Snap, Crackle, and Pop

Peter M. Thompson[1]
Systems Technology, Inc., Hawthorne, CA, 90250

A minimum time move is designed with limits on any number of derivatives. The common names for the first three derivatives are velocity, acceleration, and jerk. The not so common names for the next three derivatives are snap, crackle, and pop. Derivatives beyond pop (the sixth order derivative) are simply given a number. The minimum time moves are designed for moves between known stationary points. An alternative strategy is to design a minimum time move with limits on velocity and acceleration, followed by a linear filter called a linear jerk filter. The strategy to use depends on the application. Directly limiting higher order derivatives works well for movements between set points. The alternative strategy is more flexible because it allows changes to be made before coming to a stop.

I. Introduction

The immediate onset of acceleration has a large jerk that can be detrimental to equipment, can result in loud bangs, and can be uncomfortable for riders in transportation systems. A better strategy is to ramp up the acceleration by placing a limit on jerk. A potentially even better strategy is to ramp up the jerk by limiting the next higher derivative, and so on. A solution is presented for a minimum time move between stationary positions with limits on any number of derivatives.

There are no standard names for derivatives higher than third. Snap, crackle, and pop are not official but have been used respectively for the fourth, fifth, and sixth derivatives. An internet search has found references to these derivative names which then qualify their use by respectively calling them "facetious" and "not serious," as though engineers are not entitled to an easily remembered mnemonic. Most people know Snap, Crackle, and Pop as the three elves on Kellogg's Rice Krispies cereal boxes, introduced in the early 1930s.

Algorithms are presented and then compared for two strategies: 1) a many-derivative-limited trajectory between stationary points, and 2) a two-derivative-limited trajectory followed by a linear filter. The so-called "linear jerk filter" reduces not just the jerk but all of the higher derivatives. A minimum time move with limits on velocity and acceleration is a classic result of optimal control, which has the very nice feature that it can be implemented starting from any point in the error and velocity phase space, in other words, a change can be made in the target before coming to a stop. The same is theoretically true for the first strategy, but the implementation starting from any point in the multi-dimensional phase space is very complicated. The second approach has the not insignificant advantage of being much simpler and hence more robust. But which strategy to use depends on the application. The first approach is good for numerically controlled moves for tools, printers, amusement park rides, and the like. The second strategy is better for command following applications as varied as aircraft autopilots and telescope mount control.

The paper has the following sections:

Section II: The classic problem of min time with a limit on acceleration and velocity
Section III: The same with a limit included for jerk
Section IV: The same with limits included for snap, crackle, and pop
Section V: Limits on acceleration and velocity followed by a linear jerk filter
Section VI: Approximating the various limits as a low pass filter

---

[1] Chief Scientist, AIAA Senior Member.

Section VII: A discussion of feedback and feedforward
Section VIII: Conclusions

The minimum time trajectories are meant to be created by a command shaper and then used as input to a system. How the system is designed to follow the trajectory is not the subject of this paper, but for the sake of illustration the combined feedback and feedforward system in Section VII is included.

Nomenclature

V_{max} = max velocity (1st derivative)
A_{max} = max acceleration (2nd derivative)
W_{max} = max jerk (3rd derivative)
S_{max} = max snap (4th derivative)
C_{max} = max crackle (5th derivative)
P_{max} = max pop (6th derivative)

$r(t)$ = position
$v(t)$ = velocity (1st derivative)
$a(t)$ = acceleration (2nd derivative)
$w(t)$ = jerk (3rd derivative)
$s(t)$ = snap (4th derivative)
$c(t)$ = crackle (5th derivative)
$p(t)$ = pop (6th derivative)

$\max(v)$ = max achieved velocity (which is $\leq V_{max}$)
$\max(a)$ = max achieved acceleration (which is $\leq A_{max}$)

R_c = length of move
t_a = time to reach max acceleration
t_v = time to reach max velocity
t_c = time to reach R_c (length of move)

$f_{jerk}(s)$ = linear jerk filter

II. Limits on Velocity and Acceleration

The objective is to follow a command with limits on velocity and acceleration. This is a classic minimum time problem. The acceleration steps between $\pm A_{max}$ and zero and the problem is to determine the switching times.

Movement between Stationary Points

The command shaper is used to create trajectories between stationary points. There are two cases depending on whether or not the velocity limit is reached.

Case 1: Short movement (V_{max} not reached)

The max velocity is not reached if $R_c < V_{max}^2 / A_{max}$

$R_c = A_{max}(t_c / 2)^2 = \text{length of trajectory}$

$t_c = 2\sqrt{R_c / A_{max}} = \text{final time}$

$\max(v) = \sqrt{R_c A_{max}} = \text{max velocity achieved}$

Switching times = $0,\ t_c / 2,\ t_c$

Case 2: Long movement (V_{max} is reached):

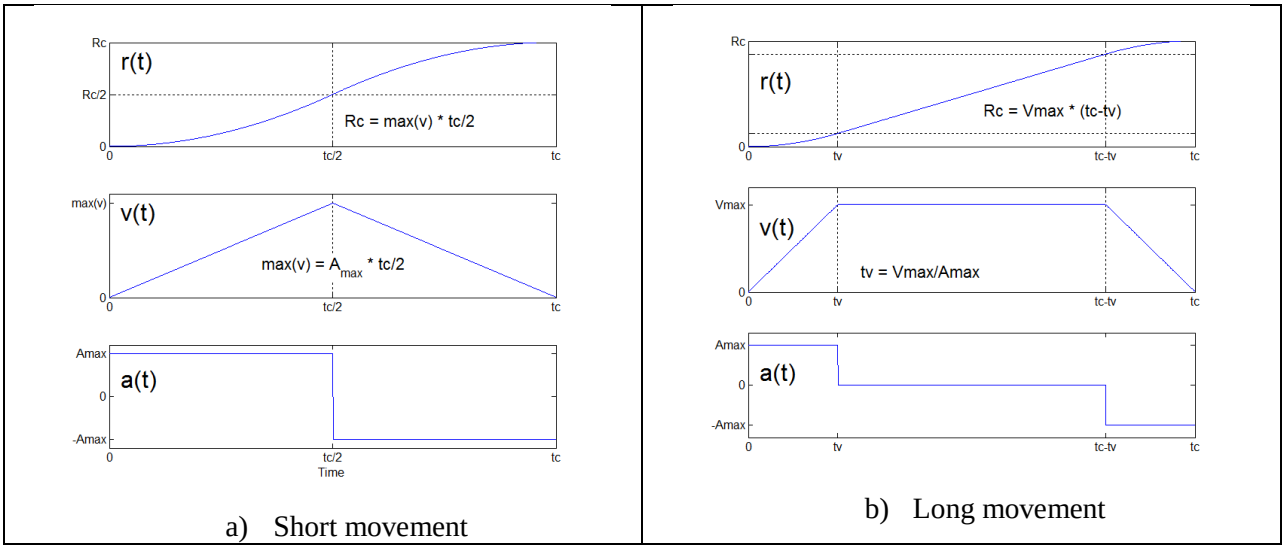
$R_c = V_{max}(t_c - t_v) = \text{length of trajectory}$

