Nonlinear Programming Approach to Filter Tuning

Dylan M. Asmar and Greg J. Eslinger

May 15, 2012

1 Introduction

State estimation is a problem that is often encountered across multiple disciplines. Filters are used to obtain state estimations based on measurements when one or both of the dynamics and measurements are inaccurate or subject to noise. Various filter methods can be applied to obtain accurate state estimations. Such filters include the Kalman filter [1], particle filter, [2], unscented Kalman filter and extended Kalman filter [3], and a constant rate filter based on the Fokker-Planck equation [4] just to name a few. However, despite a large variety in filters, most require tuning after implementation.

Filter tuning is the process of selecting measurement and dynamic noise statistics to obtain a desired performance. Even optimal filters like the Kalman filter are sensitive to tuning parameters [5]. Often, filter tuning is done via trial and error. This process may provide insight for the engineering, but often consumes a large amount of time [6].

A plethora of research has been devoted to develop algorithms to optimally and/or automatically tune filters, especially the Kalman filter. These methods include a neural network based approach [7], using the downhill simplex algorithm [6], reinforcement learning [8], and many more [9, 10]. However, gradient based methods have largely been avoided. The difficulty of analytically calculating the gradient, approximating the cost function with a well-behaved function, and the instability of numerical differentiation of noisy functions have all contributed to this avoidance [7, 6].

This paper focuses on applying gradient-based methods to tune filters which estimate the vertical state of aircraft based only on quantized altitude measurements. Different state estimation techniques have been developed for quantized measurements; however, only the Kalman filter and a modified Kalman filter [11] will be discussed.

The rest of this paper is organized as follows. Section 2 gives an overview of the aircraft tracking problem and the model used. Section 3 outlines the general mathematical program definition and then formulates a formal representation of the tracking problem. Section 4 discribes the various approaches to gradient based optimization. Section 5 shows the results of the different methods as well as a derivation of an improved search algorithm.

2 Problem Overview

The problem under consideration is finding the best filter parameters for vertical state estimation of aircraft with quantized altitude measurements. Tracking such aircraft is an applicable real-world problem. The Traffic Alert and Collision Avoidance System (TCAS) is mandated on all large aircraft worldwide. TCAS serves as a last minute defense against midair collisions and issues commands to change altitude when needed in the form of resolution advisories. The majority of the advisories in the current U.S. airspace are with intruders that broadcast their altitude with 100 ft quantization. Last minute collision avoidance systems rely on precise altitude rate estimations and accurately estimating the vertical rate from such quantized measurements is difficult [12].

2.1 Model

An aircraft's vertical state is defined in terms of altitude, h, and vertical rate, $\dot{h}$. This is formally defined as

$$x_k = \begin{bmatrix} h_k \\ \dot{h}_k \end{bmatrix} \tag{1}$$

The state estimation is performed based on observations of

$$z_k = \Delta \text{round}\left(\frac{\begin{bmatrix} 1 & 0 \end{bmatrix} x_k + w'_k}{\Delta} \right) \tag{2}$$

where k is a discrete-time index, Δ is the size of the quantization level, and w'_k is zero-mean Gaussian measurement noise with covariance R'.

The aircraft vertical motion can be modeled with a discrete-time linear dynamic system as:

$$x_{k+1} = F x_k + G v_k \tag{3}$$

where x_k is the state vector previously defined, v_k is zero-mean Gaussian process noise with covariance Q, and

$$F = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{4}$$

$$G = \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix} \tag{5}$$

where dt is the change in time from $k-1$ to k. In this formulation, the tracked aircraft is assumed to maintain constant velocity and maneuvers are modeled as accelerations due to process noise. The measurements z_k are modeled by

$$z_k = H x_k + w_k \tag{6}$$

where w_k is zero-mean Gaussian observation noise with covariance R, and

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{7}$$

2.2 Filtering

Modeling the vertical tracking problem in the format described in Section 2.1, allows for different filters to be applied. A Kalman filter and a modified Kalman filter [11] were chosen. The Kalman filter is the best linear estimator and the modified Kalman filter provides a similar nonlinear counterpart to analyze. The details of the filtering algorithms are beyond the scope of this paper, but are provided in Appendix A for reference.

3 Program Formulation

The desired performance of a filter dictates how the parameters should be selected. As described in Section 2, the vertical estimation problem is crucial to last minute air traffic collision avoidance systems since they rely on accurate vertical rate estimates. This naturally leads to a desire to minimize the error in the estimated vertical rate. The ideal situation would allow for the minimization of the error during estimation process. However, there is no access to the true data during estimation. Therefore, the parameters have to tuned a priori.

