Rapid Tuning of the Classical Motion Cueing Algorithm

Richard Romano [1], Ehsan Sadraei [1], Gustav Markkula [1]

(1) University of Leeds, Leeds LS2 9JT UK, e-mail: {r.romano, tses, g.markkula}@leeds.ac.uk

Abstract: Numerous non-linear Motion Cueing Algorithms (MCA) have been proposed and developed over the years. However, researchers quite often return to using the Classical Algorithm as a baseline case to compare against their algorithm or as a method to yield consistent phase and gain cues. The current paper presents an updated layout of the Classical Algorithm for ground vehicles that supports rapid parameter optimisation. A simple automated optimisation method is developed. The Classical Algorithm is first tuned and compared with an MPC algorithm and then with a set of objective performance parameters.

Keywords: driving simulation, motion cueing

Introduction

Numerous non-linear Motion Cueing Algorithms (MCA) have been proposed and developed over the years. However, researchers quite often return to using the Classical Algorithm as a baseline case to compare against their algorithm or as a method to yield consistent phase and gain cues [AdHP07]. Quite often the Classical Algorithm is derived from Reid and Nahon [ReNa85] which in turn is based on developments from Conrad and Schmidt [CoSc70, ScCC71] and Sinacori [Sina73]. The current paper will present an updated layout of the Classical Algorithm for ground vehicles that supports rapid parameter optimisation. The goal is an algorithm that can be used to provide a fair and simple comparison between the Classical Algorithm and more modern MCAs.

Algorithm Layout

Typically in the Classical Algorithm for flight simulators, the vehicle angular velocities are high-pass filtered to limit the total motion required and integrated to calculate the motion base orientation [ReNa85]. This is used in flight simulation because large angular motions are possible and because coordinated turns result in a total specific force that is aligned with the aircraft cabin. If the total specific force does not align with the vehicle cabin, feeding the vehicle angles in directly has shown to provide better cueing [GrHo03]. Therefore for ground vehicles when roll and pitch angles are small they can be used directly and only the yaw angle needs to be filtered (unless an unlimited yaw ring is available). The orientation of the motion base is based on the concatenation of two transforms. The first transformation, T_{TI}, moves from the motion base inertial reference frame (I) using the Euler angles θ_{TILT} and ϕ_{TILT} to the tilt coordination reference frame (T). Because the tilt coordination is performed for roll and pitch independently in the inertial reference frame, the tilt angles are extrinsic angles. The second transformation, T_{ST}, moves from the tilt coordination reference frame to the upper platform reference frame (S) using the Euler angle vector: $[\phi \quad \theta \quad \psi']$. The updated Classical Algorithm is shown in Fig. 1 while the symbols used are defined in Tab. 1.

Table 1. List of Symbols

Symbol	Meaning
a_S a_T	Scaled vehicle acceleration in simulator reference frame and in tilt frame
a_P $a_{P\,ERR}$	Total acceleration cue and its error
$\hat{a}_T$	Limited acceleration in tilt reference frame
S_I	Simulator position in inertial frame
θ_{TILT} ϕ_{TILT}	Pitch, roll tilt coordination angles
ϕ θ	Scaled vehicle roll and pitch Euler angles
ψ'	Filtered yaw Euler angle
$\hat{\dot{\psi}}$	Scaled and limited vehicle yaw Euler rate
ω	Filter break frequency
T_{TI}	Transformation from inertial to tilt frame
T_{ST}	Transformation from tilt to simulator frame
g	Gravity
$[]_{MAX}$	Maximum value
e	Euler's number

Reviewing Fig. 1, the scaled vehicle accelerations flow from the left, are transformed into the tilt coordination reference frame and limited. Since vehicle roll and pitch are used directly, specific force calculations are not required. Eq. 1 and 2, define how the simulator position and tilt coordination angles are calculated from the limited accelerations using

second order Butterworth filters (Y, Z, and ϕ_{TILT} are calculated similarly). Eq. 3, defines how the yaw

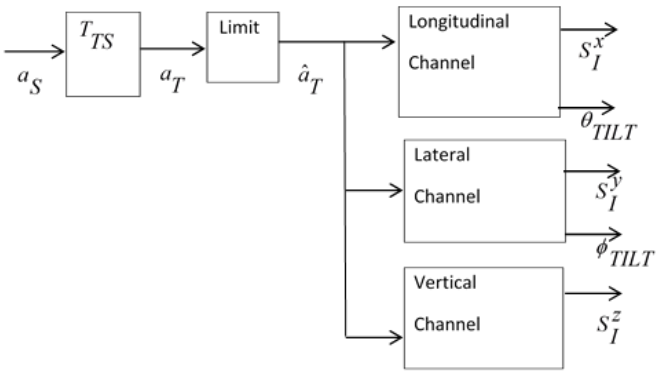


Figure 1. Classical Algorithm for Ground Vehicle

angle ψ' is calculated using a first order Butterworth filter. A first order filter is used for yaw rate to reduce the vestibular-ocular mismatch during a constant rate turning maneuver.

