# Optimizing the Input for Unstable Airplane Heading Control

Convex Optimization Techniques

Spring 2010

Professor Benjamin Davis

## I. Abstract

An autopilot system was implemented to navigate an F-22 fighter jet through the streets of Manhattan. The bank angle and directional thrust capability of the F-22's engines were used to control its heading. Optimization techniques were used to minimize the error between the actual and desired flightpath of the airplane. The primary constraints for this problem were the maximal bank angle that the plane can achieve, and the width of the streets the plane would by flying through. The non-linear program was simplified into a linear program so that it could be solved easily and efficiently using MATLAB. The autopilot effectively navigates the F-22 through Manhattan; traveling at the speed of sound, the plane follows the route in 22 seconds, without hitting any of the buildings.

## II. Introduction

Autopilot programs are used frequently, for purposes such as giving pilots of commuter planes more freedom during long journeys. For these applications, the flightpath is relatively straightforward and does not include sharp angle changes. However, autopilot programs must be more convoluted to control fighter jets. As airplanes become more capable of large accelerations and sudden shifts in direction, it becomes harder for pilots to perform difficult maneuvers. One such maneuver is navigating an F-22 fighter plane (Figure 1) through the streets of Manhattan, New York, while flying at the speed of sound, 768 miles per hour. Even a skilled pilot would not be physically capable of achieving this feat. Furthermore, the G-forces exerted on the pilot would be exceedingly large as he abruptly changes direction, likely causing injury.


*Figure 1: F-22 Fighter Jet*[1]

To precisely and consistently follow this complicated route and eliminate the risk of injury, it is beneficial to automate this process, having autopilot optimize control of the heading of an airplane (Figure 2). The plane must not hit the buildings it is flying between, nor can it exceed a maximum roll angle, i.e. banking angle, of 90 degrees (Figure 3).

To do so, it is necessary to accurately model the airplane, predict its future behavior, and modify the inputs to the system to optimize the heading of the airplane; this is known as model predictive optimal control. The primary factor influencing the heading of the airplane is the roll of the wings, because increasing the roll angle increases the rate of turning. The heading of the airplane is therefore modeled as a function of its roll angle input.

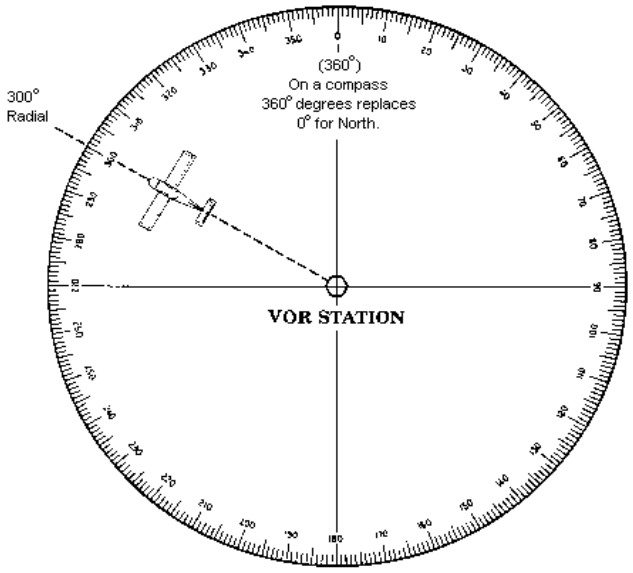



*Figure 2: Airplane Heading*[2]

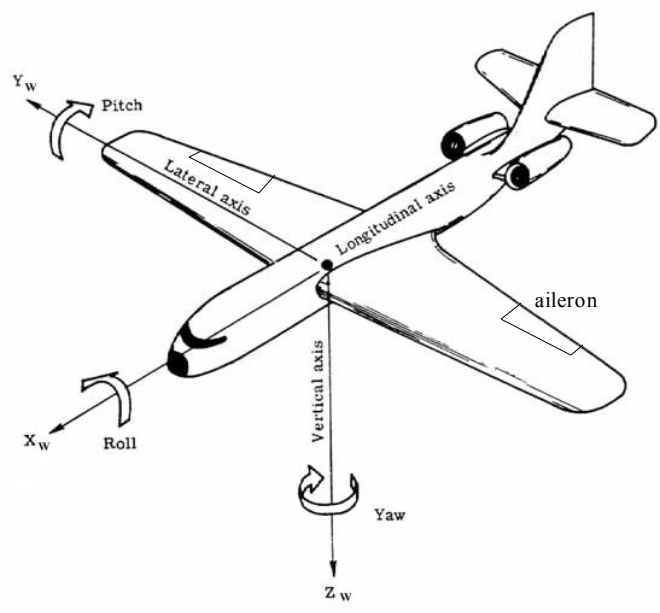


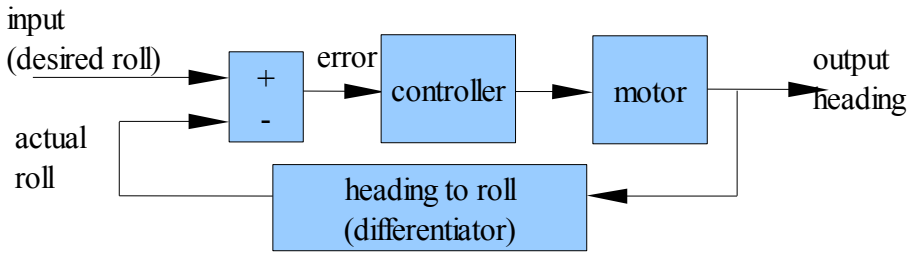

*Figure 3: Airplane Roll*[3]

The airplane of choice was the F-22 fighter jet because of its ease of maneuverability. The rate at which the F-22 changes direction is not only controlled by the rate of roll allowed by its quickly responding ailerons. The rate at which it can change direction is greatly increased using vectored thrust. The nozzle of the F-22's vectored thrust engines can move up and down by 20 degrees (Figure 4), increasing the rate at which the plane can roll by 50%.

*Figure 4: F-22 Engine on Test Stand; Blue Shows Vertical Maximum Angles of Vectored Thrust*[4]

The heading of an airplane is an inherently unstable system, which is typically controlled using a feedback control loop, which responds as a function of the difference between the desired and actual heading (Drawing 1). However, there are inefficiencies in this solution; for example, there is some delay in the system's response, and the output overshoots the desired output and oscillates until the desired value is reached. This is effective when there is enough time for the system to stabilize, as in

commercial jet applications. However, in the case of a fighter jet traveling quickly through Manhattan, the system can be improved by setting the input roll angles to the system directly, a priori, to ensure an undesired output would never be achieved. In order for this method to be used, the exact nature of the system must be known, and then analyzed to determine how it will react to changes in its input roll. For example, to achieve a 50 degree output roll, the input might be set to 30 degrees, the output would overshoot to 40 degrees, the input can then be set to 45 degrees, the output would then overshoot to 48 degrees, and so on, until the desired 50 degree output state is achieved. In this manner, the oscillations of the system caused by overshoot are diminished.




*Drawing 1: Control System: Feedback Loop*

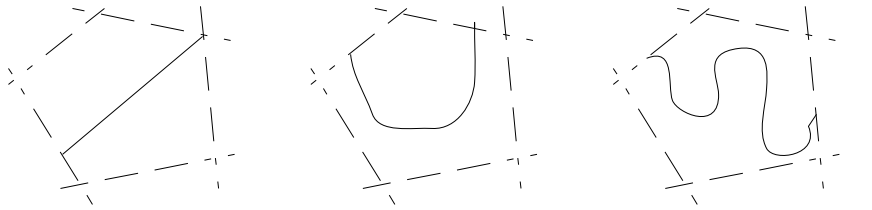
This reduction in oscillations is a major advantage of the method of a priori input control. In a commercial airplane, passengers may find the rocking back and forth of an oscillating system to be very unsettling. In a fighter jet flying through Manhattan, the oscillations may be distracting and cumbersome, and potentially disastrous if they allow the plane to hit buildings. Additionally, if the plane is not stabilized precisely, the plane can lose some of its lift, rendering it less efficient and potentially resulting in a crash.

With an unstable system, any step input can lead to exponential growth of the output. Therefore, the a priori input has to alternate between positive and negative values of the input (relative to the current state of the output) in order to maintain stability and move closer to the desired state. It is advantageous for these signals to be large in magnitude so that the desired state can be reached in the least amount of time. However, for an unstable system, any positive or negative input creates more instability, and as these increase in magnitude it becomes harder to stabilize the system. Therefore, the input signals are set to be within certain acceptable limits defined by the maximum roll angle the airplane can achieve. The input is then optimized a priori in order to minimize the total error that would occur from each change in the input.
Hence, in order to use the more precise method of a priori input, two aspects of the input must be optimized subject to the constraints of the system. The first is that the total error of the system must be as small as possible. The second aspect is minimizing the time it takes for the system to reach a steady state output, which is a function of the magnitude of the input signals.

The following methods control an airplane's heading, taking into account the two aspects of optimization necessary for the method of a priori input. A continuous, nonlinear model of heading with respect to roll is first converted into the discrete domain, and then formulated as a linear programming problem. This linear program can be solved using the Simplex method, which compares values of the objective function at corners of the feasible region defined by the constraints in order to find an optimal solution. Alternatively, it can be solved using an interior point method, which chooses a feasible point and moves in the direction of the minimum value. This second method was implemented using the function *linprog* in MATLAB.

A linear program consists of the minimization of a linear objective (cost) function, with respect to linear constraints. This is shown in the leftmost drawing in Drawing 2, where the cost function is denoted by the solid line and the constraints by the dashed lines. A convex function is one where the

second derivative is nonnegative, (for a vector function, the Hessian must be nonnegative). This means the function must be a straight line or concave up, as shown in the left and middle drawings in Drawing 2. The rightmost drawing is not convex, because there are regions where the function is concave down. It is clear from Drawing 2 that for bounded convex functions, there is only one minimum value (i.e. the local minimum is the global minimum). However, for nonconvex functions, there can be many local minima which are greater than the value of the global minimum. Because all linear functions are convex, there can only be one minimum, and this must be the optimal solution to the problem. Furthermore, a line will either increase or decrease in value when the dependent variable is increased; a nonlinear function may decrease and then increase. The problem is therefore formed as a linear program because it is much easier to solve than a nonlinear problem.