State estimation problems often have historical data on the system for which the filters were designed. Using this assumption of attainable data, the filter can be tuned offline to optimize performance over a subset of data that is a good representation of the actual system. Applying this idea in a general sense, the cost function can be defined in terms of the root mean squared error (RMSE) as:

$$J\left(p_1, p_2, \cdots, p_l\right) = \sum_{i=1}^{n} \sum_{j=1}^{s} c_j \mathrm{RMSE}_j^i \tag{8}$$

where J is the cost a function of l parameters, p, c_j is the weight of the RMSE of state j, RMSE_j^i is the RMSE of state j of item i of the subset, n is the number of items in the subset, and s is the number of state variables. For the Kalman filter and modified Kalman filter, the only change in results would be subject to the process noise covariance, Q, and the observation noise covariance, R. Therefore the formal definition of our problem in the general sense would be:

$$\min_{Q,R} J(Q,R) = \sum_{i=1}^{n} \sum_{j=1}^{s} c_j \mathrm{RMSE}_j^i$$
$$\text{s.t. } Q, R \succeq 0$$

Therefore, the program formulation for the vertical tracking problem becomes:

$$\min_{Q,R} J(Q,R) = \sum_{i=1}^{n} \left(c_j h_{\mathrm{RMSE}}^i + c_{\dot{h}} \dot{h}_{\mathrm{RMSE}}^i \right)$$

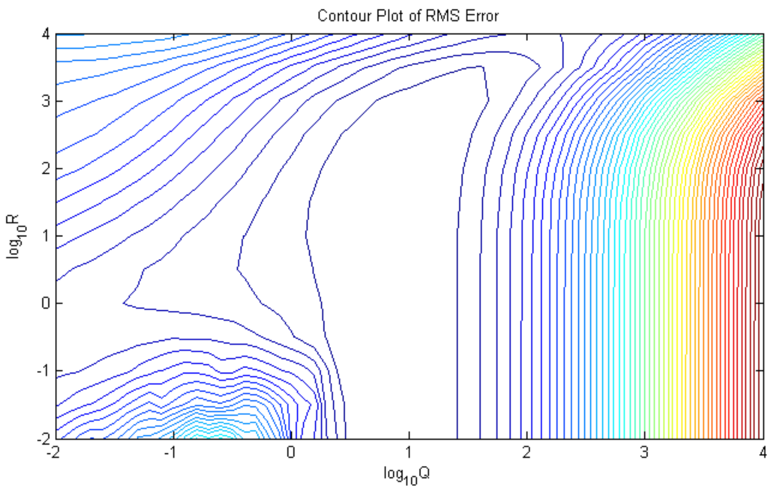

Figure 1: Contour Plot of an Example Data Set

$$\text{s.t. } Q, R \geq 0$$

$c_{\dot{h}}$ are predefined weights, h^i_{RMSE} is the altitude RMSE for track i, $\dot{h}^i_{\text{RMSE}}$ is the vertical rate RMSE for track i, and n is the number of tracks in the set for which the optimization is occurring.

The particular problem posed is advantageous from an intuitive design perspective in that any optimization will occur by changing two scalar values. This allows contour and mesh plots to be used to aid the Kalman filter and modified Kalman filter tuning. In alternative problems with higher dimensions this would not be possible; however, this problem will allow for a demonstration of the performance of the search algorithms through the aid of plots. Fig. 1 shows an example of what a contour plot might look like for this problem, while Fig. 2 is a three dimensional mesh of the same example case.

Notice in the contour and mesh plot that there did not appear to be a unique global minimum; rather, there existed a trough of Q and R values that produced similar costs for all Q and R in the trough. Upon closer inspection the trough did contain a global optima.

4 Methods

This section discusses various gradient-based methods used to solve unconstrained nonlinear optimization problems.

4.1 Gradient Derivation

Since the cost function was not explicit, there were two approaches considered for determining the gradient: adjoint methods and finite difference equations.

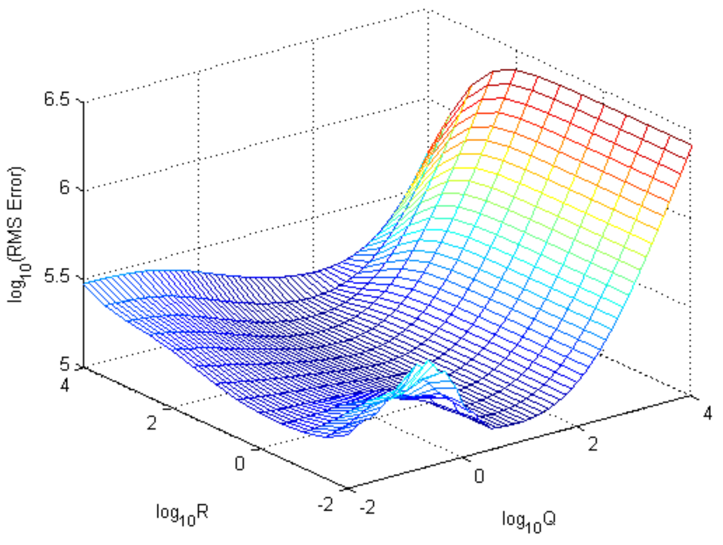

Figure 2: 3 Dimensional Mesh of an Example Data Set

4.1.1 Adjoint Methods

Adjoint methods find the derivative of a function imbeded in a program by tracking the derivitives of all variables in the program. The method is based on the fact that all programs are comprised of a set of equations. Therefore, any program can be written as

