that parameters such as peak acceleration and length of rocket flight can be set externally. We used a parameterized model so that flights could be varied based on dynamics affecting flight, to avoid accidentally fitting overfitting our filter to the data.

```
    function rocket_sim(t_length,t_accmax,hold_a_max,t_g,a_peak)
close all; % Close existing figures
dt = 10^-3; % Time steps of 1ms

% Default parameters that match conditions of measured rocket data
t_length = 15;
t_accmax = 0.05;
```

[1] This content is available online at <http://cnx.org/content/m48321/1.1/>.

```
hold_a_max = 0.35;
t_g = 0.5;
a_peak = 160;

t = linspace(0,t_length,t_length*(1/dt)); % Create Time vector
g = -9.81; % Acceleration due to gravity
Acc_t = zeros(1,length(t)); % Initialize Acceleration vector
Acc_t(1:round(t_accmax*(dt^-1))) = ... % Linear rise to peak acceleration
    linspace(g,a_peak,round(t_accmax*(dt^-1)));

Acc_t(round(t_accmax*(dt^-1))+1:round(hold_a_max*(dt^-1))) = a_peak;
% Hold peak acceleration for set amount of time

Acc_t(round(hold_a_max*(dt^-1))+1:round(t_g*(dt^-1)))=...
    linspace(a_peak,g,round((t_g-hold_a_max)*(dt^-1))); % Linear decay to g

Acc_t(round(t_g*(dt^-1))+1:end) = g; % Hold acceleration of g for remainder
                                     % of flight
figure;
plot(t,Acc_t) % Show plot of simulated acceleration
title('Simulated Acceleration of Rocket')
```

Once an acceleration vector is created, a state transition matrix is created in order to derive velocity and position states and graphs are created. Using the position graph, we can determine the true time of apogee and use this time as a baseline for the effectiveness of various filters (see Figure 4.1, Figure 4.2, Figure 4.3).

```
    state_vecs = zeros(3,length(t)); % Initialize matrix for position,
                                     % velocity and acceleration states
trans_mat = [1 dt (1/2)*(dt^2); 0 1 dt; 0 0 1]; % Initialize state
                                                % derivation matrix
prv_states = zeros(3,length(t)); % Matrix for previous states
prv_states(3,:) = Acc_t; % Set acceleration vector in previous states
for i=1:length(t)-1
    state_vecs(1,i) = trans_mat(1,:)*prv_states(:,i); % Derive position
    state_vecs(2,i) = trans_mat(2,:)*prv_states(:,i); % Derive velocity
    state_vecs(3,i) = trans_mat(3,:)*prv_states(:,i); % Derive acceleration
    prv_states(1,i+1) = state_vecs(1,i); % Use current states to derive
                                         % future values
    prv_states(2,i+1) = state_vecs(2,i);
end

s_comp = state_vecs(1,:); % Position vector
v_comp = state_vecs(2,:); % Velocity vector
a_comp = state_vecs(3,:); % Acceleration vector
figure;
plot(t,s_comp) % Show plot of computed position
title('Computed Position')
figure;
```

```
plot(t,v_comp) % Show plot of computed velocity
title('Computed Velocity')
figure;
plot(t,a_comp) % Show plot of computed acceleration
title('Computed Acceleration')
[apogee,t_index] = max(s_comp); % Determine and display actual apogee
disp('Actual Apogee occurs at t =')
disp(t(t_index))
```