*Drawing 2: Linear* *Nonlinear Convex* *Nonlinear Nonconvex*

## III. Data

The path that the plane is flying through is shown in Figure 5. The plane flies west over the East River, down 23rd St, makes a right onto Broadway, a left onto 34th Street, flies over Madison Square Garden, and makes a left on 7th Avenue. It then flies down 7th Ave, makes a left onto 59th Street, a right onto Broadway, and a left onto 72nd Street, which takes it to the Hudson River. The angle of each turn and the distances between turns are included in Appendix IV. The angles were calculated using a protractor on Google maps (see appendix), and the distances traveled were taken from an exact distance calculating website.[5]

The F-22 was chosen to fly at an altitude of approximately 400 feet. This means that it will clear low buildings like Madison Square Garden, but must fly between taller buildings such as the Marriot Marquis hotel in Times Square. The plane, with a wingspan of 44.5 feet (Figure 1), must not hit the surrounding buildings. All of the streets that the plane is flying through were measured precisely using Google maps and the provided scale (see appendix). The narrowest street that is traveled by the plane is Broadway, just north of 23rd street, and is about 75 feet wide. However, this intersection includes Madison Square Park, and the plane can fly over the trees if necessary.

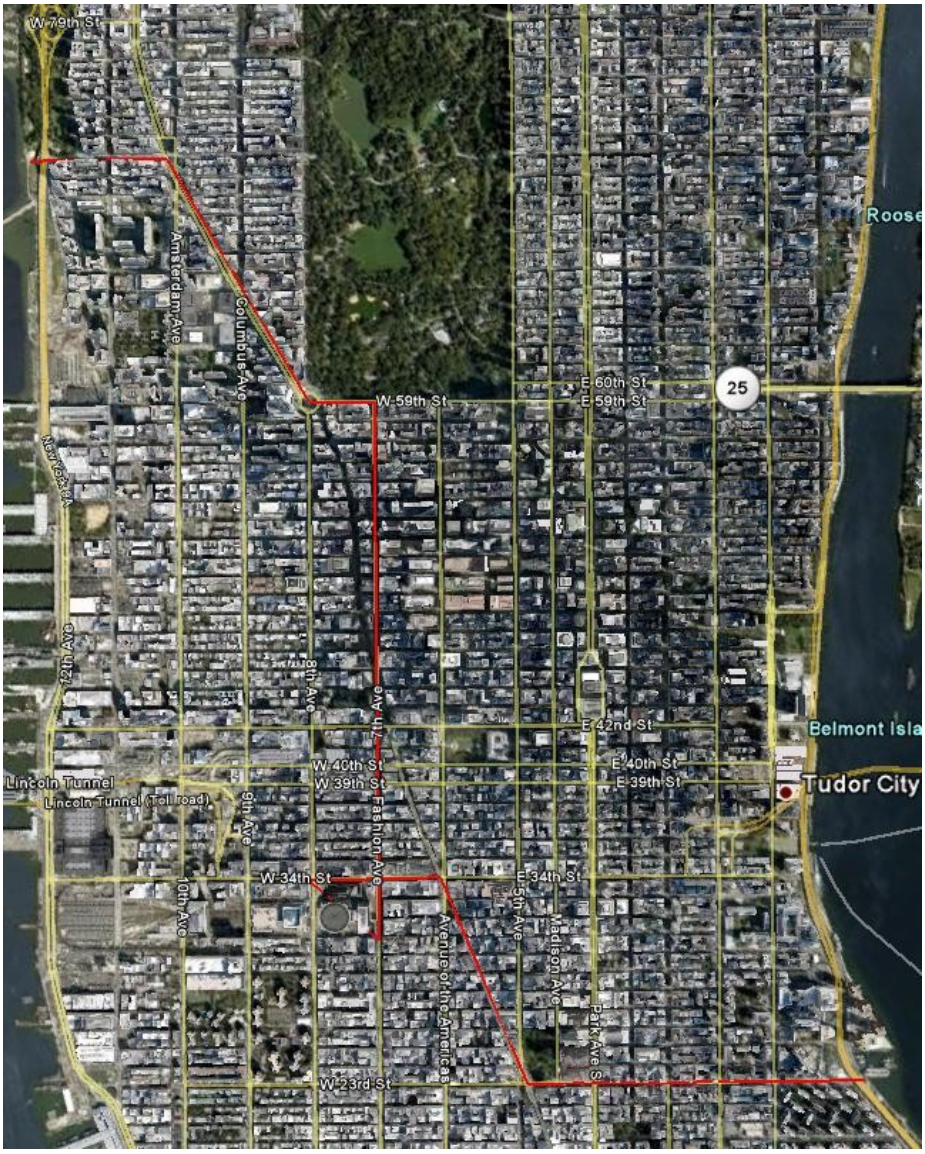


*Figure 5: Path Through Manhattan*

There is also a restriction on the rate at which the plane can change direction. Increasing the roll angle and using the F-22's capability of vectored thrust allows the plane to change its heading rapidly. The roll angle is limited to 90 degrees, and the angle of the nozzle of the vectored thrust engine is limited to 20 degrees[4]. The effect of vectored thrust is taken into account in the model of the airplane, as a gain, K, of 750.

There are also limits on how quickly the roll angle can change. However, these are not taken into account for reasons given in Section V. The plane is traveling at the speed of sound, 1225 km/hr[7], the F-22 is readily capable of achieving. A sampling frequency of 15 Hz was used, because this is fast enough to allow for quick responses from the airplane.

Finally, in order to predict how the airplane will behave, a model of the heading with respect to roll of the airplane is used. This stems from the following model (in the Laplace domain) of the output roll with respect to the input roll[8].

$$G(s)=\frac{R(s)}{X(s)}=\frac{K}{s(s+4)^2}$$

This model is then integrated (divided by *s*) to attain the output heading with respect to input roll, and converted into the discrete domain as detailed in the next section. The final expression details the heading of the airplane as a function of discrete variables representing its previous input roll angles and output headings.

## IV. The Heading Model

The actual heading of an airplane is related to the commanded roll/bank of an airplane by the following equation,

$$\frac{d^4y}{dt^4}+8\frac{d^3y}{dt^3}+16\frac{d^2y}{dt^2}=Kx$$

where $\boldsymbol{x}$ is the input bank angle, $\boldsymbol{y}$ is the actual heading, and $\boldsymbol{K}$ is the vectorized thrust gain. This is derived from the transfer function, G(s), adapted from Shinners, which assumes that heading, pitch, and roll are independent[8].

$$G(s)=\frac{Y(s)}{X(s)}=\frac{K}{s^4+8s^3+16s^2}=\frac{K}{s^2(s+4)^2}$$

In order to use optimal programming techniques and have the computer interface with the airplane, the system is converted into a digital form using a zero order hold and a sampling frequency, *fs*. Using a sampling rate of 15 Hz and a vectored thrust gain of K = 750, the digital transfer function is,

$$H(z)=\frac{0.000555z^{-1}+0.005506z^{-2}+0.004949z^{-3}+0.0004035z^{-4}}{1-3.532z^{-1}+4.65z^{-2}-2.705z^{-3}+0.5866z^{-4}}$$

Thus the digital time signal is,

$$y(n)=0.000555x(n-1)+0.005506x(n-2)+0.004949x(n-3)+0.0004035x(n-4)$$
$$+3.532y(n-1)-4.65y(n-2)+2.705y(n-3)-0.5866y(n-4)$$

## V. Formulating the Model as a Linear Programming Problem

Using the techniques outlined in Appendix III, the system can be written in the following matrix form,

$$y=\Phi x+C$$

This formulation allows the output stream for a given input stream to be calculated using simple matrix calculations, which emphasizes the linear nature of the system.

(Note: **bold face** symbols represents vector quantities.)

Now let us state the optimization problem.

It is desired that the system have a desired output, $\mathbf{y_D}$, which is the user outlined flight path. $\mathbf{y_D}$ is a vector of the desired headings for each sample period.

The goal is to find $\mathbf{x_{Opt}}$, a vector of bank angles for each sample period, that when fed into the system yields $\mathbf{y_D}$.

$$\boldsymbol{y_D=\Phi x_{Opt}+C}$$

One set of constraints are that the absolute value of each input roll angle cannot exceed a certain value,

$$|x_{Opt}(n)|<U \quad \forall n$$

This constraint is because there is a maximum angle at which the plane can bank, 90°.

Another constraint can be that the slew rate, i.e. the rate at which the input roll angle changes, cannot exceed some value,

$$|x_{Opt}(n)-x_{Opt}(n-1)|<S \quad \forall n$$

Thus, the problem can be stated as, find the input stream of commanded bank angles $\boldsymbol{x}$, that minimizes the magnitude of the error vector, $\boldsymbol{e}$, defined by $\boldsymbol{e} = \mathbf{y}\text{-}\mathbf{y_D}$, subject to the constraint that the input can not exceed the absolute value of 90° and the input cannot change by more than a set rate.

Also, it is important to remember that the input vector is in terms of time steps, so a faster sampling rate means each commanded bank is held for less time. (e.g. with a sampling rate of 15 Hz the 42nd value in the input vector is the commanded/desired bank angle for the time between 2.8 and 2.8667 seconds.) Thus for a given velocity and a set distance, the length of the required input/output vector increase as the sampling frequency increases.

Thus once the distance, velocity, and sampling rate are chosen, the problem can be put into the following form,

$$\begin{aligned} \min \quad & \|\Phi x + C - y_D\|_2 \\ \text{s.t.} \quad & |x(n)| \le U \quad \forall n \\ & |x(n) - x(n-1)| \le S \quad \forall n \end{aligned}$$

This formulation is not linear due to the magnitude and absolute value calculations. It is strongly desired to form this problem into a linear problem because linear programing techniques are quicker and guarantee to yield a global minimum. Also linear programing techniques do not require any information or proprieties of the Jacobin or Hessian of the system. Also since the system and the constraints are linear in themselves, it strongly suggests that the problem can be phrased as a linear programing problem.

The first simplification is instead of minimizing the $L_2$ norm of the error (i.e. the square root of the sum of the squares of each error), the $L_\infty$ norm (i.e. the maximum error) is minimized. This means that instead of minimizing every error we just minimize the worst case error $\zeta = \|\Phi x + C - y_D\|_\infty$ and add the constraint that the absolute value of all the individual errors must be less than $\zeta$. So we seek to minimize only the scalar value, $\zeta$, which is equal to the greatest error. This means the objective function will now be a simple linear scalar value and the constraints now include the $N$ inequality constraints that the error at time $n$ be less than the scalar value $\zeta$. Thus in order to minimize $\zeta$, the optimization algorithm will need to minimize the errors at every time step.