$t_v = V_{max} / A_{max} = \text{time to reach max velocity}$

$t_c = t_v + R_c / V_{max} = \text{final time}$

Switching times = $0,\ t_a,\ t_c - t_a,\ t_c$

Example trajectories are shown in Figure 1.



a) Short movement

b) Long movement

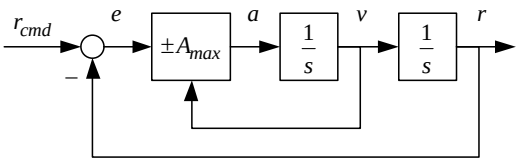
Figure 1: Limits on Velocity and Acceleration

Command Following

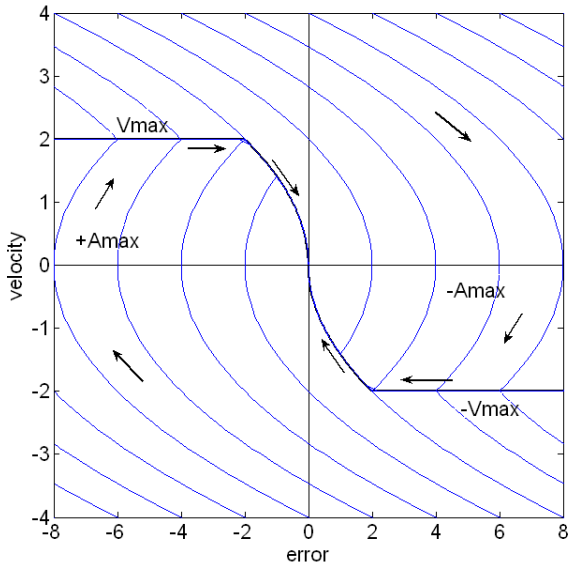
The more general implementation is to allow an arbitrary input to a command shaper and then to limit the velocity and acceleration of the output. A block diagram of the command shaper is shown in Figure 2a. The block labeled $\pm A_{max}$ contains logic that outputs acceleration equal to $\pm A_{max}$ or zero, depending on

the location of the velocity and error in the phase plane shown in Figure 2b. The acceleration command is zero if the trajectory is at the origin or on the $\pm V_{max}$ lines.

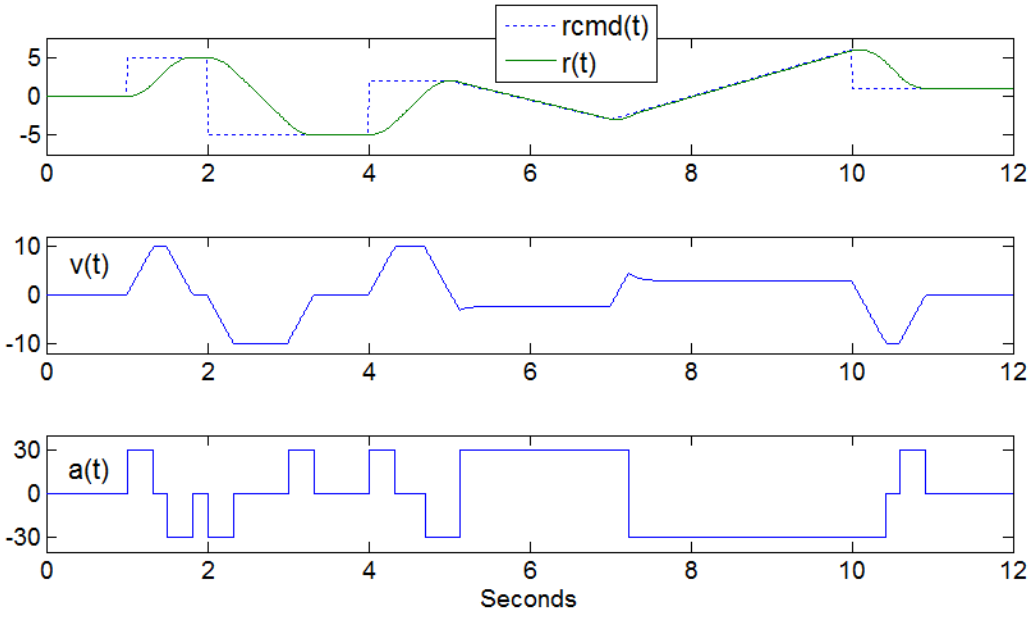
A digital implementation must take into account boundary line crossings that occur between sample times, otherwise the trajectory will chatter between $\pm A_{max}$ rather than stay at zero. Logic is needed that recognizes that line crossing has occurred in the last sample period, and then breaks the trajectory into two and sometimes three pieces. An example trajectory is shown in Figure 2c.



a) Command Shaper



b) Switching Lines



c) Sample Trajectory (V_{max} = 10, A_{max} = 30)

Figure 2: Command Following with Limits on Velocity and Acceleration

III. Limits on Velocity, Acceleration, and Jerk

Acceleration does not change instantaneously but ramps up and down. The trajectory is assumed to move between stationary points. There are four cases of interest depending on whether or not the maximum velocity and acceleration are reached. Descriptions of the four cases are listed below.