$$y = F_n(F_{n-1}(...F_2(F_1(x))...)) \tag{9}$$

where $F_i(\bullet)$ is the ith line of code in the program [13]. Using this logic, the derivitive of y with respect to x is found using the chain rule:

$$\frac{\partial y}{\partial x} = \frac{\partial F_n}{\partial z_{n-1}} \frac{\partial F_{n-1}}{\partial z_{n-2}} ... \frac{\partial F_2}{\partial z_1} \frac{\partial F_1}{\partial x} \tag{10}$$

where z_i is the vector of all internal variables at line i of the code. The adjoint can be calculated in the forward or backward direction; however, if $n = \dim(x)$ and $m = \dim(z)$, calculating the adjoint in the forward direction requires tracking an $m \times n$ matrix while calculating the adjoint in the reverse direction requires tracking a $1 \times m$ matrix.

Adjoint methods will produce the analytic gradient of the function with respect to the input variables; however, adjoint methods require three to four lines of code for every original line of code executed in the forward direction. Adjoint methods also require every value for every variable be saved for the reverse direction, which can fill up computer memory quickly.

4.1.2 Finite-Difference Equations

Finite-difference equations approximate the derivative of the function by perturbing the function by a small amount and tracking the change in the objective function. Assuming the function is differentiable, the gradient is approximated by finding the slope of the linearization at the perturbation point using the perturbation data. Because the finite-difference equations only approximate the gradient, the gradient produced by the finite-difference equations will not be as accurate as those produced by adjoint methods. However, the computational overhead of the error calculation was very small as compared to the full adjoint method. For this reason finite-difference equations were chosen to approximate the gradient.

4.2 Algorithms

Since the Hessian of the objective function is not accessable, the available optimization algorithms are:

- Steepest Descent
- Quasi-Newton Methods
- First Order Methods

These algorithms will be discussed in detail in Section 4.2.1, Section 4.2.2, and Section 4.2.3, respectivly.

4.2.1 Steepest Descent Methods

The Steepest Descent Algorithm, as presented by Prof Robert Freund, is seen in Algorithm 1:

Algorithm 1 Steepest Descent

```
Given $x^0$
$k \leftarrow 0$
repeat
    $d^k := -\nabla f(x^k)$
    $\alpha = \arg\min_\alpha f(x^k + \alpha d^k)$
    $x^{k+1} \leftarrow x^k + \alpha d^k$
    $k \leftarrow k + 1$
until $d^k = 0$
```

The steepest descent method has the advantage of being one of the simplist optimization methods. It is also requires little overhead, as the only information that is saved from an interation is the current value of x^k. However, the steepest descent method does use a line search, which can be costly depending on the function. The steepest descent method can also break down when the probrlem is ill-conditioned, that is, when the search path falls into a valley that is steep in one direction but shallow in another.

4.2.2 Quasi-Newton Methods

Quasi-Newton Methods attempt to build up an estimate of the Hessian as the search progresses by using information on the function value and the gradient. The earliest Quasi-Newton Method is the Davidon, Fletcher, Powell, or DFP, method. The most popular method nowadays is the Broyden, Fletcher, Goldfarb, Shanno, or BFGS, method [13]. The BFGS Algorithm, as presented by Prof Steven Hall, is seen in Algorithm 2:

Algorithm 2 Quasi-Newton (BFGS)

```
Given $x^0$
$B^0 := I$
$k \leftarrow 0$
**repeat**
    $d^k := -B_k \nabla f(x^k)$
    $\alpha = \arg\min_\alpha f(x^k + \alpha d^k)$
    $x^{k+1} \leftarrow x^k + \alpha d^k$
    $s_k = \alpha d_k = x^{k+1} - x^k$
    $y_k = \nabla f(x^{k+1}) - f(x^k)$
    $B^{k+1} \leftarrow \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right)^T B^k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}$
    $k \leftarrow k + 1$
**until** $d^k = 0$
```

The advantage of Quasi-Newton methods is that they do not struggle with ill-conditioned problems as much as steepest descent methods do. The downside is that additional calculations are required at each iteration. In addition, additional information, in the form of B^{k+1}, must be stored at every iteration.