A second simplification is noting that an absolute constraint can be re-written in the following manner,

$$|a| \le b \;\rightarrow\; a \le b \;\;\&\;\; -a \le b \;\rightarrow\; -b \le a \le b$$

Thus a single non-linear constraint can be written as two linear constraints.

Finally, it was found that the slew rate constraint was not really applicable in our scenario as the commanded bank angle can change between the two bounding constants of -90° and 90° in one time step when using our desired sampling frequency of 15 Hz. So it was dropped as a constraint in our problem.

The original problem can now be written in LP form as

$$\begin{aligned} \min \quad & \zeta \\ \text{s.t.} \quad & -\zeta \le \Phi x + C - y_D \le \zeta \\ & -U \le x \le U \end{aligned}$$

Or, by writing each inequality by itself,

$$\begin{aligned} \min \quad & \zeta \\ \text{s.t.} \quad & \Phi x + C - y_D \le \zeta \\ & y_D - (\Phi x + C) \le \zeta \end{aligned}$$

$$x \le U$$
$$-x \le U$$

In standard (MATLAB) LP form,

$$\begin{aligned} \min \quad & f^T \cdot u \\ \text{s.t.} \quad & A \cdot u \le b \\ & -[U\,\infty] \le u \le [U\,\infty] \end{aligned}$$

where $u$, $f$, $A$, and $b$ are defined as

$$u = [x(1)\,x(2)\,x(3)\ldots x(N)\;\zeta]^T \quad \& \quad f = [0\,0\,0\ldots 0\,1]^T$$

$$A = \begin{bmatrix} \Phi & -1 \\ -\Phi & -1 \end{bmatrix} \quad \& \quad b = \begin{bmatrix} (y_D - C) \\ (C - y_D) \end{bmatrix}$$

where -1 is a column vector of length N.

## VI. Proof of Feasibility and Existence of Optimal Solution

A feasible solution is one that falls within the set defined by the constraints of the optimization problem. An optimal solution provides the minimum value that the cost function can attain within the feasible set. This particular problem is continuous, because the inequality constraints are linear functions of the roll, and the objective function is the error, which can take on any positive value.

In order to prove that a continuous optimization problem (with an objective function and constraints that are continuous) has a feasible and optimal solution, we must look at the boundaries of the feasible set. In our problem, the variables (vector $u$) are the roll angles that are input into the plane at any time, and the error of the heading that the plane has with respect to the desired heading. Because these constraints are continuous, and the values of the roll can only be from -90 to 90 degrees (they are bounded), there must be a value in that range which produces the optimal solution to the problem. This optimum may not be a unique solution, but there will be only one minimum value in the feasible region. Without the constraint on the roll angle, there would not be an optimal solution, because the model would incorrectly predict that increasing the angle to infinity would reduce the error to zero, because increasing roll angle increases the rate at which the plane turns.

The final component of $u$ is the heading error, which is unbounded because the plane's heading could be off by a large number (the constraint on the plane not hitting the buildings is not included in the formulation of the linear program, but is checked later). Instead of checking that the variables are bounded, we can check to see that as the variable magnitude increases, the objective function value approaches infinity.

$$\lim f(u) = \infty \quad \text{as} \quad |u| \to \infty$$

This equation is true in our problem, because as the error approaches infinity, the objective function, which we defined as the error itself, clearly approaches infinity. $\zeta$ cannot be negative, because the absolute value of the error must be less than or equal to $\zeta$.

Since all the necessary functions are bounded (or bounded at infinity) and continuous, there must be a minimum point somewhere in the spectrum of the objective function. This point would either be on an corner point of the feasible set, if bounded, or at a local minimum, in the case where approaching infinity would make increase our objective value. Either way, we have proven our problem has at least one solution.

Because this is a linear program, meaning that the objective function and the constraints are linear functions, this problem is necessarily convex. This means if any minimum value is found in the feasible set, it must be the global minimum. All of the functions that our problem consists of (heading, roll, error etc.) are linear, so if the objective function increases when a variable is increased, it will never decrease when increasing that variable.

## VII. Results

Since the problem is linear, MATLAB's *linprog* command was used. The time the program took to perform the optimization was 7 seconds. There were 332 variables, 331 for the input at each time step and 1 for the error. This is because the number of variables is equal to the total distance*3600*fs/v+1, where fs = 15 Hz and v = 1225 km/hr. Thus, there were 2*331 error constraints and 2*331 input magnitude constraints. The F-22 took 22 seconds to go from start to finish, and did not collide with any structures. The results were plotted and broken down into the following graphs (figures 6-8 in the appendix):

1. Figure 6
    a. The optimal roll input in degrees at any given time
2. Figure 7
    a. The desired output of the system
    b. The output if the desired roll was fed into the system directly
    c. The output if a simple controller was used with unity feedback
    d. The output if the optimal input was fed into the system
3. Figure 8
    a. The total allowed error off-center that the plane can veer at any given time
    b. The actual error that the plane veered

From the graph plotting the optimal roll input (Figure 6), we see what looks like many random spikes. Upon further inspection and by comparing that plot to the plot of the desired output (Desired Response, Figure 7), we see that the spikes start growing in magnitude right before each change in the the desired output. Furthermore, the spikes reach each of their peak magnitudes right before that change occurs. However, we can clearly see that the spikes are clipped at 90 degrees, the maximum allowable angle for roll. This tells us two things. Firstly, and intuitively, the fact that the largest spikes in the roll input happen right before big heading changes tells us that to get the quickest change in heading, the roll must be very large. Secondly, because the roll was capped at 90 degrees, the optimization program determined that in order to generate the optimal heading, it should start the roll of the plane earlier than the actual change in heading occurs. This also accounts for the large negative spikes in the roll following the positive spikes, because if the plane banks before a turn, while it is still supposed to be traveling straight, there must be an opposite bank so that the plane stays on track and does not hit the buildings. A combination of the positive and negative roll spikes produces the final, quick and optimized, optimal heading.

The roll graph shows our optimal program at work, but in order to see how our optimal solution compares to traditional methods of solving this problem, we can look at Graph 7. Graph 7 overlays the plots of the desired output (labeled Desired Response) and the optimally produced output (labeled Optimal), as well as 2 other traditional approaches to solving our problem. As far as comparing our optimized heading with the desired heading, we see that the two plots are so similar that it is hard to tell them apart. The only differences are small irregularities at the corners of the optimally produced plot. These irregularities are not too much of a concern since they last for a very short time and are so small in magnitude that an airplane would probably hardly feel them.

In contrast, if the desired heading is fed into the system without any feedback loop (labeled Direct

Response), the output of the system does not follow the desired trajectory at all. If a traditional feedback control loop with phase lead [8] is implemented, the plane's path (labeled w/ Feedback) is much closer to the desired path. However, the feedback system overshoots the desired heading by a large percentage, following the overall shape but unable to make abrupt turns. This will allow the plane to collide with buildings. Solving this problem using optimization techniques clearly leads to much better results than could have ever been obtained otherwise.

The third and final graph (Figure 8) shows the distance that the plane veers from its desired heading, plotted against the total allowed error before the plane hits any buildings. This metric was measured using Google maps, and a chart of the distances and the original maps (with measuring markups) are given in the appendix. As stated previously, the width of the streets were not directly included in the constraints. Instead, for a given speed of the plane, the error of which the plane veered from its heading was minimized; if the error was small enough, the plane would not hit any buildings. Since the width of the streets is a constraint (as the plane cannot be allowed to hit buildings), if the plane veered too much for a given street's width, a slower speed would be required, and the optimization problem would be redone.

Looking at the graph, we can instantly see that this was not the case. At all times, the distance that the plane veered (the bottom plot) is completely under the maximum allowable veer (the top plot). The maximum drift from the center of the street was 7 meters which is less than half the wingspan. It should be noted that two periods of time were a special case. The top plot contains three local maximums. The second and third local maximums are given a maximum allowable veer of 11 meters; in reality, these streets are narrower. However, the second and third local maximums represent the plane flying over Madison Square Garden (MSG) and Central Park South (CPS), respectively. The height of MSG is less than 400 feet, the altitude of the plane, and CPS borders a park, so the plane can fly over trees. In both of these cases, the maximum allowable veer is therefore very large, because there are no tall buildings in the immediate vicinity. Therefore, a maximum veer of 11 meters was chosen, because it is equal to the maximum allowable veer, which occurs at 23$^{rd}$ street. Thus, optimization techniques were effectively used to create an autopilot program which steers the F-22 through the streets of Manhattan, without crashing.