Case 1: Short movement (neither V_{max} nor A_{max} is reached)

$R_c = W_{max} t_c^3 / 32 = \text{length of trajectory}$

$t_c = (32 R_c / W_{max})^{1/3} = \text{final time}$

$\max(a) = W_{max}(t_c / 4) = (R_c W_{max}^2 / 2)^{1/3} = \text{max acceleration achieved}$

$\max(v) = W_{max}(t_c / 4)^2 = (R_c^2 W_{max} / 4)^{1/3} = \text{max velocity achieved}$

Switching times = 0, $\frac{1}{4}t_c$, $\frac{3}{4}t_c$, t_c

Case 2: Medium movement (V_{max} not reached, A_{max} is reached)

$$R_c = A_{max}\frac{t_c}{2}\left(\frac{t_c}{2} - t_w\right) = \text{length of trajectory}$$

$t_a = A_{max} / W_{max} = \text{time to reach max acceleration}$

$$t_c = t_w + \sqrt{t_w^2 + \frac{4R_c}{A_{max}}} = \text{final time}$$

$$\max(v) = A_{max}\left(\frac{t_c}{2} - t_w\right) = \text{max velocity achieved}$$

Switching times = 0, t_w, $\frac{1}{2}t_c - t_w$, $\frac{1}{2}t_c + t_w$, $t_c - t_w$, t_c

Case 3: Medium movement (V_{max} is reached, A_{max} not reached)

$R_c = V_{max}(t_c - t_v) = \text{length of trajectory}$

$t_v = 2\sqrt{V_{max} / W_{max}} = \text{time to reach max velocity}$

$$t_c = \frac{R_c}{V_{max}} + t_v = \text{final time}$$

$\max(a) = W_{max} t_v / 2 = \sqrt{V_{max} W_{max}} = \text{max acceleration achieved}$

Switching times = 0, $\frac{1}{2}t_v$, t_v, $t_c - t_c$, $t_c - \frac{1}{2}t_v$, t_c

Case 4: Long movement (both V_{max} and A_{max} are reached)

$R_c = V_{max}(t_c - t_v) = \text{length of trajectory}$

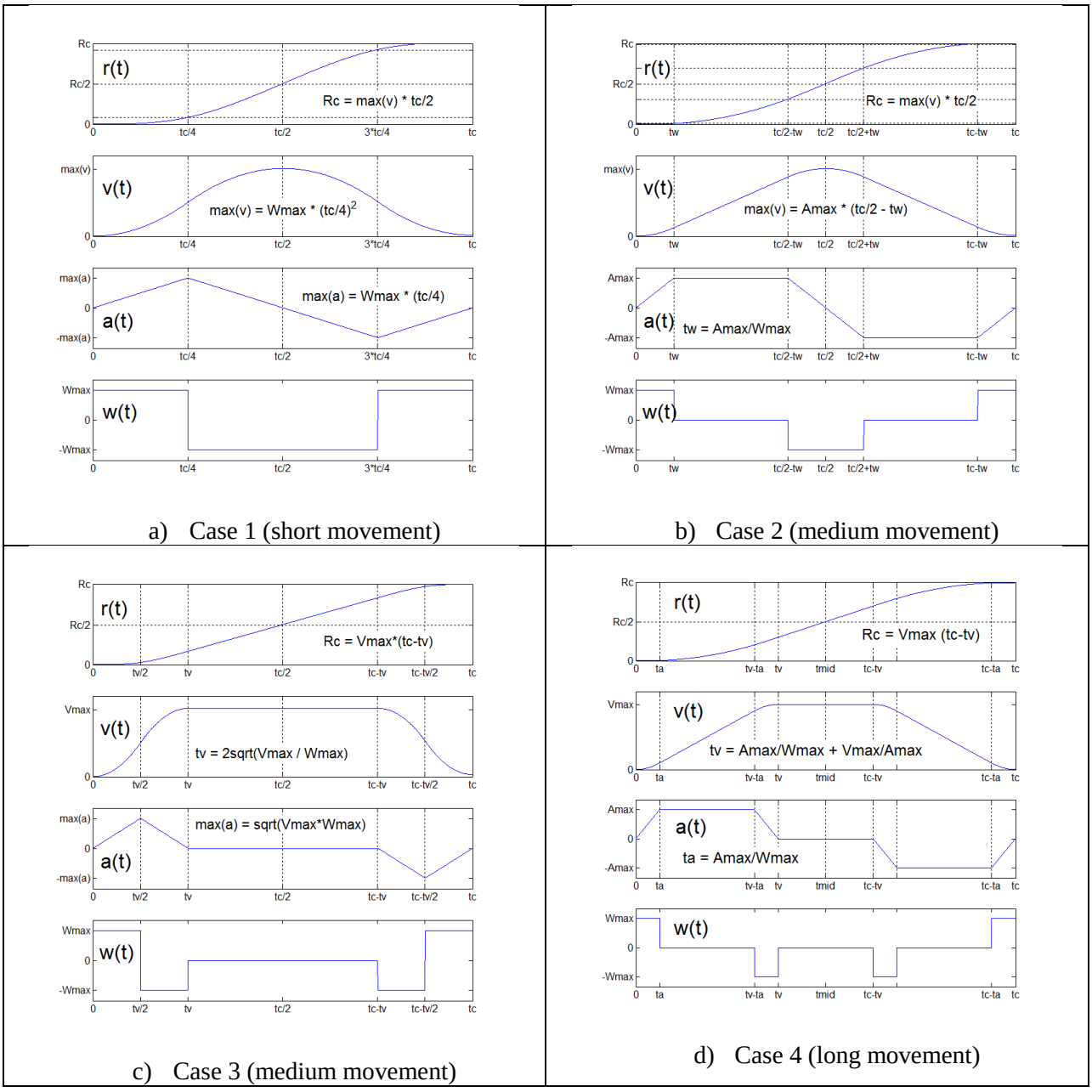
$t_a = A_{max} / W_{max} = \text{time to reach max acceleration}$

$t_v = t_a + V_{max} / A_{max} = \text{time to reach max velocity}$

$$t_c = \frac{R_c}{V_{max}} + t_v = \text{final time}$$

Switching times = 0, t_a, $t_v - t_a$, t_v, $t_c - t_v$, $t_c - t_v + t_a$, $t_c - t_a$, t_c

