Two Mitigation Strategies For Motion System Limits

Chris Schwarz
National Advanced Driving Simulator
The University of Iowa
2401 Oakdale Blvd
Iowa City, IA 52242

Abstract
Limited workspace is a challenge for all motion-based simulators, whether they are large excursion systems, like the NADS, or smaller simulators utilizing only Stewart Platforms. Two approaches for addressing this challenge are nonlinear washout scaling and software displacement limiting. This paper presents new algorithms developed for these approaches. The nonlinear scaling method makes use of the cubic Hermite interpolation polynomial to smooth the corner in the scaled output at the limit. A software displacement limiting method is introduced that generates control signals in the simulator frame of reference. As a result, unwanted acceleration artifacts caused by unbalanced limiting of the actuators can be avoided. The methods are described, and off-line simulation results using the new displacement limiting method are presented.
Introduction

In driving and flight simulators, it is desirable to minimize the bumps that result from hitting the workspace limits of the motion base. This is addressed in the motion drive algorithm (MDA) through scaling the input signals, and in software limiting schemes that can mitigate these occurrences by gently slowing the offending actuator down and keeping it away from the stroke limit. Both strategies can cause bumps that produce unacceptable false motion cues to the driver. As a result, simulator operators often have to be extremely conservative when setting the MDA parameters.

The goals of the methods described in this paper are to provide ways of mitigating the adverse effects of hitting motion limits, allowing the simulator operator to relax the constraints on the motion somewhat. As a general rule, motion limits should still be avoided; however, now the operator can tune the system for the average driver, not the worst case.

The simplest method of constraining the motion commands from the MDA block is to provide scaling factors for each degree of freedom (DOF) and clip the signal if a specified limit is reached [1,3]. One can imagine that the simulator operator will want to avoid this limit at all costs, because it will cause a bump and wipe out all cues above the limit. A variation is to replace the clipping limit with another linear scale region of reduced slope [3]. A polynomial-based nonlinear scaling method was introduced by Telban [3]. The third degree polynomial used there is able to accomplish the same things as the cubic Hermite interpolation polynomial, but the parameter set is more complicated. Moreover, there are no guidelines for preventing the user from choosing 'bad' parameters. The proposed nonlinear scaling method addresses these shortcomings.

The second strategy regards the particular class of workspace limits in the Stewart Platform caused by the actuators hitting their stroke limits. An actuator limiting method is described by Reid and Nahon [2] for the UTIAS flight simulator. This algorithm effectively protects the hardware from hitting hard limits, but still creates a significant bump for the driver.

Hermite Nonlinear Scaling Method

A nonlinear scaling algorithm has been developed that makes use of the cubic Hermite interpolation polynomial [5]. This interpolation method allows the initial and final function values to be selected, as well as the initial and final function derivatives. A set of basis functions for the cubic Hermite polynomial is given by

\[
\begin{align*}
    h_1(x) &= \frac{(x-x_1)^3(3x_0-x_1-2x)}{(x_0-x_1)^3} \\
    h_2(x) &= \frac{(x-x_0)(x-x_1)^2}{(x_0-x_1)^2} \\
    h_3(x) &= \frac{(x-x_0)^2(3x_1-x_0-2x)}{(x_1-x_0)^3} \\
    h_4(x) &= \frac{(x-x_1)(x-x_0)^2}{(x_1-x_0)^2}
\end{align*}
\]

Eq 1
where \( x_0 \) and \( x_f \) denote the initial and final inputs for the interpolation. Selecting the initial and final function values as \( s_0 \) and \( s_f \), and the initial and final function derivatives as \( d_0 \) and \( d_f \) respectively, the interpolation polynomial is written as

\[
s(x) = s_0 h_1(x) + d_0 h_2(x) + s_f h_3(x) + d_f h_4(x).
\]