## Appendix I. Additional Graphs

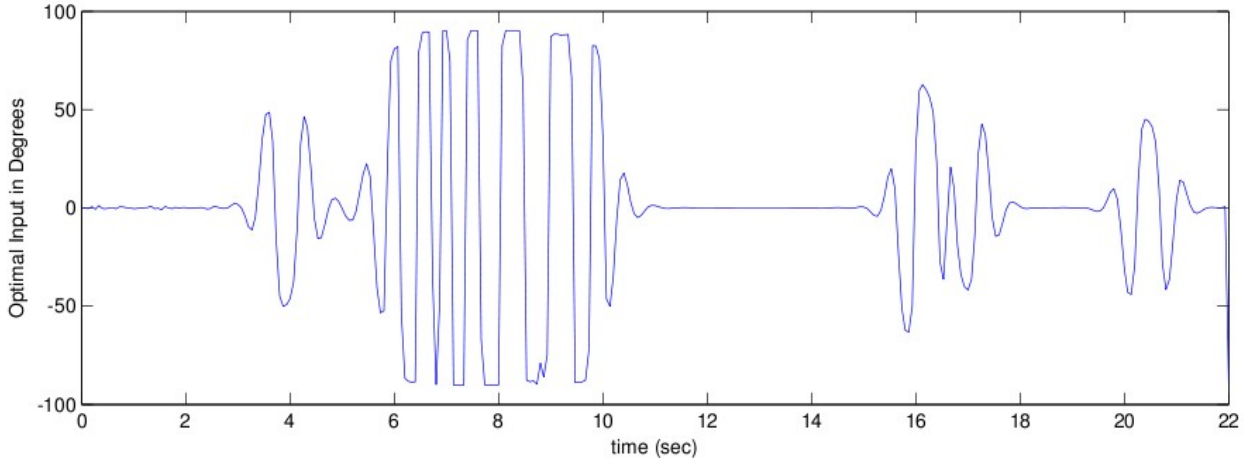



*Figure 6: Roll of the plane*

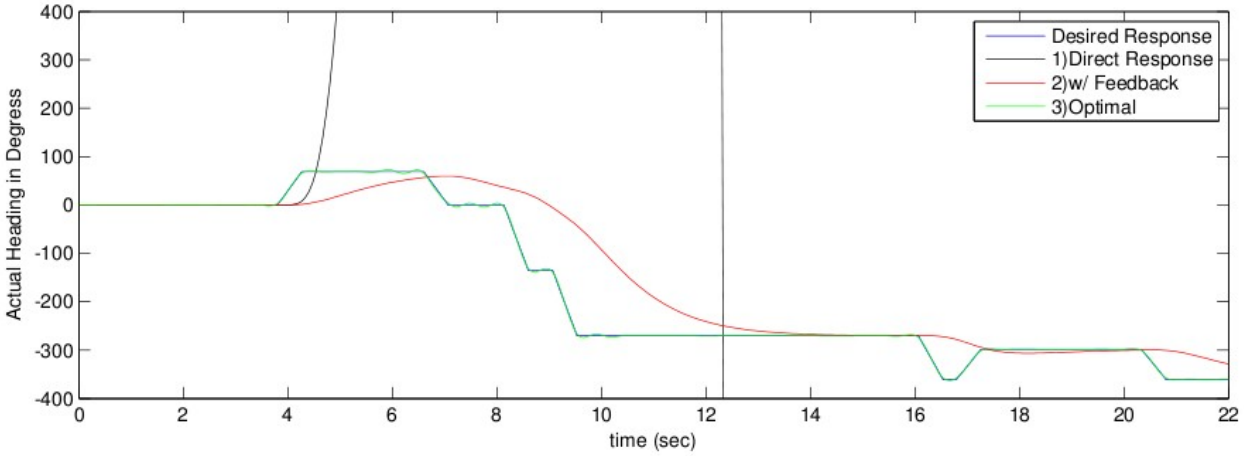



*Figure 7: Heading of the plane*

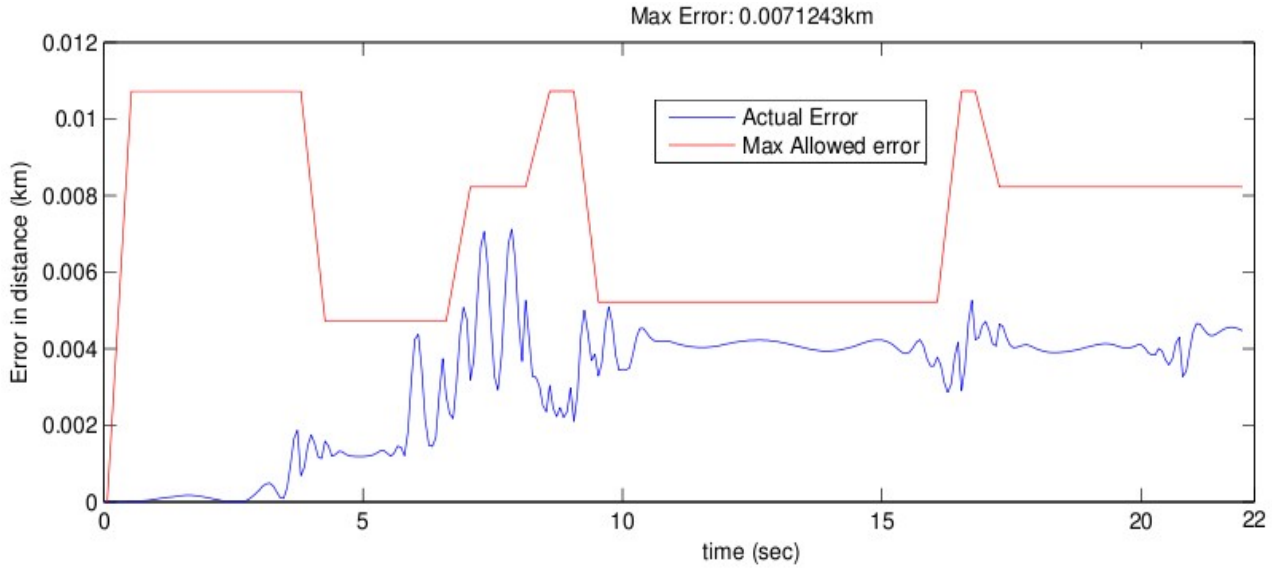



*Figure 8: Distance the plane Veered Off-Center*

## Appendix II. Derivation of the Heading Model

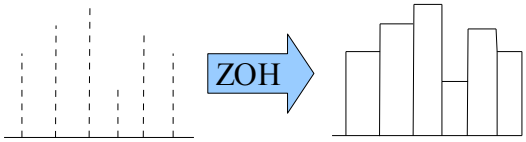
The heading (integrated roll) of an airplane is defined by,

$$\frac{d^4 y(t)}{dt^4}+8\frac{d^3 y(t)}{dt^3}+16\frac{d^2 y(t)}{dt^2}=K\,x(t)$$

where ***x(t)*** is the input bank angle, ***y(t)*** is the actual heading, and ***K*** is the vectorized thrust gain. This is derived from the transfer function, G(s), adapted from Shiners, which assumes that heading, pitch, and roll are independent.

$$G(s)=\frac{Y(s)}{X(s)}=\frac{1}{s^4+8s^3+16s^2}=\frac{1}{s^2(s+4)^2} \quad [8]$$

The transfer function is obtained by taking the Laplace transform of the function in the time-domain, assuming zero initial conditions. In order to change this from a continuous system to a discrete one, a z-transform is performed. To take the data from the digital domain to the continuous domain, a zero ordered (sample and) hold, ZOH, must be performed. This holds the value of the sample for one time cycle (Drawing 3).




*Drawing 3: Discrete Points: Zero Ordered Hold*

The system transfer function is placed in series with a zero ordered (sample and) hold, ZOH, circuit that has a transfer function of

$$G_0(S)=\frac{1-e^{-Ts}}{s} \quad \text{(Shiners Eq 4.19 p 218)}$$

Thus the over all system becomes,

$$H(s)=G_0(s)G(s)=\frac{1-e^{-Ts}}{s^3(s+4)^2}$$

Using partial fraction expansion we obtain,

$$H(s)=\left(\frac{1-e^{-Ts}}{s}\right)\left(\frac{K}{s^2(s+4)^2}\right)=K(1-e^{-Ts})\left(\frac{1}{16s^3}-\frac{1}{32s^2}+\frac{3}{256s}-\frac{1}{64(s+4)^2}-\frac{3}{256(s+4)}\right)$$

Taking the Z-Transform yields,

$$H(z)=K\left(\frac{z-1}{z}\right)\left(\frac{T^2z(z+1)}{32(z-1)^3}-\frac{Tz}{32(z-1)^2}+\frac{3z}{256(z-1)}-\frac{Tze^{-4T}}{64(z-e^{-4T})^2}-\frac{3z}{256(z-e^{-4T})}\right)$$

With T = 0.0667s ($f_s$ = 15Hz) and K = 750 the transfer function is,

$$H(z)=\frac{0.000555z^{-1}+0.005506z^{-2}+0.004949z^{-3}+0.0004035z^{-4}}{1-3.532z^{-1}+4.65z^{-2}-2.705z^{-3}+0.5866z^{-4}}$$

Each exponent of ***z*** in the above equation represents a time delay. For example, $z^{-1}$ represents the previous time step. If this term is multiplied by ***x***, then in the time-domain, it would be represented by ***x(n-1)***, the input signal at the time step before the current one, ***x(n)***. In terms of a difference equation this digital system can be written in the discrete time domain as,

$$y(n)=0.000555x(n-1)+0.005506x(n-2)+0.004949x(n-3)+0.0004035x(n-4)$$
$$+3.532y(n-1)-4.65y(n-2)+2.705y(n-3)-0.5866y(n-4)$$

## Appendix III . Formulating a Model as an Optimization Problem

If a linear system is subjected to a number of impulse inputs, ***x(n)***, scaled by some numbers, $h_k$, then

the ensuing finite impulse response, ***y(n)***, is:

$$y(n)=h_0x(n)+h_1x(n-1)+h_2x(n-2)+...+h_kx(n-k)$$

so the Z-transform yields the following:

$$\frac{Y(z)}{X(z)}=H(z)=h_0+h_1z^{-1}+h_2z^{-2}+...+h_kz^{-k}$$

Hence, for some input sequence,

$$[x(0)\ x(1)\ x(2)\ ...\ x(N)]^T$$

the filtered output would be

$$[y(0)\ y(1)\ y(2)\ ...\ y(N)]^T$$

which is given by,

In the time domain, $y=h*x$ (where * is the convolution operator)

In the Z-domain (frequency), $Y(z)=H(z)X(z)$

This filtering process can also be written in the time-domain as a matrix operator in the following form,

$y=\phi x$ , where $\phi$ is an NxN toeplitz matrix, defined as follows:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ . \\ . \\ . \\ y(N-1) \\ y(N) \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0 & . & . & . & 0 \\ h_1 & h_0 & 0 & 0 & . & . & . & 0 \\ h_2 & h_1 & h_0 & 0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . \\ h_k & ... & h_2 & h_1 & h_0 & 0 & ... & 0 \\ 0 & h_k & ... & h_2 & h_1 & h_0 & ... & 0 \\ . & . & . & . & . & . & . & . \\ 0 & ... & 0 & h_k & ... & h_2 & h_1 & h_0 \end{bmatrix} * \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ . \\ . \\ . \\ x(N-1) \\ x(N) \end{bmatrix}$$

From the matrix form, it is easy to see that the filtering process is linear.

An infinite impulse response (IIR) system takes into account not only previous inputs to the system, but also previous outputs from it. This infinite impulse response evolves with the following difference equation,

$$y(n)=a_0x(n)+a_1x(n-1)+a_2x(n-2)+...+a_kx(n-k)$$
$$-b_1y(n-1)-b_2y(n-2)-...-b_ky(n-k)$$

It has the following Z-Domain transfer function

$$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}+...+a_kz^{-k}}{1+b_1z^{-1}+b_2z^{-2}+...+b_kz^{-k}}$$

This IIR system can also be written in the form of a matrix multiplication,

$$y=\Phi x$$

The Matrix, $\Phi$ , can be constructed by first creating a toeplitz matrix, $\phi$ using the elements of the numerator, *a*, that correspond to the FIR part of the system. Then, each row can be recursively updated, starting with n = 1, as follows,

$$\Phi_{(\text{row}\,n)}=\Phi_{(\text{row}\,n)}-b_1\Phi_{(\text{row}\,n-1)}-b_2\Phi_{(\text{row}\,n-2)}-...-b_k\Phi_{(\text{row}\,n-k)}$$

In element wise notation:

$$\Phi_{(n,i)}=\Phi_{(n,i)}-b_1\Phi_{(n-1,i)}-b_2\Phi_{(n-2,i)}-...-b_k\Phi_{(n-k,i)} \quad \forall i=1...N$$

This is true because each row of $\Phi$ , when multiplied by x, represents the values of y. As such, previous rows of $\Phi$ are scaled by values of b and subtracted from the current row of $\Phi$ to obtain the value of the current row.