4.2.3 First Order Methods

The Simple Gradient Scheme of the First Order Method, as presented by Prof Robert Freund, is seen in Algorithm 3:

Algorithm 3 First Order

```
Given $x^0$, $L$
$k \leftarrow 0$
**repeat**
    $d^k := -\nabla f(x^k)$
    $x^{k+1} \leftarrow x^k - \frac{1}{L} \nabla f(x^k)$
    $k \leftarrow k + 1$
**until** $d^k = 0$
```

This method is very similar to the steepest descent method in that the descent direction is simply the negative of the gradient of the function evaluated at x^k. However, one of the major assumptions of the first order method is that $f(\bullet)$ has a *Lipschitz gradient.* That is, there is a scaler L for which

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\| \;\forall\; x, y \in \mathbb{R}^n \tag{11}$$

Since L is unknown for this application, L will be initialized at $L = 1$. At every iteration, if Eq. (12) holds, then the current value of L is sufficent for the search algorithm to converge.

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2 \tag{12}$$

Should the current iteration fail this test, the algorithm is set to double the current guess of L and recompute the step. Therefore, the true algorithm that will be implimented for this project is seen in Algorithm 4

Algorithm 4 First Order (Modified)

```
Given $x^0$
$L = L^0$                                   ▷ The user must provide a guess for $L^0$
$k \leftarrow 0$
repeat
    $d^k := -\nabla f(x^k)$
    $x^{k+1} \leftarrow x^k - \frac{1}{L}\nabla f(x^k)$
    if $f(x^{k+1}) > f(x^k) + \nabla f(x^k)^T(x^{k+1} - x^{k+1}) + \frac{L}{2}\|y - x\|^2$ then
        $L = 2L$
        $x^{k+1} \leftarrow x^k$            ▷ This reruns the iteration with the new value for $L$
    end if
    $k \leftarrow k + 1$
until $d^k = 0$
```

5 Results

This section discusses the results of each algorithm discussed in Section 4. Multiple data sets were used to simulate various true data sets.

5.1 Data Set Generation/Evaluation

Three types of data sets were used when testing the algorithms.

- Synthetic tracks from an assumed model
- Synthetic tracks generated form a high fidelity encounter model
- Actual radar data of aircraft collision avoidance advisory events

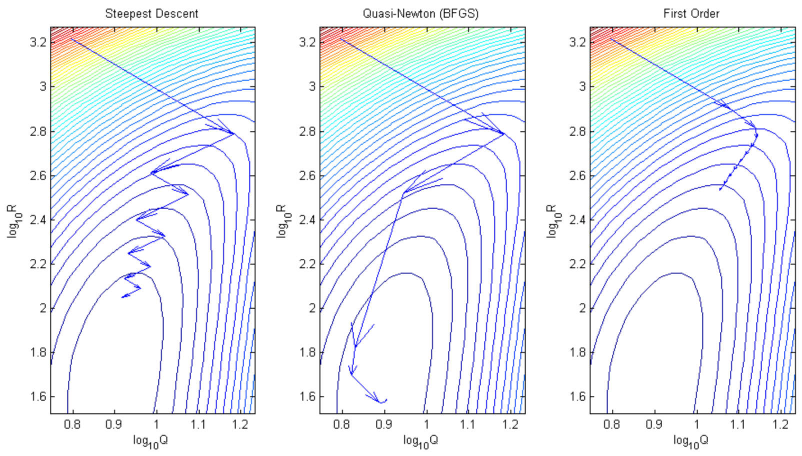

Figure 3: Convergence Paths of Different Search Algorithms

The first set of data was used as a starting point and a control. Tracks were generated from the dynamic model given by Eqs. (2) to (5). The altitude was quantized at 100 ft levels after the observation noise was added to generated the measurements. The process and observation noise covariances were varied to generate various data sets. The starting altitude was sampled from a normal distribution with a mean of 15000 ft and a standard deviation of 5000 ft. The initial altitude rate was sample from a normal distribution with a mean of 0 ft/s and a standard deviation of 50 ft/s.

The second type of data set were generated from a high fidelity encounter model [14]. The tracks are generated from a correlated model based on recorded advisory events in the U.S. airspace. The third type of data set was actual radar data of encounters with aircraft of 100 ft quantization levels.

Each scenario resulted in the convergence of at least one algorithm to an optimum. The parameter settings found using subsets of true data (the radar data) were applied to the larger set. In each case, the mean RMSE over all tracks was lower than multiple other settings tested. Including settings from an experienced engineering that had been working with the Kalman filter and modified Kalman filter for this problem.

5.2 Algorithm Performance

The three described algorithms were run on a several different sets of data. The results for all three methods from a radar data set are shown in Fig. 3. The purpose of this section is not to present the result of one data set; rather, it is to compare and contrast the performance of the different search algorithms on this particular type of optimization.