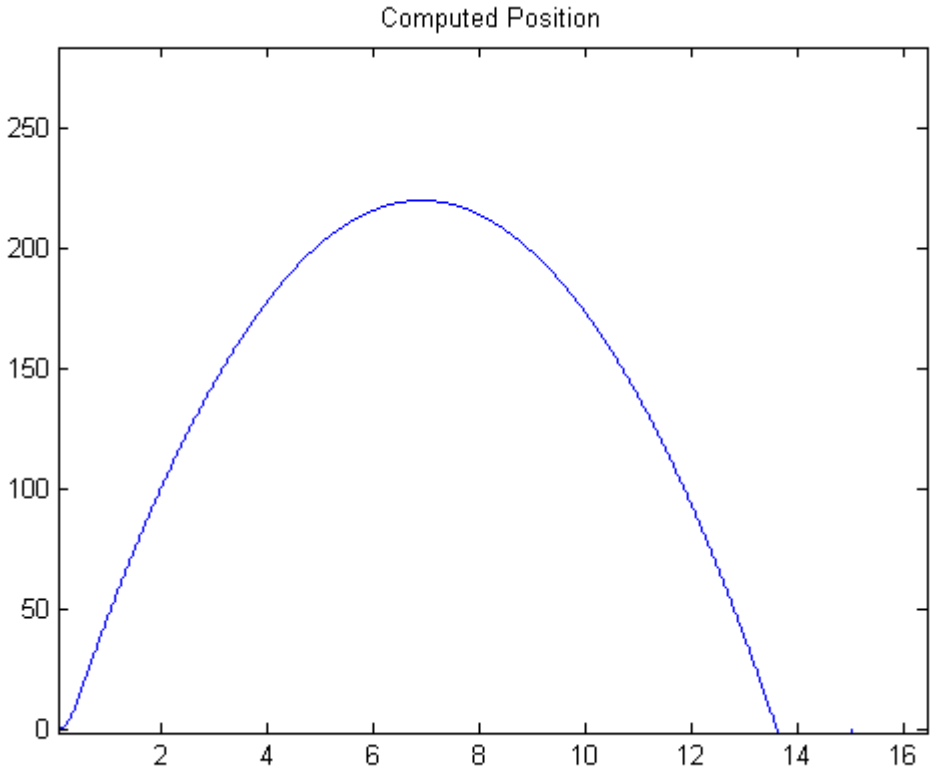

Figure 4.1

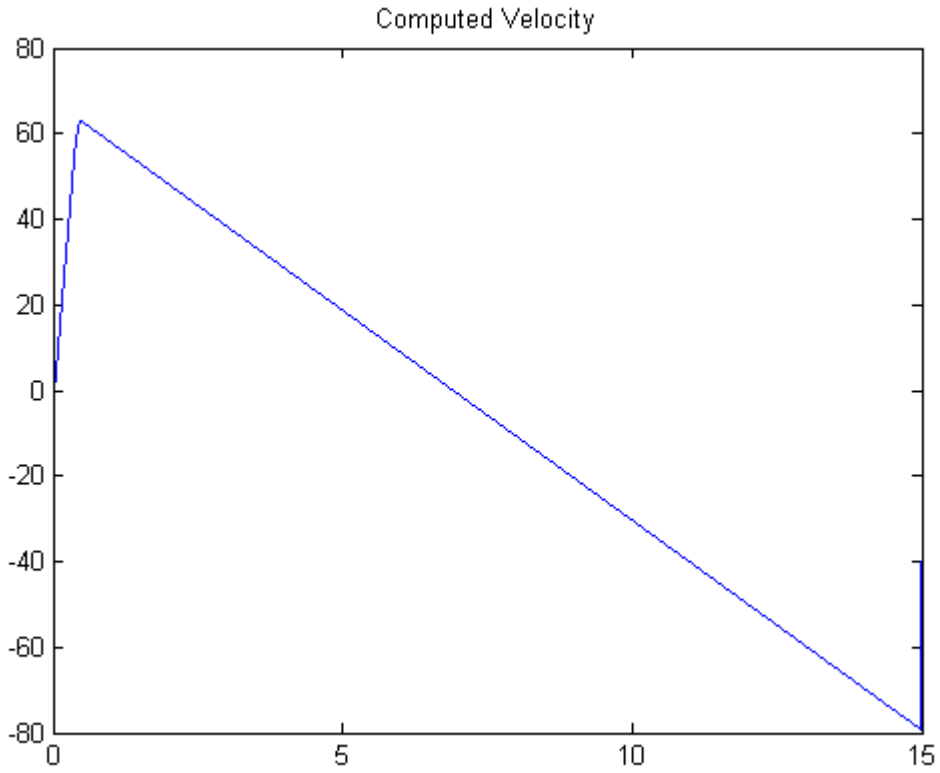

Figure 4.2

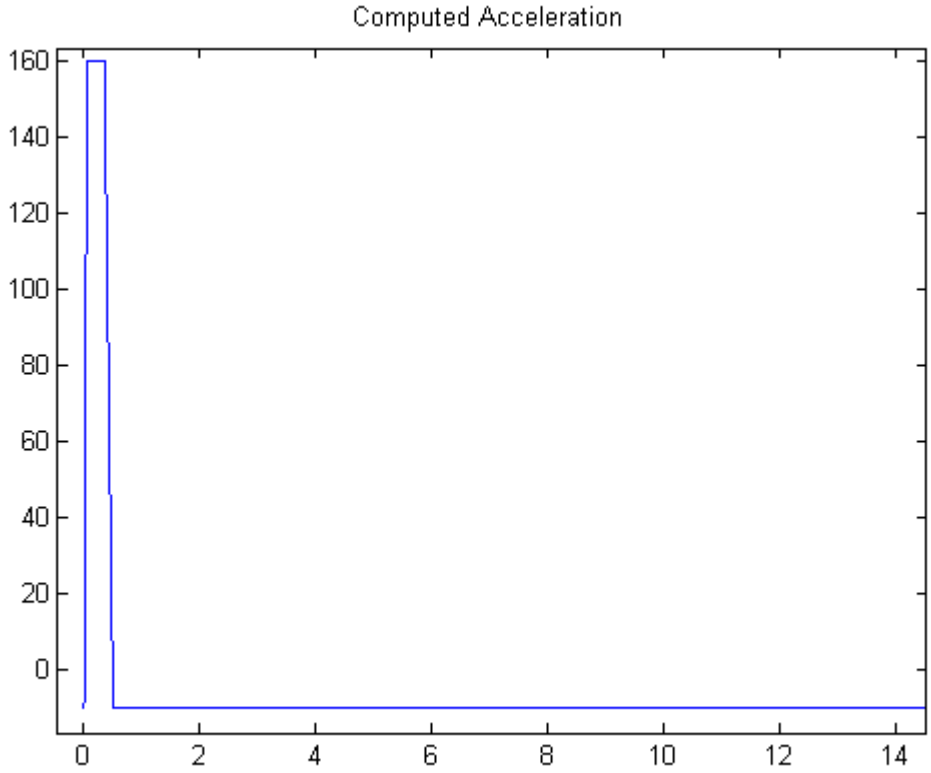

Figure 4.3

Using the var() command in MATLAB, we determined that the measured data from the rocket launch got corrupted with Gaussian noise with mean=0 and different variances for position, velocity, and acceleration.

State	**Variance**
Position	5.985
Velocity	2
Acceleration	0.0346

Table 4.1

In addition, we introduced A/D quantization from our sensors into the position, velocity, and acceleration states. Specifically, the sensors introduced a time quantization of 50ms and an amplitude quantization in increments of three (see Figure 4.4, Figure 4.5 Figure 4.6):

```
% Measurement noise
```

```
% Position corrupted with Gaussian noise with standard deviation = 5.985
s_n = state_vecs(1,:)+(5.985.*randn(1,length(t)));
% Velocity corrupted with Gaussian noise with standard deviation = 2
v_n = state_vecs(2,:)+(2.*randn(1,length(t)));
% Position corrupted with Gaussian noise with standard deviation = 0.0346
a_n = state_vecs(3,:)+(0.0346.*randn(1,length(t)));
figure;
plot(t,s_n) % Plot noisy position
title('Noisy Position')
figure;
plot(t,v_n) % Plot noisy velocity
title('Noisy Velocity')
figure;
plot(t,a_n) % Plot noisy acceleration
title('Noisy Acceleration')
%Quantization
%Time Quantized to 0.05 seconds
%Amplitude quantization is 3 meters
t_q = t(1:50:end);
s_n_q = floor(s_n(1,1:50:end)./3).*3;
v_n_q = floor(v_n(1,1:50:end)./3).*3;
a_n_q = floor(a_n(1,1:50:end)./3).*3;
figure;
plot(t_q,s_n_q) % Plot noisy and quantized position
title('Noisy and Quantized Position')
figure;
plot(t_q,v_n_q) % Plot noisy and quantized velocity
title('Noisy and Quantized Velocity')
figure;
plot(t_q,a_n_q) % Plot noisy and quantized acceleration
title('Noisy and Quantized Acceleration')
```