Finally, the effect of initial conditions can also be added into the matrix model as a constant, ***C***, that get's added to each output, such that

$$y = \Phi x + C$$

This matrix constant is initialized using the initial conditions, $x_0$ and $y_0$ which are obtained by looking back in time at k-n previous inputs and outputs of the system.

$$C(n) = a_{n+1} x_0(k) + a_{n+2} x_0(k-1) + ... + a_k x_0(n) - b_{n+1} y_0(k) - b_{n+2} y_0(k-1) - ... - b_k y_0(n) \quad \forall\, n < k$$

The matrix is then update recursively using the following formula,

$$C(n) = -b_1 C(n-1) - b_2 C(n-2) - ... - b_k C(n-k)$$

Thus in recap, we can formulate an IIR system as just a simple linear matrix operation given by,

$$y = \Phi x + C$$

## Appendix IV. Measurements

| Turn | angle (degrees) | current heading (degrees) | Street | distance (km) | distance (miles) |
|---|---|---|---|---|---|
| 23 to broadway | 70 | 70 | 23 | 1.38 | 0.86 |
| broadway to 34 | -70 | 0 | broadway | 0.95 | 0.59 |
| 34 to MSG | -135 | -135 | 34 | 0.53 | 0.33 |
| MSG to 7th | -135 | -270 | MSGto7th | 0.38 | 0.24 |
| 7th to 59 | -90 | -360 | 7th | 2.28 | 1.41 |
| 59 to broadway | 65 | -295 | 59 | 0.27 | 0.17 |
| broadway to 72 | -65 | -360 | broadway | 1.19 | 0.74 |
| total | -360 | | | | |

| Street | width (m) | allowed error (half the distance - half the wingspan) |
|---|---|---|
| 23 | 35 | 10.72 |
| broadway | 23 | 4.72 |
| 34 | 30 | 8.22 |
| MSG | 50 | 18.22 |
| 7th | 24 | 5.22 |
| 59 | 50 | 18.22 |
| broadway | 30 | 8.22 |
| 72 | 30 | 8.22 |

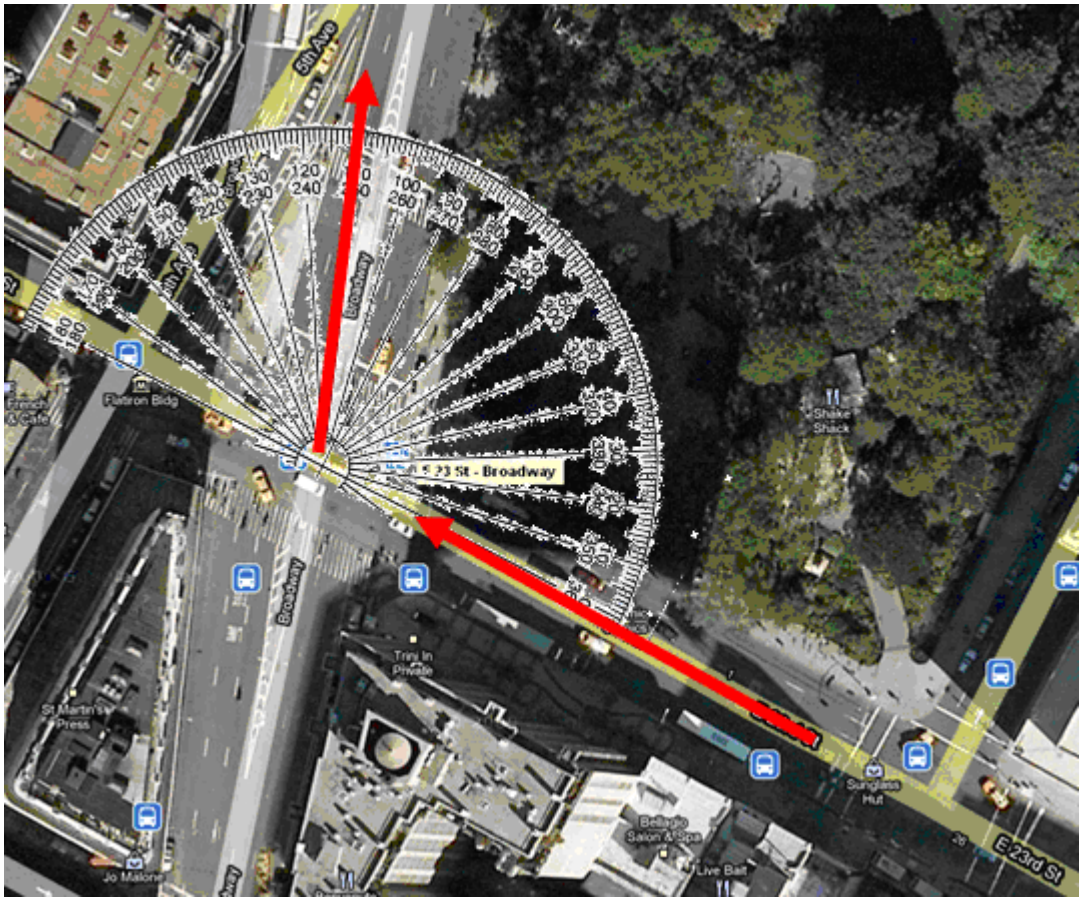
| | | |
|---|---|---|
| half wingspan = | 6.78 | meters |

## References

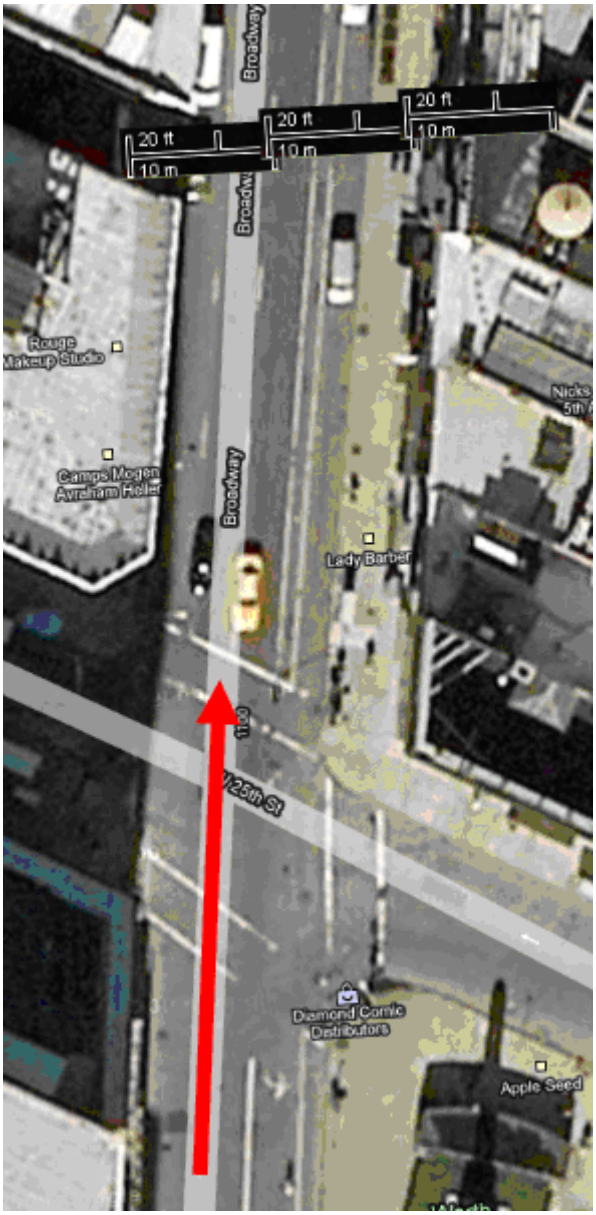

1. *<http://www.aerospaceweb.org/question/planes/f35/f16-f35-f22.jpg >Accessed 5/7/10*
2. *<http://www.allstar.fiu.edu/aero/images/L1-Pic5-2.gif>Accessed 4/25/10*
3. *<http://mtp.jpl.nasa.gov/notes/pointing/Aircraft_Attitude2.png >Accessed 4/25/10*
4. *<http://science.howstuffworks.com/f-22-raptor5.htm> Accessed 5/7/10*
5. *<http://www.daftlogic.com/projects-google-maps-distance-calculator.htm> Accessed 5/7/10*
6. *<http://www.ausairpower.net/API-Metz-Interview.html> Accessed 5/7/10*
7. *<http://www.fighter-planes.com/jetmach1.htm > Accessed 5/10/10*
8. *Shinners, Stanely M. Advanced Modern Control System Theory and Design. New York: John Wiley and Sons, Inc., 1998. p45.*
9. *Boyd, Stephen, et. al. Control Applications of Nonlinear Convex Programming. Information Systems Laboratory, Electrical Engineering Department, Stanford University. Elsevier, Sept 17, 1997.*