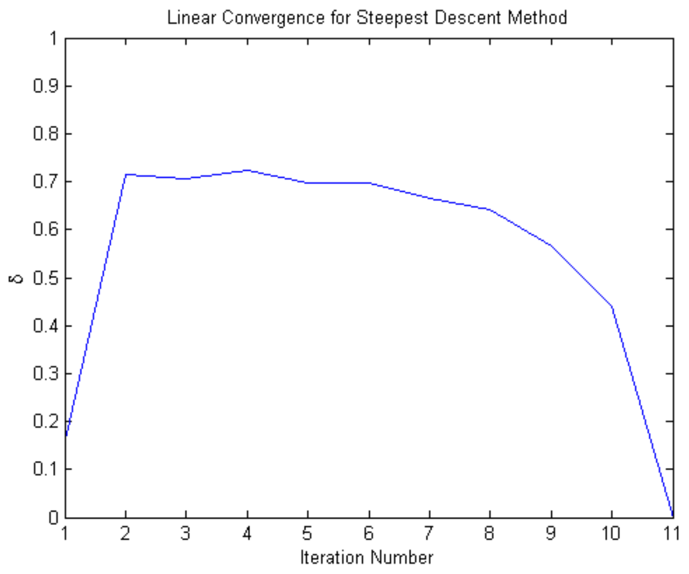

Figure 4: Linear Convergence of Steepest Descent Algorithm

5.3 Steepest Descent Analysis

The steepest descent algorithm was able to converge in a region that effectively minimized the objective function. Fig. 3 also shows that hem-stitching occured during the search. The steepest descent mehtod also converged linearly. An algorithm is said to converge linearly if there is a constant $\delta < 1$ such that for all k sufficently large, the iterates x^k satisfy:

$$\frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} \leq \delta \tag{13}$$

where x^* is some optimal value of the problem[15]. The plot of δ at each iteration can be seen in Fig. 4. In this particular example the algorithm did not run for enough interations to make a definitve statement on whether the convergence was linear. There appears to be a linear section between Iteration 2 and 8 as well as a superlinear section between Iteration 8 and 12; however, this is not enough iterations to clearly prove or disprove linear convergence.

It is also interesting to note that $\nabla g(x^{k+1}) \bullet \nabla g(x^k) \neq 0$. When an exact line search is used and the exact gradient is provided, every iteration travels perpedicular to the previous iteration. Since the gradient of the function was only a numerical approximation and the line search used was not an exact line search, the steps in the steepest descent algoirthm were not perfectly perpedicular.

5.4 Quasi-Newton Methods

The Quasi-Newton Method converged in the fewest iterations as compared to the other two methods. The team also checked to see if this algorithm demonstrated quadratic convergence. An algorithm is said to converge quadtratically if there exists some $\delta < \infty$ where:

$$\lim_{k \to \infty} \frac{f(x^{k+1}) - f(x^*)}{\left(f(x^k) - f(x^*)\right)^2} \leq \delta \tag{14}$$

Although Newton and Quasi-Newton methods often times converge quadratically, this algorithm did not. This is partially because the Hessian was approximated using numerical approximations of the gradients. This introduces error into the Hessian, which was already an approximation. Nevertheless, it is significant to notice that the BFGS converged quickly on a solution.

5.5 First Order Methods

Out of all three algorithms, the First Order Method converged on the greatest function value. Fig. 3 clearly shows that the First Order Method could have continued to decrease the function value. The algorithm's termination was caused by the decreasing gradient. Since L is fixed, as the magnitude of the gradient decreases, so does the step size. As the step sizes decrease, the changes between function values decrease. Eventually, the changes in the function value become so small the algorithm deems the path converged. Steepest Descent Methods do not have this issue because the step size is variable.

On the other hand, the First Order Method is the most stable method. The First Order Method has guaranteed convergece in a convex region. This is evidinced by the fact that the convergence path of the First Order Method was heading directly towards the minimum. Steepest Descent algorithms, although typically robust, cannot fully boast guaranteed convergence when using an in-exact line search. Since Newton and Quasi-Newton methods also use line searches, the same can be said about these methods.

Although there are too few data points to make a definitive asserstion, Fig. 5 suggests that the First Order algorithm has a near linear convergence. It is also interesting that the First Order path seemed to find and follow an eigenvector of the system. This is significant because line search algorithms perform very well when aligned with the eigenvector. This lead to the development of the Augmented First Order Method

5.6 Augmented First Order Method

In order to create a more efficent algorithm for solving the posed problem, a hybrid algorithm was developed that combined First Order Methods with Steepest Descent Methods. Since the First Order Method was effective at finding an eigenvector, the algorithm starts the same way as the First Order Method. When the First Order Method has converged on a trough, the algorithm switchs