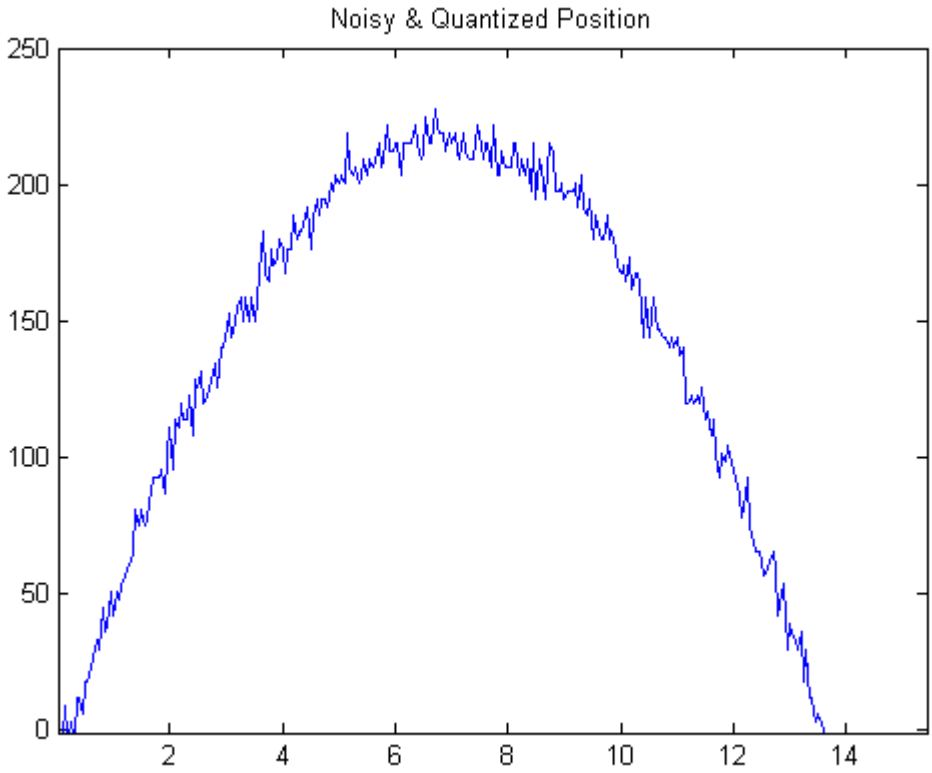

Figure 4.4

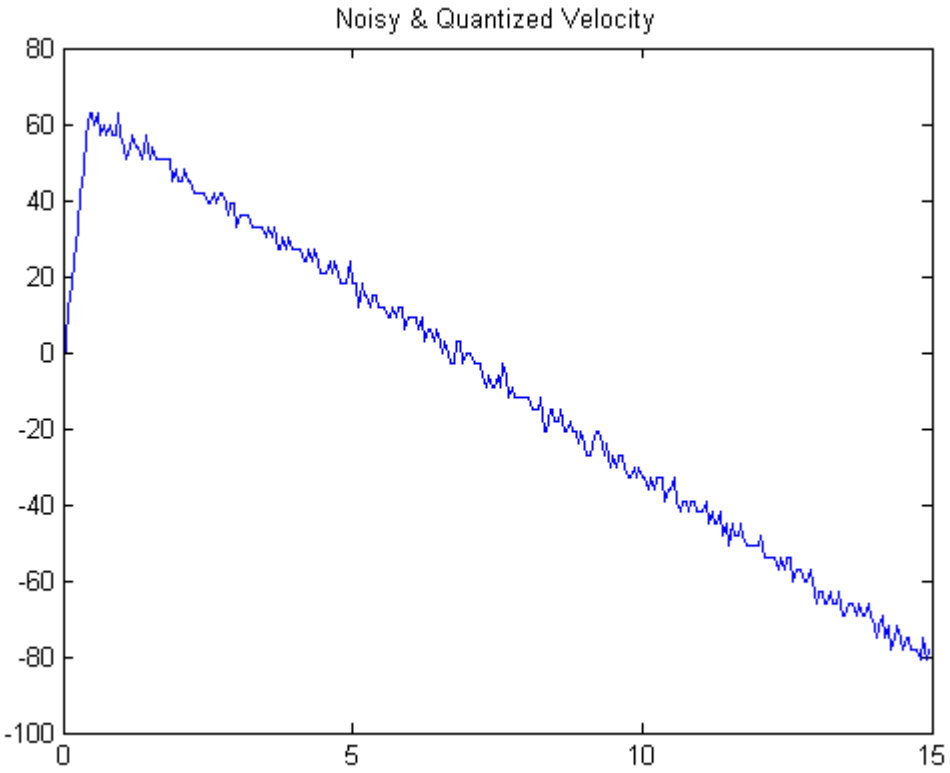

Figure 4.5

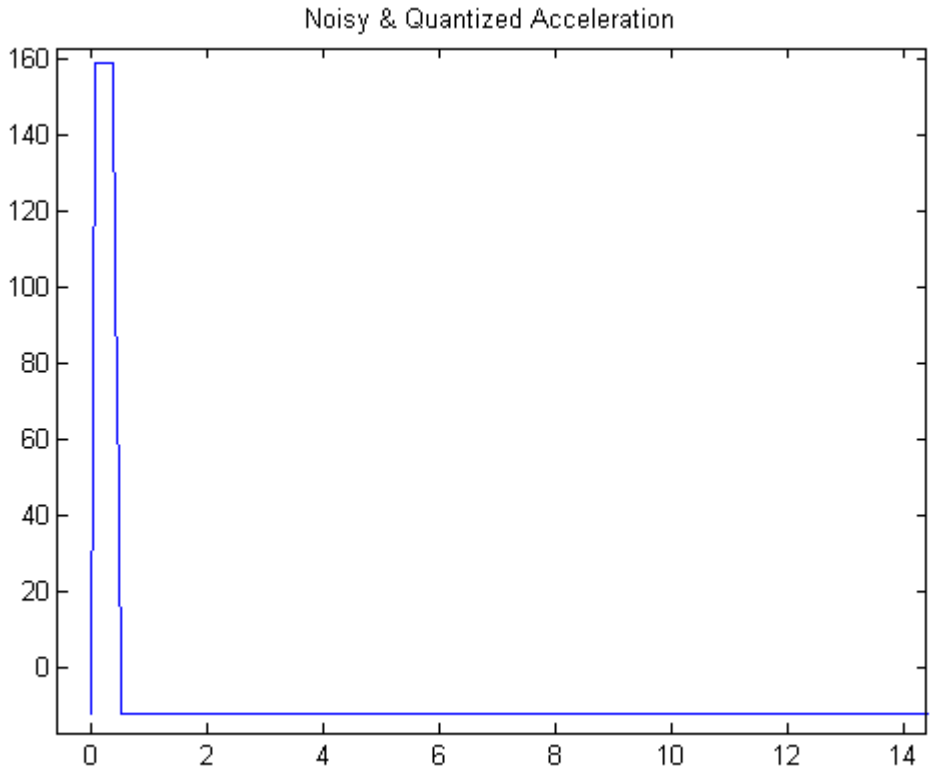

Figure 4.6

4.3 Conclusion

Once corrupted with noise and quantized, the simulation provides an accurate approximation of our measured data, and allows for multiple simulated, random, and parameterized rocket tests (see Figure 4.7):