**Width of 23rd st**

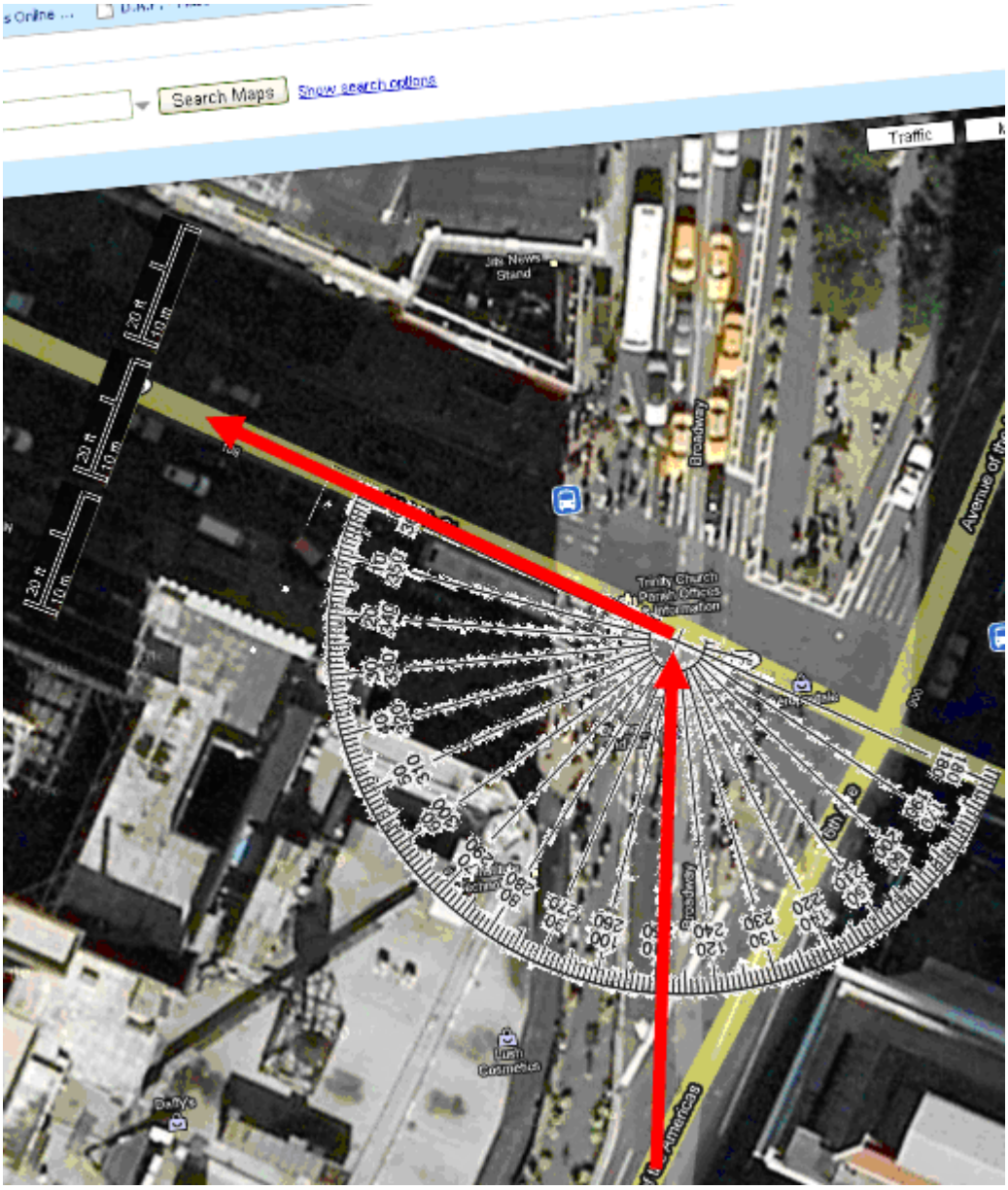
## Angle of the turn from 23rd st to Broadway



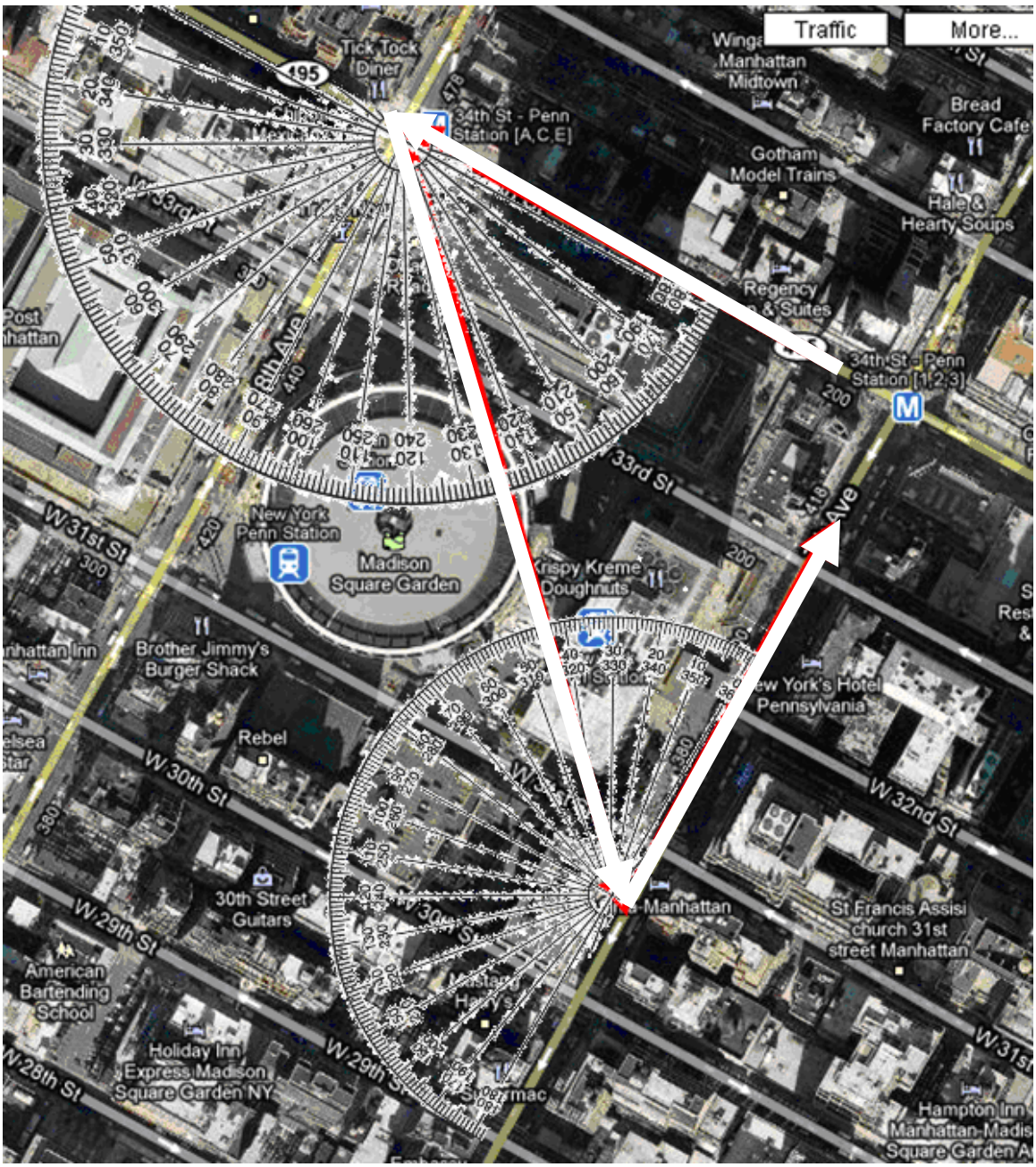
## Width of Broadway (midtown)



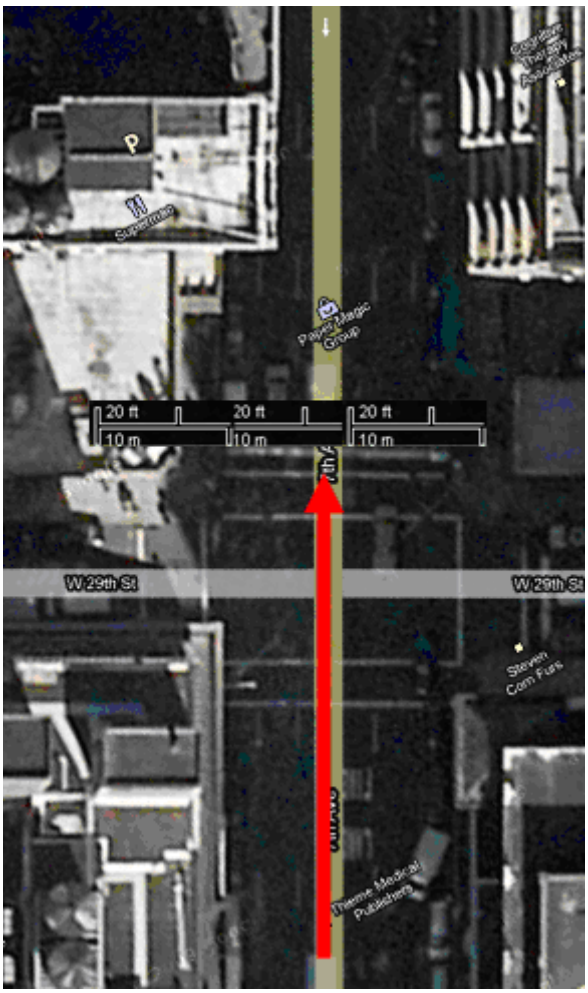
**Turn from Broadway to 34th st and width of 34th st**



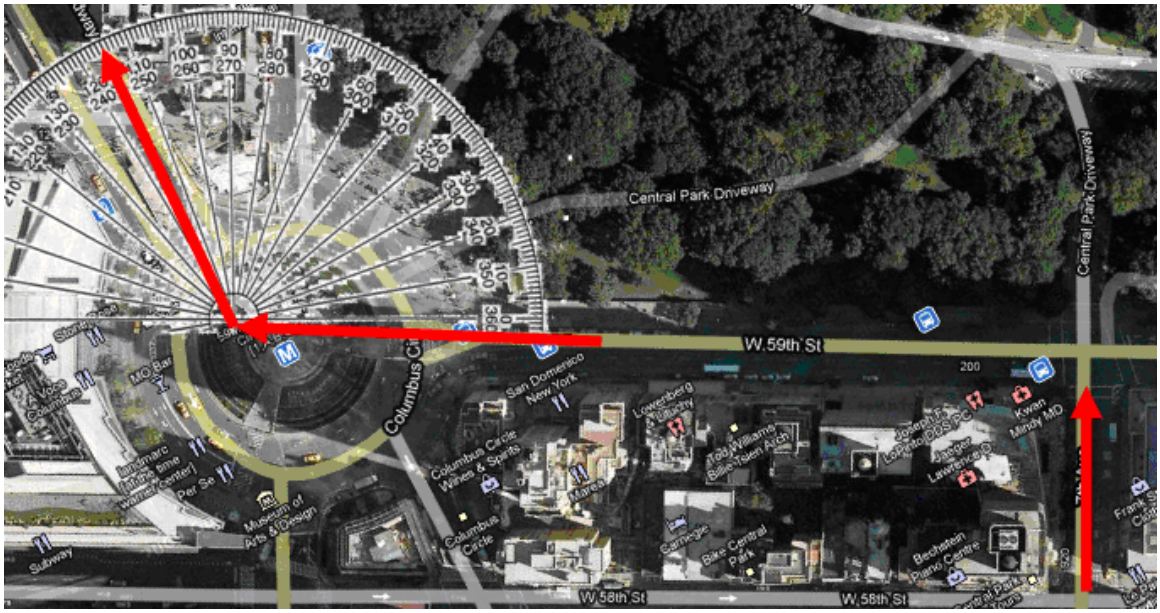
**Turn from 34th over MSG and then turn onto 7th**