The four cases are plotted in Figure 3. The level of complexity is significantly increased when a jerk limit is imposed. The command following case is not attempted.



a) Case 1 (short movement)

b) Case 2 (medium movement)

c) Case 3 (medium movement)

d) Case 4 (long movement)

Figure 3: Limits on Velocity and Acceleration

IV. Limits on Snap, Crackle, and Pop

The highest derivative is limited and the move is assumed to be short enough so that none of the other derivatives reach their limit. The distance moved in time t_c with just the highest derivative limited is shown in Table 1:

Table 1: Derivatives and Length of Trajectory

Order of derivative	Name	Step size R_c reached in time t_c
1	Velocity	$R_c = V_{max} t_c$
2	Acceleration	$R_c = A_{max} t_c^2 / 4$ (divide by 2^2)
3	Jerk	$R_c = W_{max} t_c^3 / 32$ (divide by 2^5)
4	Snap	$R_c = S_{max} t_c^4 / 384$ (divide by $3 \cdot 2^7$)
5	Crackle	$R_c = C_{max} t_c^5 / 6144$ (divide by $3 \cdot 2^{11}$)
6	Pop	$R_c = P_{max} t_c^6 / 122880$ (divide by $3 \cdot 5 \cdot 2^{13}$)
>6	No-name	

Switch Times (Short Moves)

The objective is to find a trajectory where the highest derivative n is limited and switches between its maximum values and zero. No limit is placed on lower derivatives 1 to $n - 1$. The key step is to find the switch times. These times are not found analytically, but by setting a normalized version of the trajectory and then solving an optimization problem. The initial guess of the normalized switch times follows the pattern shown in Table 2.

Table 2: Switch Times (Initial Guesses)

1					0		1					
2				0		1/2		1				
3			0		1/4		3/4		1			
4		0		1/8		1/2		7/8		1		
5	0		1/16		3/8		5/8		15/16		1	
6	0	1/32		7/32		1/2		25/32		31/32		1

The initial guesses at the switch times are the average of the previous values. This pattern works exactly up to jerk (the third derivative) but then is only approximate. There is symmetry, and for derivatives $n \geq 4$ there are floor((n–1)/2) unknowns. The Matlab fsolve command is used to search for the unknown switch times based on the endpoint values of derivatives 1 to n–1 being exactly zero. When this is done:

```
1: 0  1.00000000
2: 0  0.50000000  1.00000000
3: 0  0.25000000  0.75000000  1.00000000
4: 0  0.14644661  0.50000000  0.85355339  1.00000000
5: 0  0.09549150  0.34549150  0.65450850  0.90450850  1.00000000
6: 0  0.06698730  0.25000000  0.50000000  0.75000000  0.93301270  1.00000000
7: 0  0.04951557  0.18825510  0.38873953  0.61126047  0.81174490  0.95048443  1.00000000
8: 0  0.03806023  0.14644661  0.30865828  0.50000000  0.69134172  0.85355339  0.96193977  1.0
```

Piecewise Polynomials

The simulation used to create the trajectories presents a difficulty in that the switch times of the highest derivative will usually occur in the middle of a sample period, which after several integrations will cause significant errors in the final position (several percent). Desired accuracy is usually much less than one percent. For example, consider a telescope slew of 3600 arcseconds (1 degree) which is required to settle within 50 mas, which is a required accuracy of 1 part in 72,000. In order to achieve this: 1) the normalized trajectories (with D_{max} =1 and t_c =1, where D_{max} is the size of the highest derivative) are computed as piece-wise polynomials which are analytically integrated for each interval for each derivative, 2) the piece-wise polynomials are sampled at the desired set of times, 3) the normalized trajectory is then scaled both in time and amplitude. This is all done in Matlab and takes only milliseconds.

Normalization

The normalized trajectory (computed with D_{maxn} =1 and t_{cn} = 1) achieves a normalized step size R_{cn}. The un-normalized trajectory that achieves a change R_c and t_c seconds is computed as follows:

$$r(t) = \alpha r_n(t / T_c)$$
$$(d^m / dt^m) r(t) = (\alpha / T_c^m)(d^m / dt_n^m) r_n(t / T_c)$$

Where:

$r_n(t_n) =$ normalized trajectory (reached R_{cn} in $t_n = 1$ second with $D_{maxn} = 1$)
$\alpha = R_c / R_{cn}$
$D_{max} = \alpha D_{maxn} =$ size of maximum derivative
$t_s = T_c t_{sn} =$ un-normalized switch times

Examples

The Matlab function *mintimestep* has been written for this study and returns a sampled trajectory. Several versions of the problem are available. Each works for any N_{max}, the maximum finite derivative. The routines been tested up through N_{max} = 10, well beyond pop (N_{max} = 6). Example trajectories are shown in xx.

Longer Moves

Longer moves can be implemented with limits on acceleration and velocity. It is only considered necessary place a limit on three derivatives: the max derivative, acceleration, and velocity. For example if a limit is placed on pop, then no limits are needed on jerk, snap, and crackle. A value can be determined for the maximum derivative by specifying the max acceleration and the desired time to reach max acceleration. The trajectory is then "stretched" by holding the acceleration constant at the max and min values, and if needed by further stretching the trajectory by holding the velocity constant at its max value. The details are not included here.