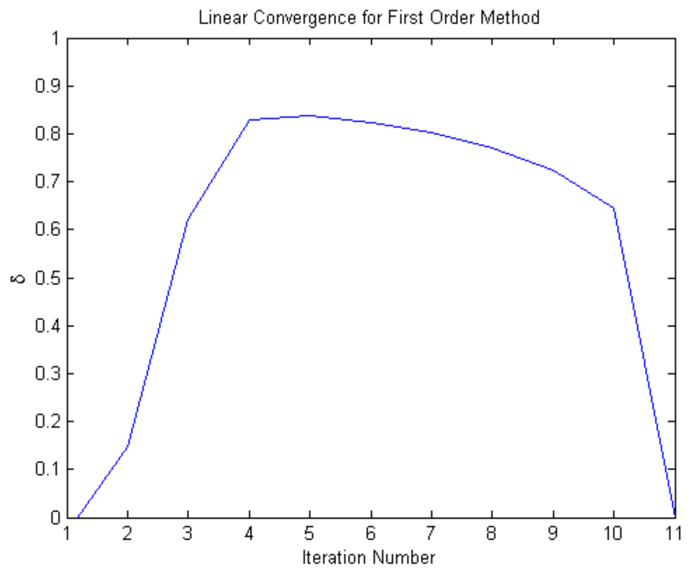

Figure 5: Linear Convergence of First Order Method

to a line search, which rapidly follows the trough down towards the minimum. The swtiching criteria for the algorithm was derived from the dot product:

$$\cos\theta = \frac{u \bullet v}{\|u\|\|v\|} \tag{15}$$

The dot product is used in the algorithm to measure the angle between the last step and the current descent direction. When Eq. (16) holds true, the descent direction is effectively lined up with the previous direction of travel, indicating a line search would be appropriate:

$$d^k \bullet (x^k - x^{k-1}) > c\|d^k\|\|x^k - x^{k-1}\| \tag{16}$$

where $0 < c < 1$. The value of c determines the range of θ that allows the algorithm to use a line search instead of the First Order Method. In practice, $c = 0.97$ produced satisfactory results. This value of c makes the allowable angle deviation approximitely 15 degrees. A comparison of the Augmented First Order Algorithm, the Steepest Descent Algorithm, and the First Order Method are seen in Fig. 6. After multiple tests, the Augmented First Order Method typically converged on a solution in 30% less iterations than the steepest descent or first order method. The aglorithm can be seen in Algorithm 5.

6 Remarks

- The cases analyzed in this paper were simple cases of a much more general problem set. However, the results suggest that gradient-based methods

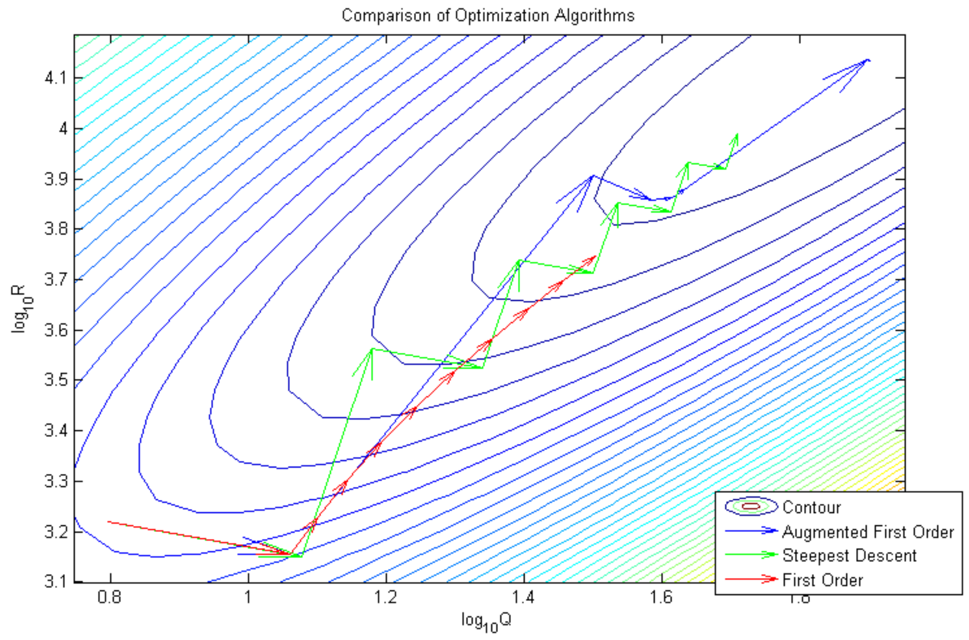

Figure 6: Comparison of the Different Search Algorithms

Algorithm 5 Augmented First Order Method

```
Given $x^0$
$L = L^0$                                        ▷ The user must provide a guess for $L^0$
$k \leftarrow 0$
repeat
    $d^k := -\nabla f(x^k)$
    if $d^k \bullet (x^k - x^{k-1}) > c\|d^k\|\|x^k - x^{k-1}\|$ then        ▷ $0 < c < 1$
        $\alpha = \arg\min_\alpha f(x^k + \alpha d^k)$
        $x^{k+1} \leftarrow x^k + \alpha d^k$
    else
        $x^{k+1} \leftarrow x^k - \frac{1}{L}\nabla f(x^k)$
        if $f(x^{k+1}) > f(x^k) + \nabla f(x^k)^T(x^{k+1} - x^{k+1}) + \frac{L}{2}\|y - x\|^2$ then
            $L = 2L$
            $x^{k+1} \leftarrow x^k$        ▷ This reruns the iteration with the new value for $L$
        end if
    end if
    $k \leftarrow k + 1$
until $d^k = 0$
```

should be considered when for optimal filter tuning.