This polynomial is chosen to be the scaling function in the MDA. However, certain conditions should be met to be a good scaling function. First, the derivative of the scaling function at the initial point is assumed to be zero, since a constant scaling is likely to be used before some upper boundary is encountered. At the other end, a hard limit must be set that the output of the scaled signal will not exceed. This was accomplished by a simple clipping function in the linear scaling approach, giving rise to a potential sharp corner in the scaled output. The Hermite scaling polynomial allows the scaled output to gradually and smoothly attain the limit. Therefore, the upper boundary of the scaled output should equal the limit, denoted by \( L \); and the derivative of the scaling function at the boundary should be negative and perfectly counteract the positive slope of the input so that the output is flat at the final boundary. Finally, the scaling function should be positive for all values of the input, positive or negative. The key parameters of the interpolation function may be selected to satisfy the above conditions by the following assignments:

\[
x_0 = 0, \quad x_f = C \times L, \quad d_0 = 0, \quad d_f = -\frac{s_f}{x_f}, \quad s_0 = \text{free}, \quad s_f = \frac{L}{x_f}.
\]

The independent parameters that determine the shape of the scaling function are \( s_0, L, \) and \( C \). Then the scaled signal output, for a generic value of the \( x_0 \) parameter, is simply expressed by

\[
y = \begin{cases} 
  s_0 x, & 0 \leq x < x_0 \\
  s(|x|)x, & x_0 \leq x < x_f \\
  L, & x \geq x_f 
\end{cases}
\]

The initial scaling factor, \( s_0 \), is a free parameter. The limit, \( L \), sets the maximum value of the scaled signal, and is important to prevent motion aborts. The parameter, \( C \), is more mysterious, but has an important effect on the shape of the scaling function, and thus the scaled output. Notice in Eq 3, that the parameter, \( C \), directly determines the location of the final boundary point, \( x_f \). The separation between \( x_0 \) and \( x_f \) has a significant effect on the shape of the scaling curve.

Consider three possible curves shown in Figure 1, corresponding to values of 0.5, 2, and 5 for \( C \). The initial scale factor, \( s_f \), was arbitrarily chosen to be 0.9 for this example. The first curve rises very rapidly to the limit of one because the scaling peaks well above unity. The second curve rises more gently to the limit and then is flat. The third curve is peculiar in that the output actually exceeds the limit before dropping back down to it at the final boundary of the interpolating region. The first and third cases are undesirable since we don’t want the scaling to exceed the user’s initial scale factor; and we don’t want the output to exceed the user’s specified limit. We wish to select an optimal value.
for $C$ and make it invisible to the operator, leaving only the initial scale factor and the limit as tunable parameters.

![Figure 1 Three Scaling Functions and Resulting Outputs](image)

What is the optimal value of $C$, in this application? The purpose of using the nonlinear scaling algorithm is to avoid motion bumps, i.e. maximize smoothness. The use of the Hermite polynomial allows the endpoints and first derivatives of the scaling function to be selected. There may, however, be a discontinuity in the second derivative of the scaled output at the final boundary point, $x_1$. It would therefore be optimal, from a smoothness standpoint, to use $C$ to set the second derivative of the scaled output to zero at the final boundary. This is accomplished by the assignment $C = 2/s_0$ which can be verified by directly evaluating the second derivative of Eq 4 at $x = x_1$, which is derived from the assignment of $C$ to be $x_1 = 2L/s_0$. Note that in the previous discussion, the derivative is always calculated with respect to the input signal, $x$, not time.

We state, without proof, the main result of this section. The assignment, $C = 2/s_0$, along with the parameter selections in Eq 3, always avoids the two ‘bad’ curves in Figure 1. Then we are left with two independent parameters, the initial scaling factor, $s_0$, and the output limit, $L$, exactly the same parameters used in the linear scaling with clipping method. Moreover, the new method provides smooth limiting of the scaled outputs.