## Width of 7th ave

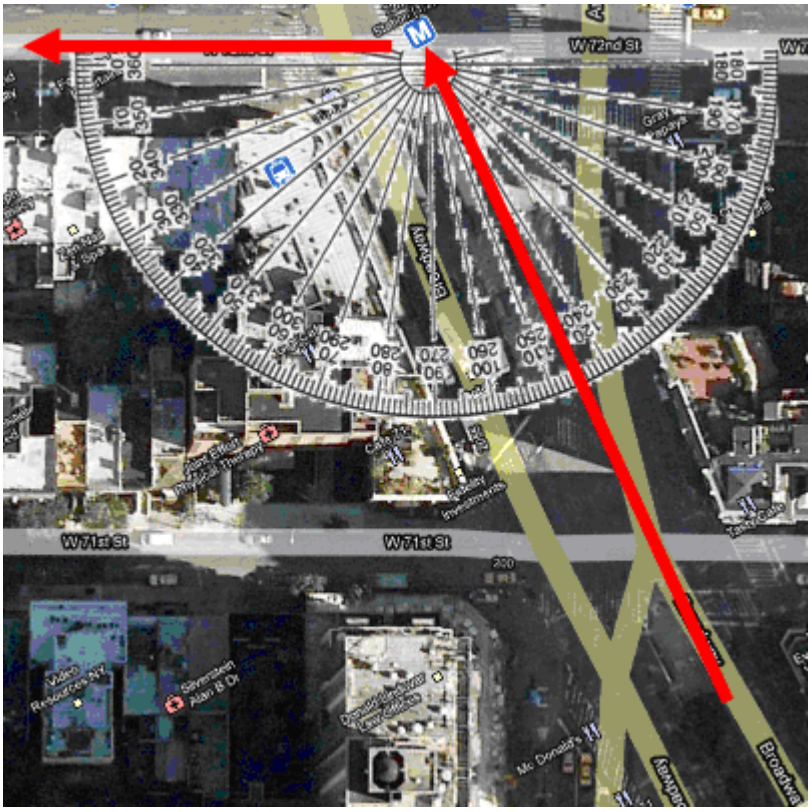


**Turn from rom 7th to 59th st and then from 59th st to Broadway**

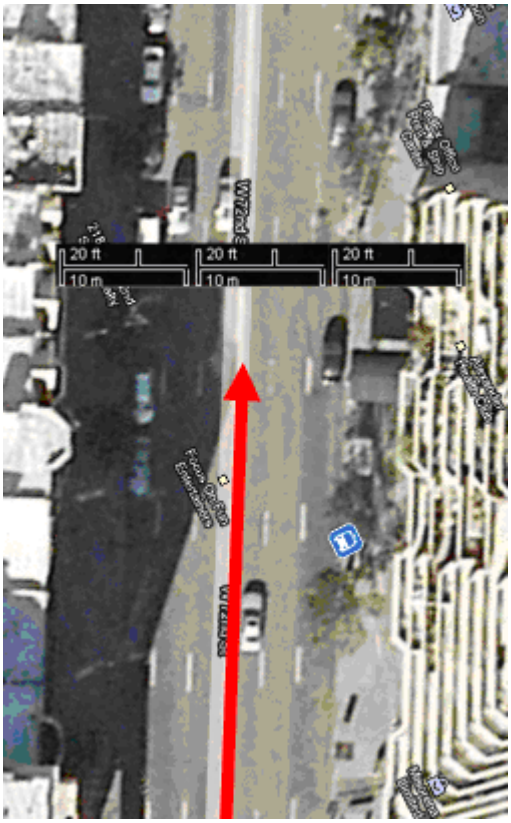


**Width of Broadway (uptown)**

## Turn from Broadway onto 72nd st



**Width of 72nd st**



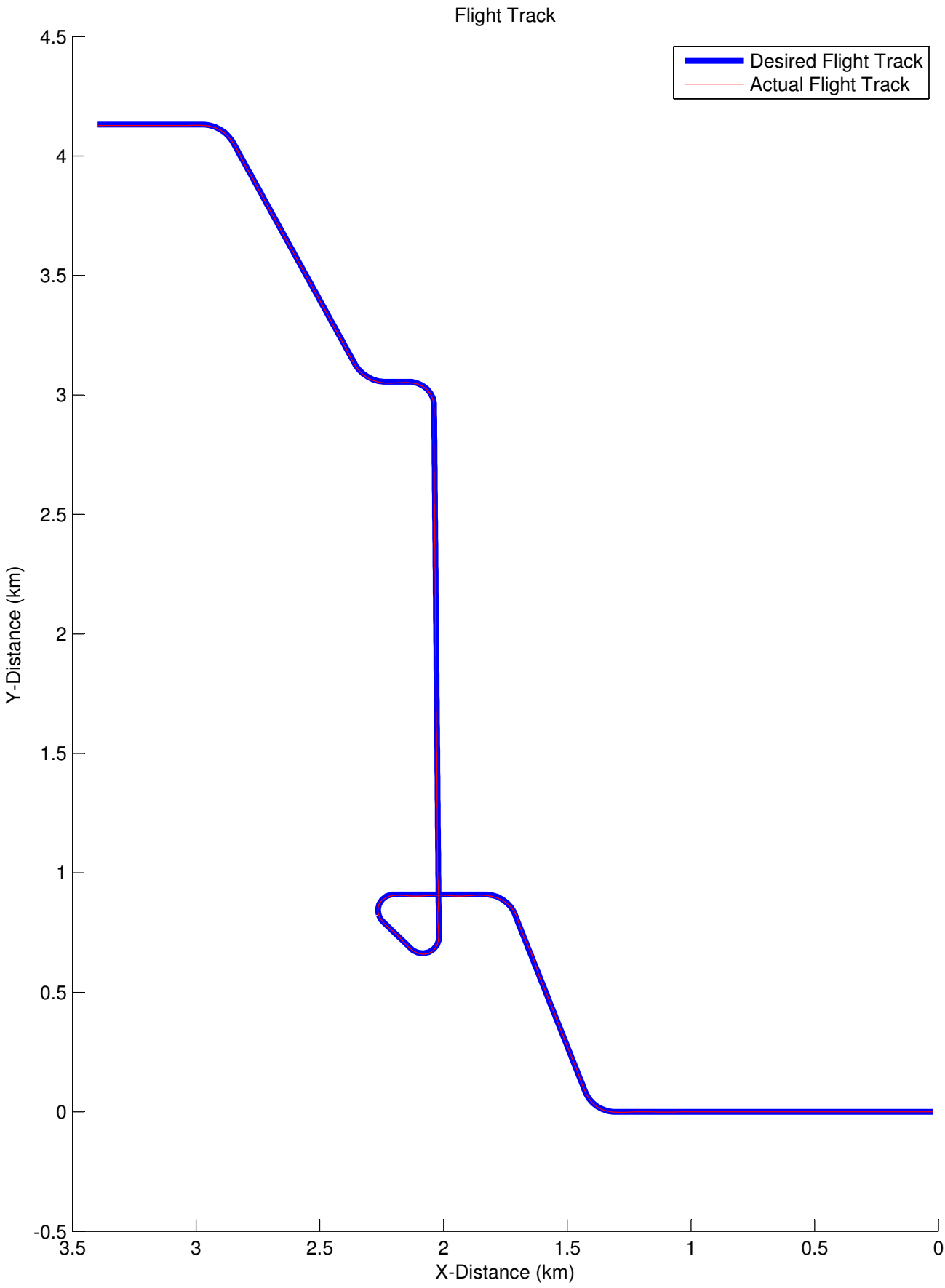
```
clc
clear all
%-----------------------------------------------
%Define the System
fs = 15;Ts =1/fs;%Sampling Rate
K = 750;%The gain of the system and the vectorized Thrust
G = tf(K*[0 0 0 0 1],[1 8 16 0 0]);%This is the Analog system
Gd = c2d(G,Ts);%This converts the system to te digital/discreete domain
[a,b] = tfdata(Gd,'v');%This gets the coeffeciants of the Digital Transfer function
%obtain TF for simple unity feedback with a phase lead controller to be used in comparision
C = tf(.19*[1 -1],[0 1],Ts);% create the phase lead controller
FGd = feedback(series(C,Gd),1);%create the open loop transfer function of the closed loop system
[af,bf] = tfdata(FGd,'v');%obtain the coeffecients of the feedback system for comparision
% ---------------------------------------
%Define the desired flight path
%Headings = [heading(degree)  distance(km)  maxError(m)]
headings = [0        1.35     10.72;...
            69.005       0.95     4.72;...
            0        0.5      8.22;...
            -135     .3       10.72;...
            -270.5     2.38   5.22;...
            -360     0.23     10.72;...
            -298     1.194    8.22;...
            -360     .5       8.22];

th = headings(:,1);
d = headings(:,2);
v = 1225;%This is the velocity of the plane in km/hour , Mach 1
%-------------------------------------------------------------------------------
%Create the vector, yD, of desired headings per a unit of time.
%First, let's smooth out the transitions to accomidate the
%actual turns. Let's give each turn .5 sec, so subtract
%out a half second from each strech,
%1/4 sec from the begining and 1/4 sec from the end.
%Now to simulate the turns themselves, in between the
%straight pathes add .5 seconds of a ramp
%that starts at the old direction and transitions to the
%new direction.
%Note: dt*v/3600 = d, Thus the following subtracts off .5 sec worth of distance
d = d-.5*v/3600;
%add a quater second back to the first and last strech because they only
%lose time from one transion region
d([1,end]) = d([1,end])+.25*v/3600;
%subtract another half second from the first strech because of the extra
%delay added for the transions regions from 0 to 0
d(1) = d(1) - .5*v/3600;
% yD is going to be the vector of desired headings per a unit time
yD = 0;
%Note: #samples = 3600*d*fs/v
for n= 1:length(th)
    yD = [yD linspace(yD(end),th(n),ceil(fs/2)) th(n)*ones(1,round(d(n)*3600*fs/v))];
end
% ---------------------------------------------------------------
```