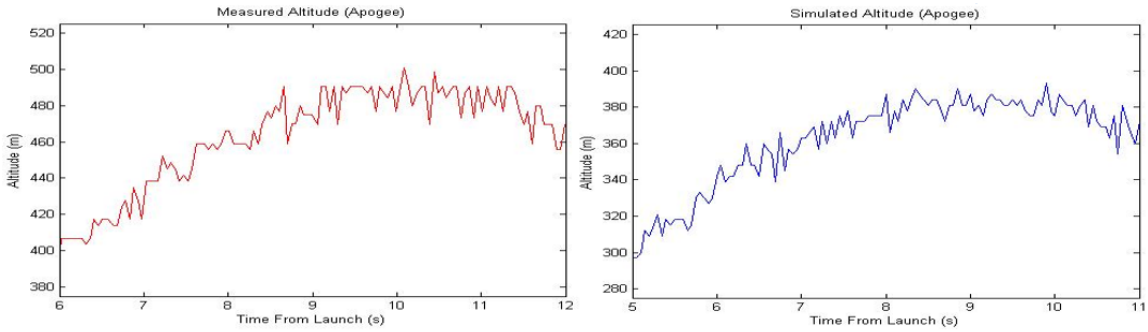
Measured Altitude (Apogee)
Altitude (m)
520
500
480
460
440
420
400
380
6
7
8
9
10
11
12
Time From Launch (s)
Simulated Altitude (Apogee)
Altitude (m)
420
400
380
360
340
320
300
280
5
6
7
8
9
10
11
Time From Launch (s)

Figure 4.7

Chapter 5

Results[1]

5.1 Accurate Detection of Apogee

Because the Kalman filter takes the physical model of a noisy system into account, it was more effective at predicting states of a system than other commonly used filters. While other filters rely only on current and past states, the Kalman filter's ability to estimate future states mitigate the delay and error which make it superior to other common solutions. In particular, with respect to our rocket flight, we compared the effectiveness of the Kalman filter with the other filter that successfully smoothed the signal, the Boxcar filter. As Figure 5.1 shows, the Kalman filter has significantly less delay, almost matching the actual apogee line in shape, than the filtered output of a Boxcar filter.

The Kalman filter does a remarkable job of removing noise from the position signal, but in order to actually determine apogee a decision rule is needed. Originally, the plan was to determine that the rocket was at apogee when its velocity was equal to zero. However, despite the fact that the accelerometer had very low variance compared to the barometer (which weighted it heavily in the Kalman gain), it had a significant bias term in it, which made the velocity term reach zero about a second before actual apogee. A more effective decision rule was to look for the position to have negative slope for three consecutive samples. Thus, apogee was defined as the time when the , position term was less than the previous position term, which in turn was less than the position term before that.

While the output of the Kalman filter has a negative amplitude bias compared when compared to actual apogee, it still accurately predicted the time of apogee, which is key to parachute deployment and minimizing forces on the rocket airframe and parachute.

[1] This content is available online at <http://cnx.org/content/m48307/1.1/>.