**Stewart Platform Actuator Limiting**

The Stewart Platform is the standard choice for driving and flight simulator motion bases. The parallel nature of the mechanism provides for a larger load-bearing capacity. A comprehensive review of this device is provided by Dasgupta and Mruthyunjaya [4]. The inverse kinematics of the Stewart platform are much more easily obtained than the
forward kinematics map, as is the case with all parallel actuators. On the other hand, the inverse Jacobian is computed with little difficulty [6,7]; and it can be inverted with a modest computational effort. Therefore, velocity signals are favored over position signals in the proposed algorithm.

The UTIAS displacement limiting scheme [2] modifies each actuator independently to prevent their respective limits from being hit. This approach completely neglects what's happening in the simulator frame of reference. Unfortunately, there can be unwanted artifacts in the simulator frame that are perceivable to the driver; and it would be beneficial to have a way of reducing or eliminating them. We present a control scheme that takes advantage of the simulator frame of reference to accomplish this objective.

The block diagram in Figure 2 shows a PI control loop that can be used to enforce a desired actuator velocity trajectory, which is generated here as a limited version of the actual velocity. Velocity is tracked rather than position since we are converting between frames of reference using velocity signals. Observe that the control signals are generated in the simulator frame, in deference to our desire to directly control the simulator velocity signals. The vehicle head point velocities are injected as feed-forward signals after the PI controller, and a weighting block (to be described later).

\[ u = xJ x + \sum_{i=1}^{n} (s_i - \text{sat}(s_i)) \]

Figure 2  NADS Actuator Displacement Limit Control Loop

**Limiting Algorithm**

A nonlinear gain, \( K \), is inserted into the feedback path to multiply the actual velocities, resulting in the desired ones, \( l^* = K \times l \). The limiting algorithm that underlies the nonlinear gain was arrived at using the same first principles that the UTIAS limiting algorithm uses [2]. If a maximum acceleration of the actuator during limiting is specified, then the position threshold at which deceleration must begin is given by the relation

\[ l_{SW} = l_{\text{max}} - \frac{j^2}{2a_{\text{max}}} . \]

Eq 5

A more conservative linear switching line may be used instead of Eq 5, and given by
where $T_{SW} = V_{lim}/2A_{lim}$.

The resulting nonlinear gain is a function of the actuator positions, velocities, and accelerations. Moreover, it is filtered, and therefore also a function of the Laplace operator, $s$. The expression for the gain is given by

$$K(l,i,i,s) = \frac{s}{s + 50} \sin\left(\frac{\pi}{2} k(l,i,i)\right).$$

where the argument of the sine is a value between zero and one, calculated as

$$k(l,i,i) = \begin{cases} 
1 & |l| \leq l_{lim} - T_{SW} |i| \\
\frac{l_{lim} - |i|}{T_{SW} |i|} & |l| > l_{lim} - T_{SW} |i| \\
0 & |l| = 0 \text{ or } k < 0
\end{cases}$$

**Eliminating Artifacts**

Now, consider the problem of eliminating table frame artifacts caused by the actuator limiting. These artifacts arise because the limiting algorithm operates on each actuator independently. As a result, an extreme input to, for example, the lateral DOF can generate perceivable transients in pitch and yaw as well. How can these unwanted transients be corrected for? A simple scheme is proposed which uses the normalized magnitudes of the inputs to weight the controlled error signals. In the event of an artifact, there will be error signals present in multiple DOFs after the transformation to the simulator frame. However, if 90% of the input signal is coming from one DOF, the weighting scheme will reduce the artifact error signals to around 10% of their original value. If the input is purely in one DOF (not that common in practice), the artifacts will be completely eliminated. The weighting formula is given by

$$W(u,s) = \frac{s}{s + 50} \frac{u^2}{\|u\|^2}.$$

Observe that it has also been filtered. This helps keep the weighting coefficients flat as the velocity vector passes through zero, a singularity point in the above formula.