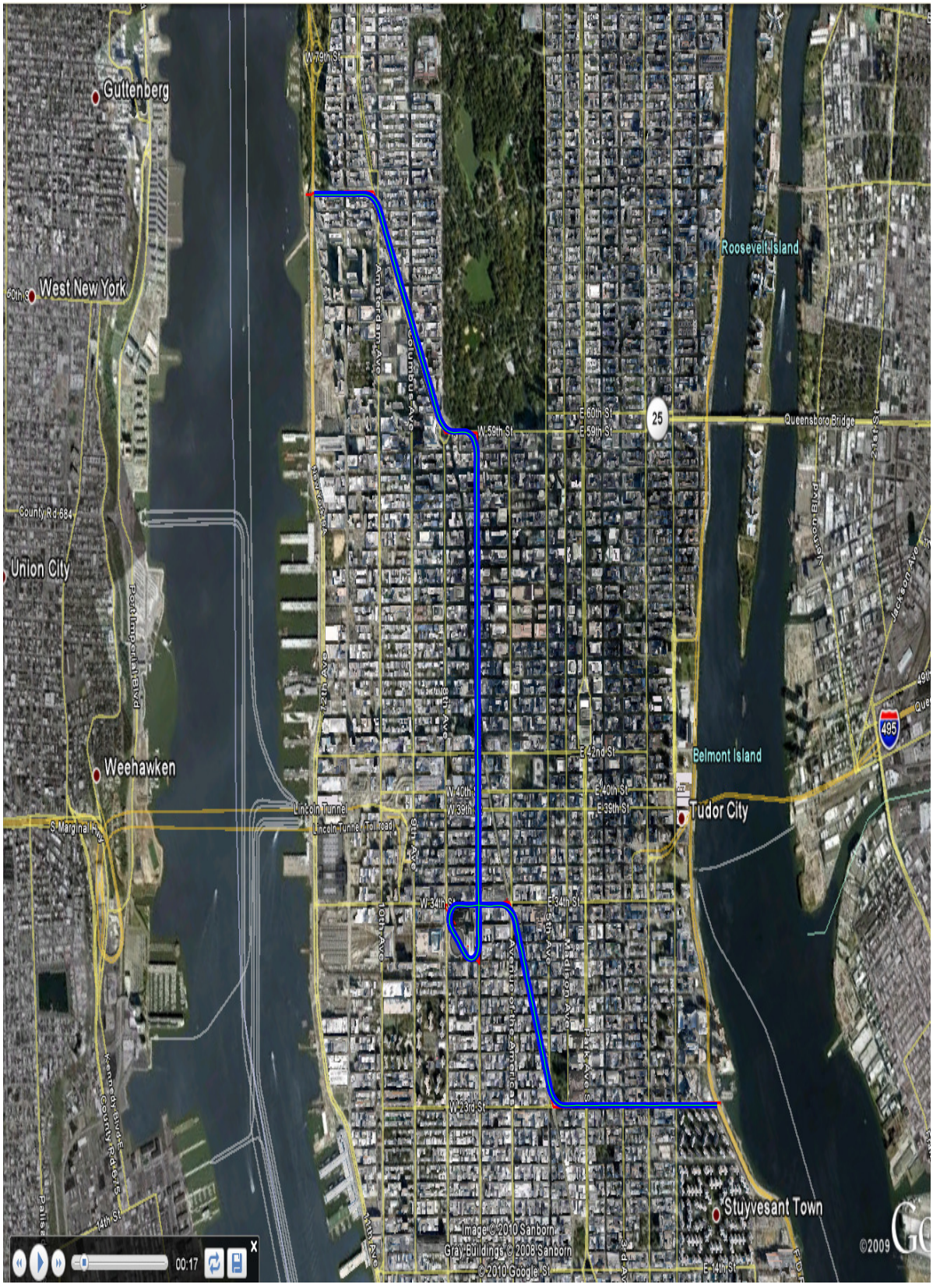
```
U=90; %Enter the constraints on the max Amplitude of input
tic%time how long the optimization takes
% a,b, are the transfer function of the system
% yD is the desired flight path
% U is the constraint on the input amplitude
[x_opt,z,e] = OptController(a,b,yD,U); %optain optimal input
toc
%-----------------------------------------
%obtain Results
% by simulating feeding the respective inputs into the system
y_opt = filter(a,b,x_opt);
y = filter(a,b,yD);
yf = filter(af,bf,yD);

%-------------------------------------------------------------
%Plot the results
n = Ts*(0:length(yD)-1); %This is a vector of time stamps

figure(1)
subplot(211) %Plots the system responces
plot(n,yD,'b',n,y,'k',n,yf,'r',n,y_opt,'g')
axis([0,n(end),-400,400])
legend('Desired Response','1)Direct Response','2)w/ Feedback','3)Optimal')
xlabel('time (sec)');ylabel('Actual Heading in Degress')
subplot(212) %Plots the calculated Optimal Input signal
plot(n,x_opt)
set(gca,'XLim',[0,22])
xlabel('time (sec)')
ylabel('Optimal Input in Degrees')
disp(['max error: ' num2str(e) '%'])

figure(2) %plots the Desired and Optimal Flight Track
xm = cumsum((v/3600)*Ts*cosd(yD));
ym = cumsum((v/3600)*Ts*sind(yD));
xmO = cumsum((v/3600)*Ts*cosd(y_opt));
ymO = cumsum((v/3600)*Ts*sind(y_opt));
clf
line(xm,ym,'LineWidth',3)
line(xmO,ymO,'Color','r')
%reverse the x-axis direction so that 0, the start point,
%is on the right
set(gca,'XDir','reverse')

figure(3) %Plots the Errors
d_error = sqrt((xm-xmO').^2 + (ym-ymO').^2);
subplot(211) %Plot The error in terms of Degrees
plot(n,abs(yD-y_opt'));
xlabel('time (sec)')
ylabel('Error in Degrees')
title(['Max Error: ' num2str(e) '%'])
subplot(212) %Plots the Error in terms of offset from the center of the street in km
de = headings(:,3)/1e3;
dEmax = 0;
```

```
for i= 1:length(de)
    dEmax = [dEmax linspace(dEmax(end),de(i),ceil(fs/2)) de(i)*ones(1,round(d(i)*3600*fs/v))];
end
plot(n,d_error,n,dEmax,'r')
xlabel('time (sec)')
ylabel('Error in distance (km)')
title(['Max Error: ' num2str(max(d_error)) 'km'])
legend('Actual Error','Max Allowed error')

figure(4) %Plots the Flight Path on top of the map
NYC = imread('map1.tif');
imagesc(NYC)

hax1 = gca;
axis off
hax2 = axes('Position',get(hax1,'Position'),...
            'XAxisLocation','top',...
            'YAxisLocation','right',...
            'Color','none');
set(hax2,'XDir','reverse')

x_scale=1.2;
xm = xm/x_scale;
xmO = xmO/x_scale;

line(xm,ym,'LineWidth',3,'Parent',hax2);
line(xmO,ymO,'Color','g','Parent',hax2);

axis(hax2,[-1.47,5,-.8,5])
axis off
```

```
function [x_opt,z,e] = OptController(a,b,yD,U,x0,y0)
%Inputs :
% a,b are the system's transfer function
% yD is the desired Output
% U is the max Amplitude constraint onn the input
% x0, y0 are the intial condtions
% if x0, y0 are omitted they are assumed to be zero
if(nargin<5),x0 = zeros(1,length(a)-1);y0=x0;end
L = length(yD);
S = SysMatrix(a,b,L);%obtain the System Matrix, see Apendix for dervation details
[C,z] = SysIntial(a,b,x0,y0,L);%Obtain the offsets due to the intial conditions.
[x_opt,e] = OptControl(S,C,yD,L,U);%Obtains the optimal input
end

function [inOpt,e] = OptControl(S,C,inD,L,U)
%This function sets up and solves the linear program for the Problem as
% defined in the  "Formulating the Model as a Linear Programming Problem" section
%inputs:
% S is the system matrix
% C is the intial conditions constant vector
% inD is the desired output
% L is the length of the desired output
% U is the constraint on the max amplitude of the input
U = [U*ones(1,L) inf];
f = [zeros(1,L) 1]';
A = [S -1*ones(1,L)';-[S ones(1,L)']];
B = [(inD-C) (C-inD)]';
x = linprog(f,A,B,[],[],-U,U);
inOpt = x(1:end-1);
e = x(end);
end

function S = SysMatrix(a,b,L)
% This function is used to create the System matrix as defined in the appendix
%Inputs:
% a,b, are the system's transfer function
% L is the lengh of the desired input vector
k = length(b);
S = toeplitz([a zeros(1,L-k)],[a(1) zeros(1,L-1)]);
for i = 2:k-1
    S(i,:) = S(i,:) -b(i:-1:2)*S(1:i-1,:);
end
for i = k:L
    S(i,:) = S(i,:) -b(k:-1:2)*S(i-(k-1):i-1,:);
end
end

function [C,z] = SysIntial(a,b,x0,y0,L)
% This function creates the constant vector, C, that is used to adjust the system
% based off of it's intial conditions
% Inputs:
% a,b, are the system's transfer function
% x0, y0 are the intial condtions
```

```
% L is the lengh of the desired input vector
k = length(b);
C = zeros(1,L);
z = zeros(1,k-1);
for i = 1:k-1
    C(i) = a(i+1:end)*x0(1:k-i)'-b(i+1:end)*y0(1:k-i)' -b(2:i)*C(i-1:-1:1)';
    z(i) = C(i) + b(2:i)*C(i-1:-1:1)';
end
for i = k:L
    C(i) = -b(2:k)*C(i-1:-1:i-(k-1))';
end
end
```


Flight Track
Desired Flight Track
Actual Flight Track
Y-Distance (km)
X-Distance (km)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
3.5
3
2.5
2
1.5
1
0.5
0


Guttenberg
West New York
Union City
Weehawken
Roosevelt Island
Belmont Island
Tudor City
Stuyvesant Town
Lincoln Tunnel
Lincoln Tunnel (Toll road)
Queensboro Bridge
Palisade Interstate Pkwy
Kennedy Blvd E
County Rd 675
County Rd 684
W 79th St
W 59th St
E 60th St
E 59th St
W 42nd St
E 42nd St
W 40th St
W 39th St
E 40th St
E 39th St
W 34th St
E 34th St
W 23rd St
E 14th St
Avenue of the Americas
Madison Ave
5th Ave
6th Ave
10th Ave
11th Ave
12th Ave
Columbus Ave
Amsterdam Ave
Park Ave S
Jackson Ave
Vernon Blvd
21st St
25
495
00:17
Image © 2010 Sanborn
Gray: Buildings © 2008 Sanborn
© 2010 Google
©2009