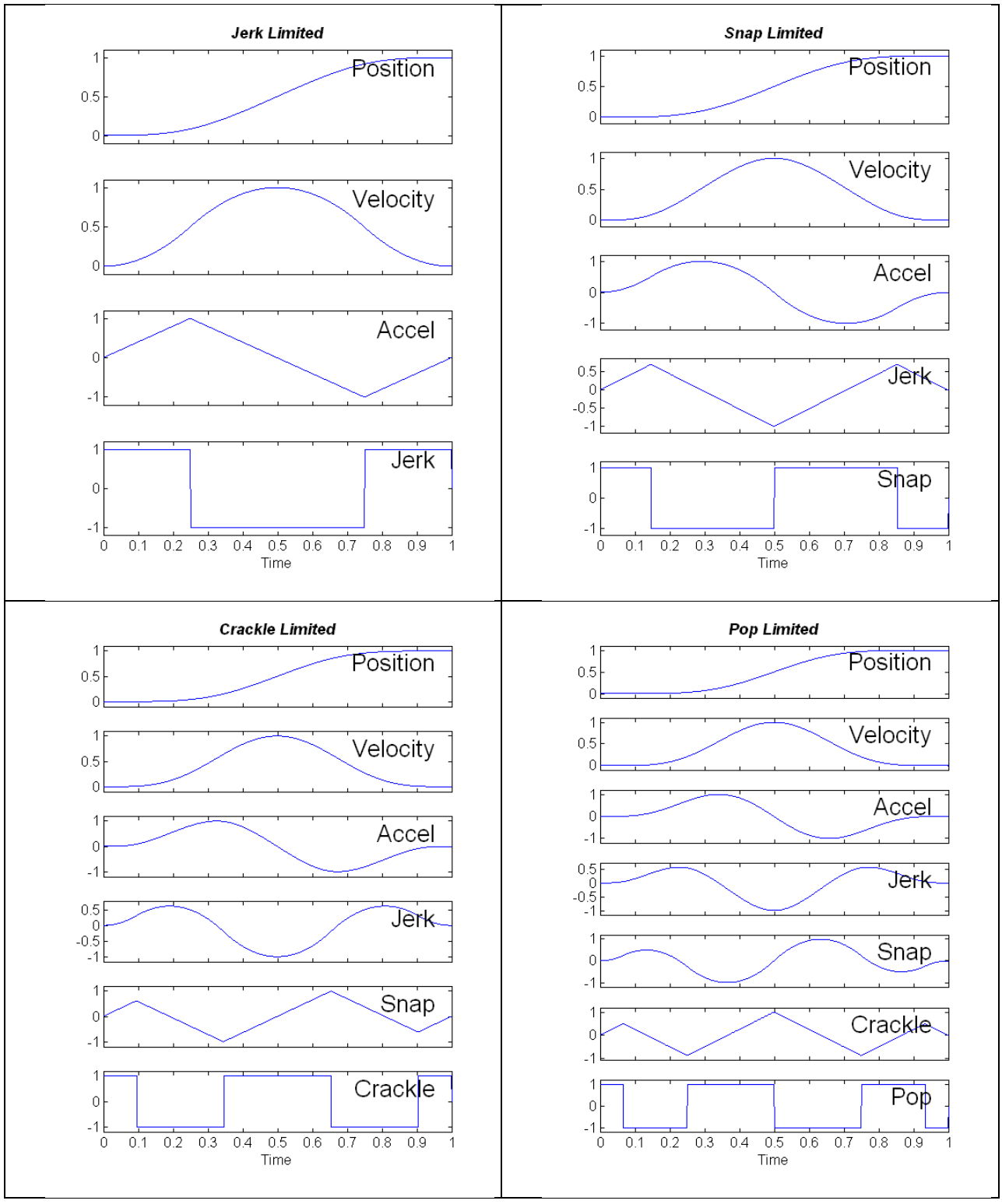


Figure 4: Snap, Crackle, and Pop Limited Responses

Note: vertical axes are normalized so that the maximum value of each derivative is unity

V. Linear Jerk Filter

A min-time command with limits on velocity and acceleration is follows by a linear low-pass filter, called a "jerk filter," as shown in Figure 5.

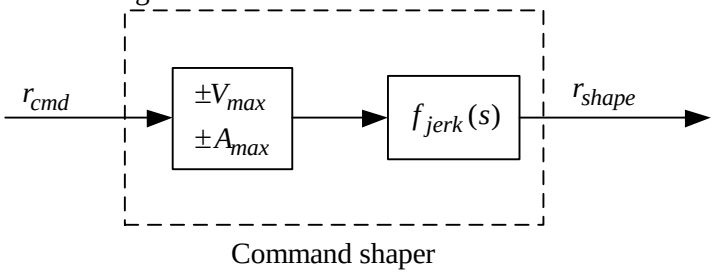


Figure 5: Command Shaper with Linear Jerk Filter

An example filter used for a telescope problem is:

$$f_{jerk}(s) = \left[\frac{2\pi f_c}{s + 2\pi f_c} \right]^6 \text{ where } f_c = 2 \text{ Hz}$$

This is a cascaded set of six each first order filters, each with a break at 2 Hz. The total rise time for the response to reach 1% of the final value is 1.04 seconds. The cascaded arrangement guarantees no overshoot.

Some comments:

- A sixth order jerk filter limits not just jerk but jerk and the next 5 derivatives.
- Additional outputs can be included for the velocity and acceleration, or higher derivatives if needed, to be used for feedforward.
- In principle the jerk filter can be placed at the end of any command shaper, for example a command shaper that limits jerk, snap, crackle, and pop. This is overkill.
- The advantage of just limiting velocity and acceleration is that a command shaper that does this can be made to operate on any command, long or short (360 degrees to milli-arcseconds), always operating in the background, and gracefully responding to mid-move changes in the commands.
- The other min-time responses are commanded moves that are of the set and forget variety. They can always be stopped by braking, but not gracefully (or as least not easily) brought to a stop with the same limits on higher derivatives.

The Matlab command *mintimestep* that has been written for this study includes an option that puts a jerk filter at the end of a limited acceleration response. A pop-limited response is compared to a linear jerk filter response (fc = 2 Hz, and order =6). The responses are very similar through acceleration, and then start to differ.