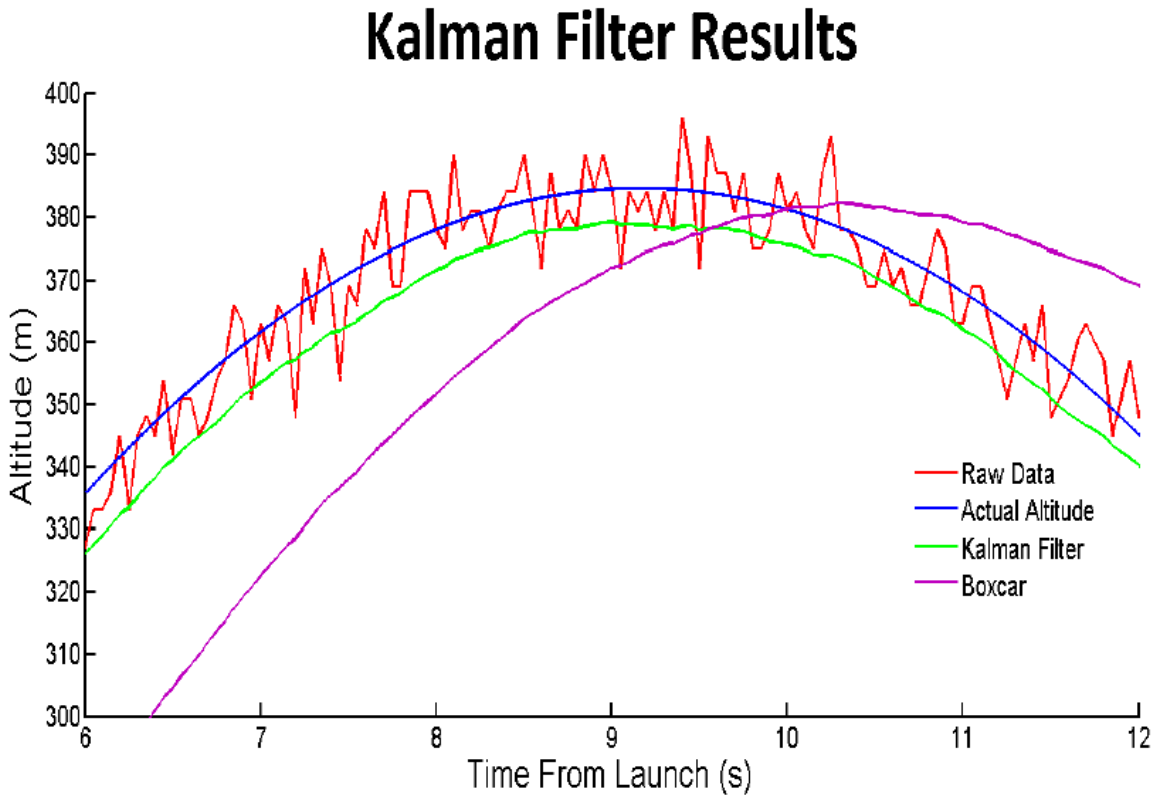

Figure 5.1

Figure 5.2 makes the effectiveness of the Kalman filter even clearer. Across 200 flights with randomized parameters, the Kalman filter displayed a mean error in detecting apogee of 0.269 seconds while the Boxcar filter has a mean error of 1.52 seconds. In terms of velocity, the Kalman filter, on average, mitigates the velocity experienced by the rocket to 2.64 m/s, as opposed to the 14.9 m/s when a Boxcar filter is used. Effectively, the Kalman filter is about six times more accurate at determining apogee.

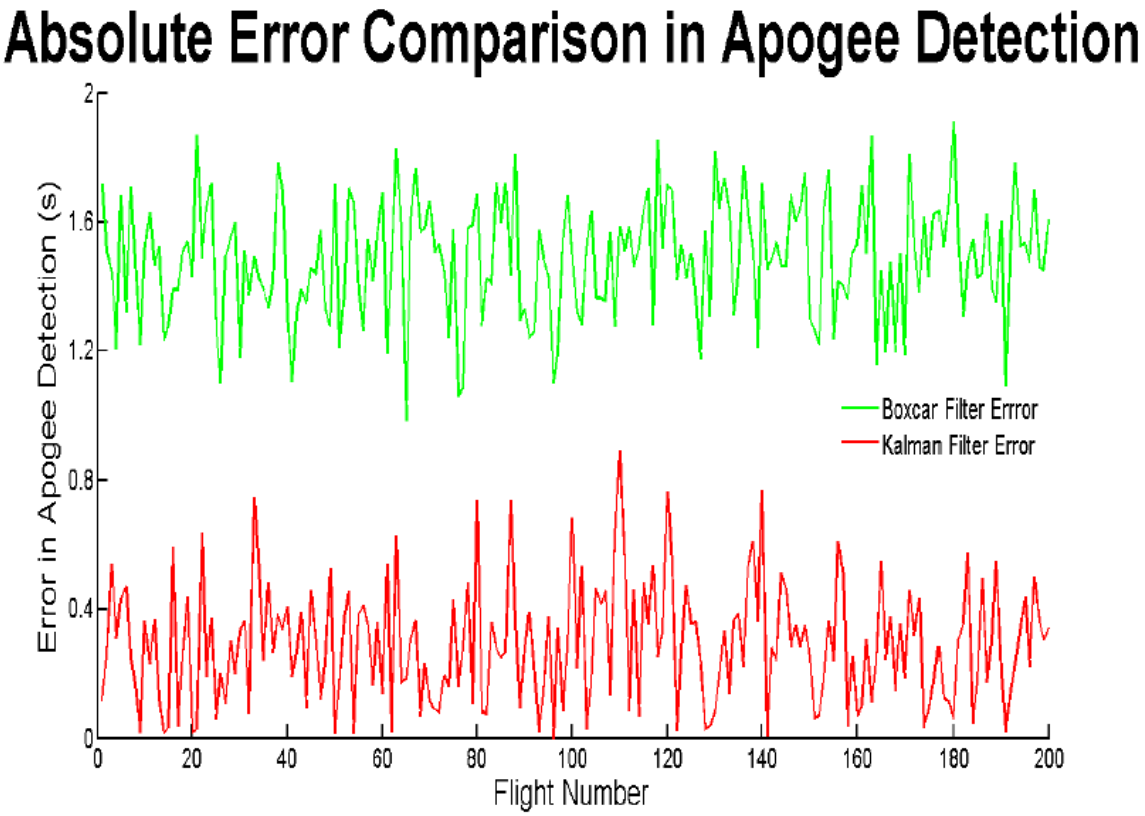
Absolute Error Comparison in Apogee Detection
Error in Apogee Detection (s)
Flight Number
Boxcar Filter Errror
Kalman Filter Error
0
0.4
0.8
1.2
1.6
2
0
20
40
60
80
100
120
140
160
180
200