**Results and Discussion**

The new NADS limiting algorithm was compared with the existing one, which was based on an approach documented for the UTIAS flight simulator [2]. As a reference, an unlimited response was also recorded, assuming that there were no actual hardware limits to prevent excessive actuator stroke.
A sinusoidal input was injected into the Y DOF of the Stewart Platform to test the limiting methods. No MDA, or other scaling mechanism was present between the input and the limiting block. The input sinusoid was given an amplitude of 0.2 G and a frequency of 2 rad/sec. The figures below show the specific forces and rotational velocities of the Stewart Platform table, the actuator accelerations, and the most severe actuator positions.

First observe the actuator accelerations in Figure 3. We see immediately that the most severe limiting is done on the actuators numbered three and four. The accelerations for these two actuators also highlight the difference in the operation of the old and new limiting methods. Because of the closed-loop nature of the PI controller, the amount of distortion in the acceleration is greatly reduced, even though the underlying limiting algorithms are similar.

Notice that the new method modifies actuator accelerations which are left untouched by the original method. In fact, an actuator may not be anywhere near a limiting zone; but the new method does what it needs to do to prevent artifacts in the simulator frame.

The position signals for actuators number three and four are shown in Figure 4. A comparison between the position signals resulting from the two methods reveals two interesting things. First, the limited position tends to drift from cycle to cycle in the new method due to the integral gain in the controller. This 'self-centering' behavior may be desirable in the translational DOFs, but is not in the rotational ones. Second, the position
of actuator four momentarily exceeds the actuator position limit of 18 inches during the first cycle. So, the new method does not guarantee hard limits like the original does. Rather, proper tuning of the controller parameters is required to ensure the desired functionality.

The bottom line for motion analysis is what happens in the simulator frame; because this is what is perceived by the driver. The specific forces and rotational velocities generated in the table frame are shown in Figure 5 and Figure 6. First examine the driven degree of freedom, the Y axis. The new and original methods have the same characteristics that were observed in the actuator acceleration plots. We make the point here though, that the new method provides a much smoother limiting profile. Said another way, it is clear that the original method causes much more jerk that the driver would certainly feel.

Additionally, this is the first time we see the artifacts that have been cited as the motivation for this research. The original limiting algorithm generates extraneous accelerations in all the other channels. The X channel artifact peaks around 50 mG, which is certainly detectable to the human vestibular system. The Z channel artifact peaks at above 20 mG, which may be detectable. Additionally, the amplitude of the rotations in the pitch and yaw DOFs exceed 2 degrees/second, which is about the threshold of human detection. The new limiting method is able to completely prevent these extraneous accelerations and rotations.

**Figure 5** Simulator Specific Forces  
**Figure 6** Simulator Rotational Velocities

**Conclusions**

Two mitigation strategies for motion limits in Stewart Platforms were presented. The first was a nonlinear scaling method for use in motion drive algorithms. The scaling method is based on the cubic Hermite interpolation polynomial. It is an attractive alternative to linear scaling with clipping because the sharp corner at the limit is smoothed out, and the required parameters are exactly the same as those used for linear
scaling. Because the continuity of the first and second derivatives of the scaled output is ensured into the limiting region, the basic shape of the input signal is not changed.

The second was a new actuator displacement limiting method. The new method has some advantages over the existing algorithm. The severity of the distortion in actuator accelerations is significantly reduced with the new method. Moreover, unwanted artifacts in the simulator frame of reference caused by unbalanced actuator limiting are greatly reduced, and potentially eliminated. The new method does require some tuning of the controller parameters to obtain the functionality desired by the user. Liberal use of the integral gain will introduce a self-centering property that biases the signal away from the motion limits. Therefore, integral gains must be used conservatively in the roll and pitch degrees of freedom to prevent an unwanted tilt bias.

These methods offer additional flexibility to the users of driving and flight simulators, who are worried about the adverse effects of hitting motion limits on the comfort of the driver, and the validity of the simulation results. The experiment designer can afford to be, if not aggressive, at least not quite so conservative, when setting up the MDA (washout) parameters.

References