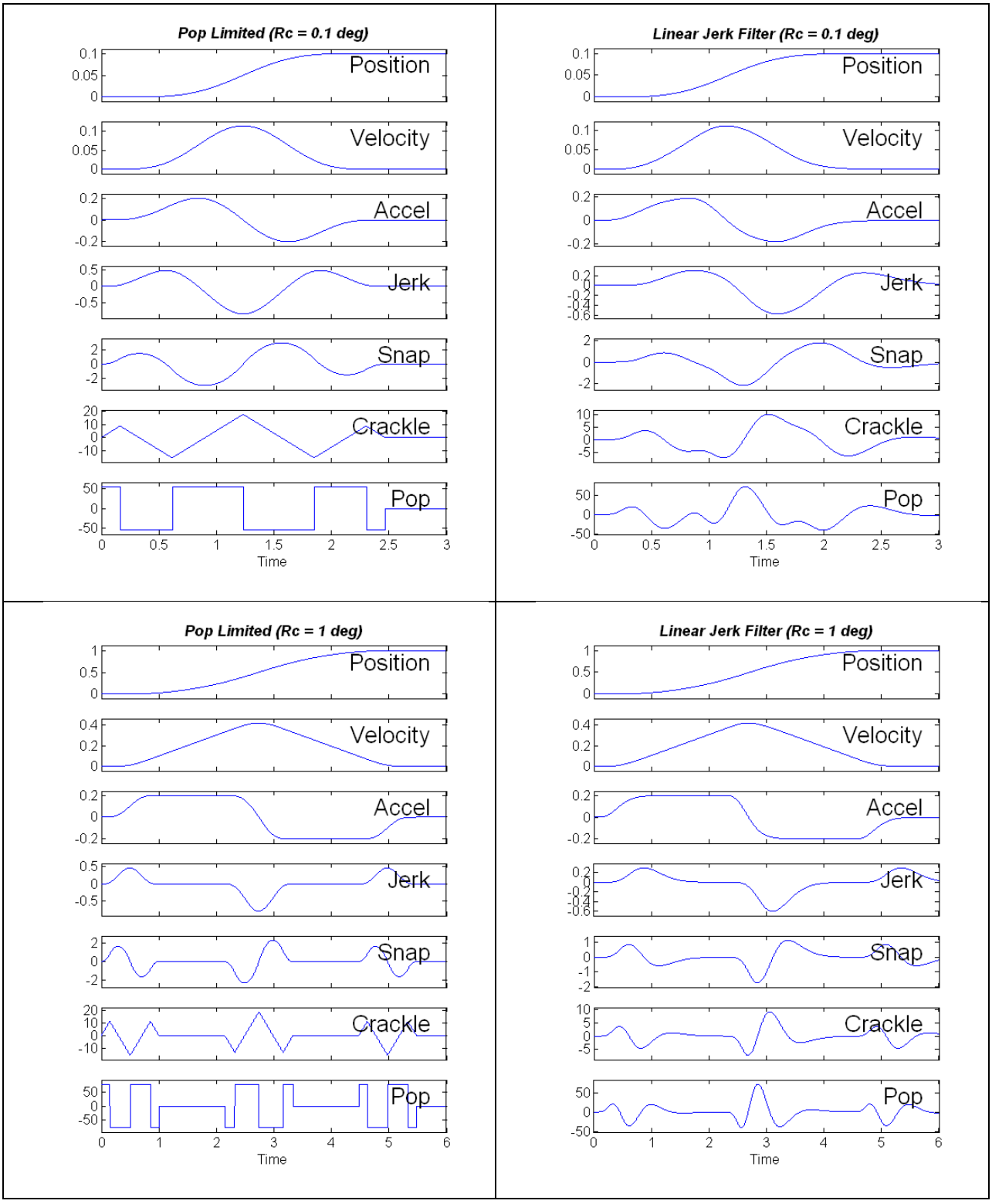


Figure 6: Comparison Pop Limited and Linear Jerk Filter Responses

Note: Amax = 0.2 deg/sec^2, Pop-limited responses have 1.0 sec rise time to Amax, Jerk filter is six cascaded 2 Hz filters with a 1.0 sec rise time

VI. Approximating Command Shaping as a Low Pass Filter

A rate limit can be approximated as the first order lag shown in Figure 7.

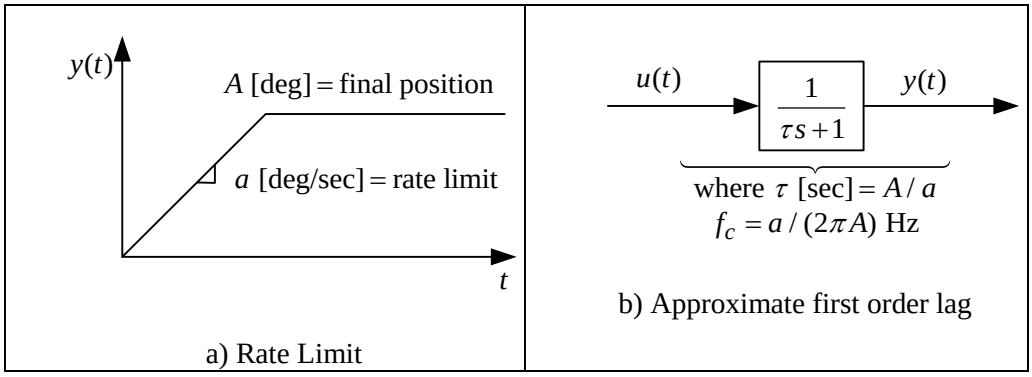


Figure 7: Rate Limit

This approximation can be formalized using describing function analysis, where the fundamental response to sinusoidal inputs produces a first order lag. It is important to note that the break frequency changes with the magnitude of the response. For larger inputs, the filter break frequency goes down.

A command shaper that limits both velocity and acceleration, or higher derivatives, can be approximated as cascaded first order filters. Set $a = R_c/T_c$ and then the break frequency is at $1/(2\pi T_c)$ Hz and the high frequency rollover is $n\times 20$ dB/decade, where n is the highest derivative that is limited.

Step response outputs of the command shapers can be used to approximate the low pass filter response. Consider a command shaper with a step input $u(t) = R_c$ for $t \geq 0$, which has the Laplace transform $u(s) = R_c/s$. The position output is $r(t)$ with Laplace transform $r(s)$, and the ratio $H(s) = r(s)/u(s)$ is the approximate low-pass filter. For a step input it follows that $H(s) = sr(s)/R_c = v(s)/R_c$, where $v(s)$ is the Laplace transfer of the velocity output $v(t)$. Therefore, use the velocity response scaled by the input magnitude as the approximation of the low-pass filter:

$$H(s) \approx v(s) / R_c, \text{ where } R_c = \text{size of step input and } v(s) = \text{velocity response}$$