- The cost function considered is not valid for all tuning situations. Certain situations might require the filter to be more robust to very noisy observations. Therefore, minimizing over the RMSE would not be the ideal case.
- When optimizing over a data set, generally a bound on the converged covariance from the Kalman filter would be set. This would provide additional constraints to consider.
- The analysis in this paper involved the rapid simulation of a Kalman filter and a modified Kalman filter over numerous tracks at various lengths. For slower filters (i.e. the particle filter) or for extremely large data sets, this process would not be tractable.
- Despite multiple evaluations of the filters per algorithm, the optimization method was much faster than a gridding scheme.

7 Conclusion

It is important to note that the goal of this publication was not to tune a specific Kalman filter or modified Kalman filter. Instead, the objective was to prove it is possible to use gradient methods on a function that is not necessary convex in order to optimize a given filter. The steepest descent method, the BFGS algorithm, and the first order methods were all able to find an optimum of the given function. Through a convex combination of the steepest descent algorithm and first order algorithm, it was even possible to decrease the number of iterations required to optimize the filter, which becomes significant if calling the function to return a value is time consuming. The application used in this publication was particularly unique in that the search space was only two dimensional. This allowed the plotting of the search space to ensure the search space had the proper curvature to allow optimization. In higher dimensions it would not be possible to graph the problem in this manner, meaning there is no guaruntee algorithm convergence or global minima. Nevertheless, this publication provides the proof of cocept for gradient-based optimization of a filter.

Appendices

A Filters

This section outlines the Kalman filter and the modified Kalman filter and how they are applied to the problem defined in Section 2.

A.1 Kalman Filter

If the state dynamics and measurements are properly modeled by Eqs. (3) to (7), then a Kalman filter provides an exact solution for estimating the posterior state distribution [16]. The updated state estimate would be a distribution with mean and covariance:

$$\hat{x}_k = F\hat{x}_{k-1} + K_k(z_k - HF\hat{x}_{k-1}) \tag{17}$$

$$\hat{P}_k = (I - K_k H)(F\hat{P}_{k-1}F^T + Q) \tag{18}$$

where K_k is the Kalman gain given by

$$K_k = (F\hat{P}_{k-1}F^T + Q)H^T S_k^{-1} \tag{19}$$

and S_k is the innovation covariance given by

$$S_k = H(F\hat{P}_{k-1}F^T + Q)H^T + R \tag{20}$$

The initial mean and covariance, $\hat{x}_0$ and $\hat{P}_0$, can be calculated based on the first measurement or set a priori.

Without quantization, the observation noise covariance would be set to a value based on knowledge of error in aircraft transponders; however, since the observations are quantized, the covariance needs to be handled differently. A common approach to handling measurement quantization is to use the Sheppard correction [3]. This approach treats the measurements as being uniformly distributed over the interval $(-\Delta/2, \Delta/2)$, which results in $R = \Delta^2/12$.

A.2 Kalman Filter Modified for Quantized Measurements

Curry et al. present an approach for quantized measurements [11]. Instead of computing $\mathrm{E}[x|z]$ using Eq. (17) like the KF, they compute

$$\mathrm{E}[x|z \in A] = \mathrm{E}[\mathrm{E}[x|z]|z \in A] \tag{21}$$

where A represents a quantization interval. The mean and covariance are given by

$$\mathrm{E}[x_k|z_k \in A_k] = \bar{x}_k + K_k(\mathrm{E}[z_k|z_k \in A_k] - H\bar{x}_k) \tag{22}$$

$$\mathrm{cov}[x_k|z_k \in A_k] = \hat{P}_k + K_k \mathrm{cov}[z_k|z_k \in A_k] K_k^T \tag{23}$$

where $\bar{x}_k = F\hat{x}_{k-1}$. Equation (22) is equivalent to Eq. (17) and Eq. (23) is equivalent to Eq. (18) if the measurements are not quantized. Eq. (23) shows that the quantization is treated as if it were uncertainty added after a linear measurement had been processed. The uncertainty induced by the quantization is added to $\hat{P}_k$, which is the minimum covariance if the measurements are not quantized.