Figure 5.2

Chapter 6

References[1]

References

1: Welch, Greg, and Gary Bishop. An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, 2006. .
2: Schultz, David. Barometric Apogee Detection Using the Kalman Filter. NAR Research and Development Report, 2002. .
3: Schultz, David. Application of the Kalman Filter to Rocket Apogee Detection. NAR Research and Development Report, 2004 .
4: "Beginner's Guide to Rockets." Beginner's Guide to Rockets. <https://www.grc.nasa.gov/www/k-12/rocket/bgmr.html>. .
5: Milligan, Tim. "Phases of a Rocket's Flight." Apogee Rockets.<http://www.apogeerockets.com/Tech/Phases_of_a_R
.
6: "How Does the Parachute in a Model Rocket Work?" Curious About Astronomy:. <http://curious.astro.cornell.edu/question.php?number=479> .
7: Nakka, Richard. "Parachute Recovery System." Richard Nakka's Experimental Rocketry Site. <http://www.nakka-rocketry.net/parasys.html>. .
8: Culp, Randy. "The Flight of the Model Rocket." <http://my.execpc.com/~culp/space/flight.html>.

[1] This content is available online at <http://cnx.org/content/m48306/1.1/>.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *Digital Detection of Rocket Apogee*
Edited by: Edward Luckett
URL: http://cnx.org/content/col11599/1.1/
License: http://creativecommons.org/licenses/by/3.0/

Module: "Introduction"
By: Julia Kwok
URL: http://cnx.org/content/m48335/1.1/
Pages: 1-3
Copyright: Julia Kwok
License: http://creativecommons.org/licenses/by/3.0/

Module: "Background"
By: Julia Kwok
URL: http://cnx.org/content/m48330/1.1/
Pages: 5-10
Copyright: Julia Kwok
License: http://creativecommons.org/licenses/by/3.0/

Module: "Implementation of the Kalman Filter"
By: Evan Dougal
URL: http://cnx.org/content/m48325/1.1/
Pages: 11-13
Copyright: Evan Dougal
License: http://creativecommons.org/licenses/by/3.0/

Module: "Rocket Flight Simulation"
By: Edward Luckett
URL: http://cnx.org/content/m48321/1.1/
Pages: 15-24
Copyright: Edward Luckett
License: http://creativecommons.org/licenses/by/3.0/

Module: "Results"
By: Edward Luckett
URL: http://cnx.org/content/m48307/1.1/
Pages: 25-27
Copyright: Edward Luckett
License: http://creativecommons.org/licenses/by/3.0/

Module: "References"
By: Evan Dougal
URL: http://cnx.org/content/m48306/1.1/
Page: 29
Copyright: Evan Dougal
License: http://creativecommons.org/licenses/by/3.0/

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.