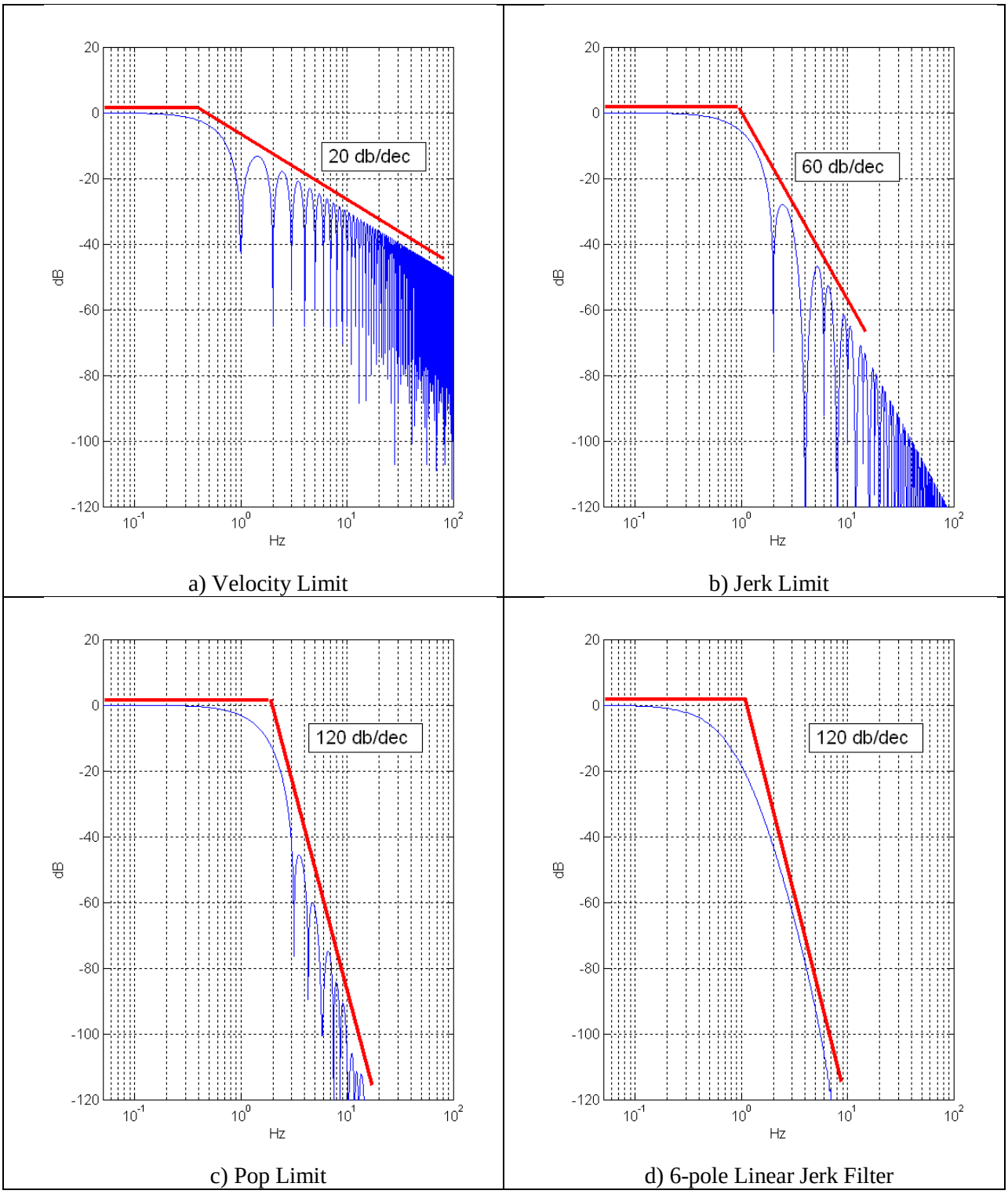
An FFT of the sampled velocity response is used to create plots of $H(s)$. The velocity response has the important added advantage that it starts and stops at zero, a condition which allows the periodic FFT to give a good approximation of the Fourier transform. Here are the details:

$v(ndt) = \text{sampled velocity for } n = 0,..., N\text{-}1 \text{ and } T = Ndt$

$V_k = \text{FFT}(v, mN)$, use mN points with $m > 1$ for added frequency resolution

$$v(s)\big|_{s=2\pi k/T} \approx V_k dt \text{ (use frequencies well below } 1/(2dt))$$

Bode plots are shown Figure 8 for different levels of command shaping. The relative sizes of the step and rise time are set to $R_c/T_c = 2\pi$ so that the filter approximation has a break at 1 Hz. The pop-limited response has about the same approximation as a 6-pole linear jerk filter.



a) Velocity Limit

b) Jerk Limit

c) Pop Limit

d) 6-pole Linear Jerk Filter

Figure 8: Approximate Low-pass Filters

VII. FeedForward

The feedback and feedforward architecture is in Figure 9. The position command θ_{cmd} is shaped by a nonlinear circuit that limits velocity and acceleration and then is lagged by a linear circuit $f_{lag}(s)$ to reduce jerk, resulting in θ_{shape}. The shaped command enters in two locations: as a position command and through a feedforward filter $f(s)$. The output of the feedforward adds to the torque command.

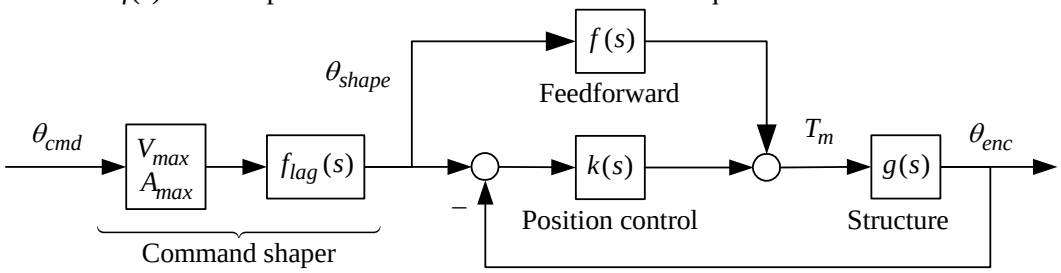


Figure 9: Ideal Feedforward

The closed loop response from the shaped command is:

$$\frac{\theta_{enc}}{\theta_{shape}} = \frac{g(k+f)}{1+gk} = 1 \text{ if } f = g^{-1}$$

The ideal feedfoward inverts the plant so that the closed loop response is unity. This result is true for any controller. Velocity and acceleration limits are maintained by shaping the command to stay within limits. Define a structure that is pure inertia, where:

$$g(s) = \frac{1}{Js^2}\ ,\ f(s) = \hat{J}s^2$$

The feedfoward has one gain term $\hat{J}$, which ideally is equal to J, but might include an error. Implement the command shaper to output both the command and its second derivative, thereby avoiding noise issues.