Duan, Jilkov, and Li present an efficient method of numerically calculating $\mathrm{E}[z|z \in A]$ and $\mathrm{cov}[z|z \in A]$ [17]. Their approach uses a grid to approximate the

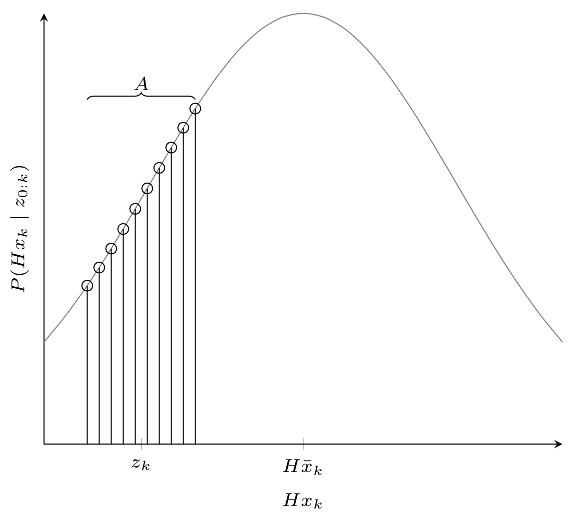

Figure 7: Illustration of the numerical process used to calculate $\mathrm{E}[z|z \in A]$.

probability density function of the predicted observation, $H_k\bar{x}_k$, as a discrete probability mass function. In the problem of vertical state estimation, z is a scalar. Applying this technique to this problem, the equations become

$$\mathrm{E}[z_k|z_k \in A_k] \approx \sum_{i=1}^{L} p_i g_i \tag{24}$$

$$\mathrm{cov}[z_k|z_k \in A_k] \approx \sum_{i=1}^{L} p_i g_i^2 - (\mathrm{E}[z_k|z_k \in A_k])^2 \tag{25}$$

where p_i is the probability associated with the grid point g_i used in approximating the density function as the mass function, and L is the number of grid points. An illustration of the process is shown in Fig. 7. The initial distribution is Gaussian with mean $H\bar{x}_k$ and covariance S_k. The density function is then truncated by the quantization bin. A set of L discrete probabilities are calculated, transforming the probability density function into a probability mass function.

References

[1] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME. Series D, Journal of Basic Engineering*, vol. 82D, pp. 35–45, 1960.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian track-

ing," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002. doi: 10.1109/78.978374.

[3] T. Soderstrom, *Discrete-time Stochastic Systems.* Springer-Verlag, 2002.

[4] E. Sviestins and T. Wigren, "Nonlinear techniques for Mode C climb/descent rate estimation in ATC systems," *IEEE Trans. Contr. Syst. Technol.*, vol. 9, no. 1, pp. 163–174, 2001. doi: 10.1109/87.896757.

[5] A. Gelb, *Applied Optimal Estimation.* MIT Press, 1974.

[6] T. D. Powell, "Automated tuning of an extended kalman filter using the downhill simplex algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 5, pp. 901–908, September-October 2002.

[7] O. V. Korniyenko, M. S. Sharawi, and D. N. Aloi, "Neural network based approach for tuning kalman filter," vol. 25, no. 5, 2005, pp. 1–5.

[8] C. Goodall and N. El-Sheimy, "Intellegient tuning of a kalman filter using low-cost mems inertial sensors."

[9] B. M. Akesson, J. B. Jorgensen, N. K. Poulsen, and S. B. Jorgensen, "A tool for kalman filter tuning," in *17th European Symposium on Computer Aided Process Engineering - ESCAPE17*, 2007.

[10] M. Saha, B. Goswami, and R. Ghosh, *Two Novel Costs for Determining the Tuning Parameters of the Kalman Filter*, Department of Instrumentation and Electronics Engineering Std.

[11] R. E. Curry, W. E. V. Velde, and J. E. Potter, "Nonlinear estimation with quantized measurements-PCM, predictive quantization, and data compression," *IEEE Transactions on Information Theory*, vol. 16(2), pp. 152–161, 1970.

[12] J. W. Andrews, "An improved technique for altitude tracking of aircraft," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-105, Mar. 1981.

[13] S. R. Hall, "Principles of optimal control," Slides, February 2012.

[14] M. J. Kochenderfer, M. W. M. Edwards, L. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Airspace encounter models for estimating collision risk," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, 2010.

[15] R. M. Freund, "The steepest descent algorithm for unconstrained optimization and a bisection line-search method," February 2012.

[16] G. M. Siouris, *An Engineering Approach to Optimal Control and Estimation Theory.* John Wiley & Sons, Inc., 1996.

[17] Z. Duan, V. P. Jilkov, and X. R. Li, “State estimation with quantized measurements: Approximate MMSE approach,” in *Proc. 11th Int Information Fusion Conf*, 2008, pp. 1–6.