The PID controller has three gains designed to meet three specification: 1) unit magnitude crossover of 1 Hz, 2) phase margin 45 degrees, and 3) lower gain margin of 12 dB (this much reduction destabilizes):

$$k(s) = k_p + \tfrac{1}{s}k_i + sk_r, \text{ where } k_r = 11.09J,\ k_i = 6.286k_rJ,\ k_p = 3.187k_rJ$$

The $f_{lag}(s)$ filter in the command shaper is six cascaded single-pole filters each at 2 Hz. The total time is 1 second:

$$f_{jerk}(s) = \left(\frac{\omega_{cs}}{s+\omega_{cs}}\right)^6 \text{ where } \omega_{cs} = 2\pi f_{cs},\ f_{cs} = 2 \text{ Hz}$$

A graphical survey is in Figure 10. Comments:

- The loop transfer function $gk(s)$ verifies the control system specifications.
- The closed loop unit step response has 40% overshoot. A zero velocity system such as this will always have overshoot. High overshoot is desirable because it increases low frequency gain and hence reduces the tracking error. The overshoot, however is not desirable for short moves, and some help in the form of command shaping or feedfoward is needed for short moves.

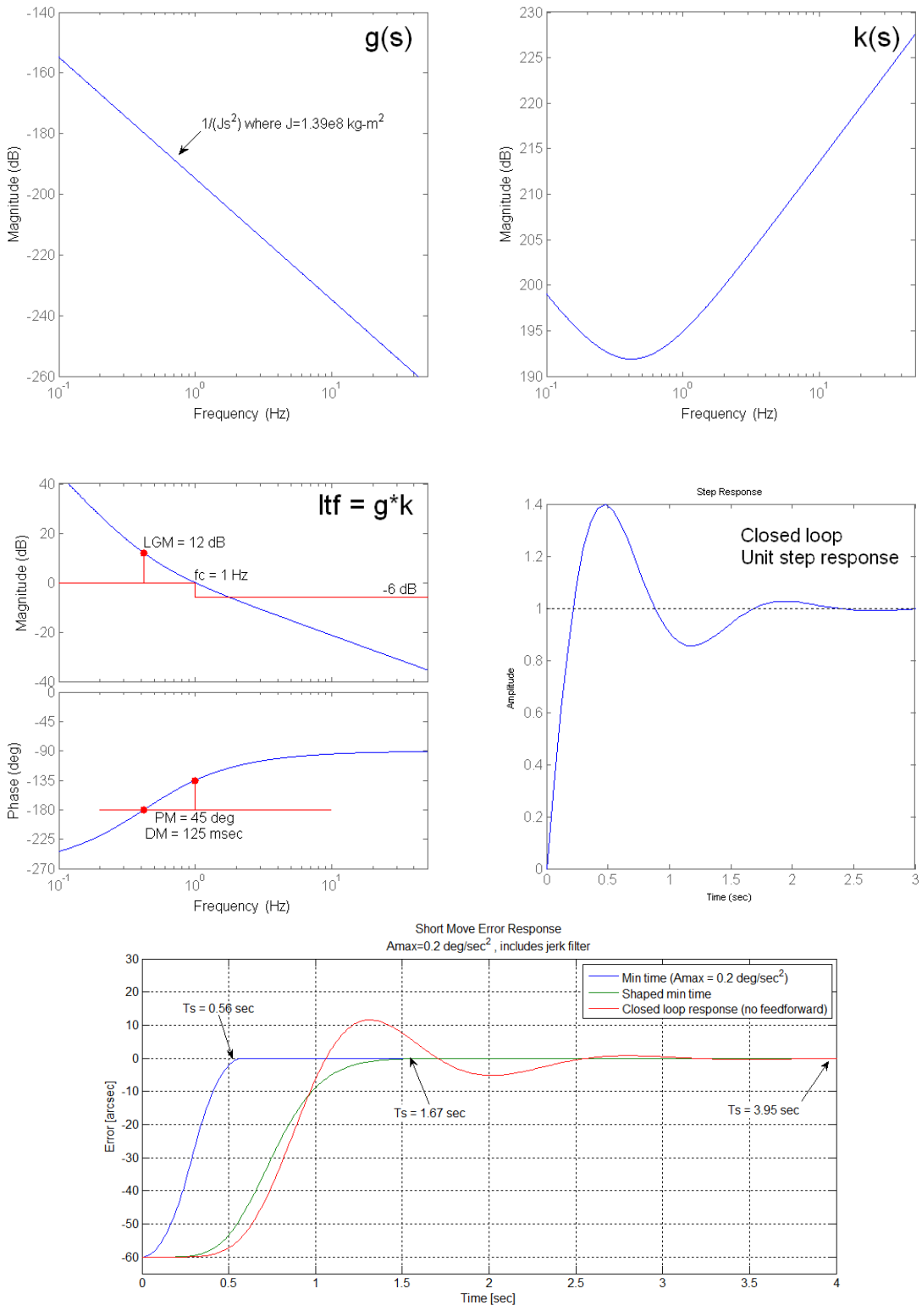


Figure 10: Ideal System Control Survey

- Three 60 arcsec response are shown: 1) just the acceleration limited response, 2) the same through the jerk limiting filter, and 3) the shaped response through the system with no feedforward. The second response is for the ideal case exactly the same as the system response with feedforward. The third response is what happens when feedforward is turned off, and again demonstrates the need for feedforward.

VIII. Conclusions

Methods are described to create min time trajectories with limits on selected derivatives. There are two approaches: 1) Exactly limit a high order derivative, acceleration, and velocity, and 2) Exactly limit acceleration and velocity followed by a linear filter. Either method works well for moves between stationary points. The second method is recommended for command following applications.