$$\frac{S_I^x}{\hat{a}_T^x} = \frac{1}{s^2 + 2\omega_h^x s + \omega_h^{x2}} \tag{1}$$

$$\frac{\theta_{TILT}}{\hat{a}_T^x} = \frac{1}{g}\frac{\omega_l^{\theta 2}}{s^2 + 2\omega_l^{\theta} s + \omega_l^{\theta 2}} \tag{2}$$

$$\frac{\psi'}{\hat{\dot{\psi}}} = \frac{1}{s + \omega_h^{\psi}} \tag{3}$$

There are three important differences between the algorithm in Fig. 1 and the algorithm from Reid and Nahon that make rapid tuning possible: the tilt coordination channels are calculated in the inertial coordinate system rather than the local coordinate system (similar to [Roma99]), acceleration limiting is performed in the inertial coordinate system right before the input to Eq. 1 and 2, and finally the acceleration fed to the high pass filters does not include any feedback from the tilt coordination, which makes the system response second order (note that this creates a small linearization error).

Algorithm Tuning

Since the accelerations are limited just before the input to Eq. 1 and 2, the final value theorem can be used to calculate the maximum position and tilt angle of the simulator as given in Eq. 4 and 5 and the maximum tilt rate can be calculated as given in Eq. 6 using the derivative of the step input response of Eq. 2.

$$\omega_h^x = \sqrt{\frac{\hat{a}_{T\max}^x}{S_{I\max}^x}} \tag{4}$$

$$\theta_{TILT\max} = \frac{\hat{a}_{T\max}^x}{g} \tag{5}$$

$$\omega_l^{\theta} = \frac{\dot{\theta}_{TILT\max} g e}{\hat{a}_{T\max}^x} \tag{6}$$

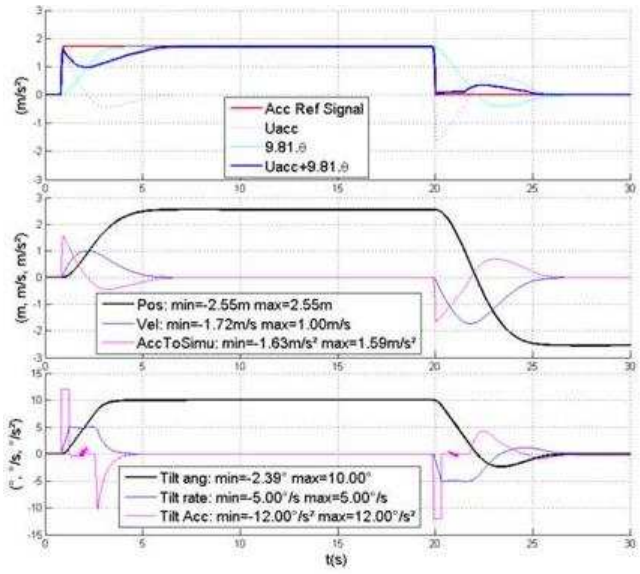
The step input response of the total acceleration cue, Eq. 7, to an input of size $\hat{a}_{T\,MAX}$ can be used to calculate the maximum acceleration error a_{PSAG}.

$$a_p^x = \left[\frac{s^2}{s^2 + 2\omega_h^x s + \omega_h^{x2}} + \frac{\omega_l^{\theta 2}}{s^2 + 2\omega_l^{\theta} s + \omega_l^{\theta 2}}\right]\hat{a}_T \tag{7}$$

For the yaw rate channel Eq. 8, again the final value theorem can be can be used to calculate the maximum angle.

$$\omega_h^{\psi} = \frac{\hat{\dot{\psi}}_{\max}}{\psi'_{\max}} \tag{8}$$

Any number of optimisations can be performed to rapidly calculate the linear filter break frequencies with tilt coordination using Eq. 4, 6, and 7. For this example the MPC response from Fig. 4 in [FaKe14] will be used to define constraints to tune the Classical algorithm. The MPC response is reproduced in Fig. 2. Reviewing Fig. 2 one can see that the total acceleration (in blue) sags from 1.72 m/s to 1 m/s while the linear acceleration is filtered out and the tilt angle ramps in.



Time (sec)

Figure 2. MPC Response (taken from [FaKe14] Fig. 4)

The optimisation of the Classical algorithm will focus on minimizing tilt rate since tilt rate performance is typically the most observable difference between non-linear algorithms and the Classical algorithm.

Minimize	$\dot{\theta}_{\text{TILT max}}$
Subject to	$\hat{a}^{x}_{T\,max} \geq 1.72$ m/s^2
	$S^{x}_{I\,max} \leq 2.6$ meters
	$a_{P\,SAG} \leq 0.72$ m/s/s
Yields	$\dot{\theta}_{\text{TILT max}}$ = 6.3 deg/sec
	ω_l^{θ} =1.705 rad/sec
	ω_h^{x} =0.8134 rad/sec

Fig. 3 shows the response of the Classical Algorithm to a 1.72 m/s^2 acceleration for 20 seconds with the optimised parameters. Comparing Fig. 2 and 3, the MPC response, to the same input, results in a similar sag at onset but limits the tilt rate to 5 deg/sec while the Classical requires 6.3 deg/sec. The MPC algorithm has a much smaller false cue at the end of the manoeuvre by using the complete negative excursion.

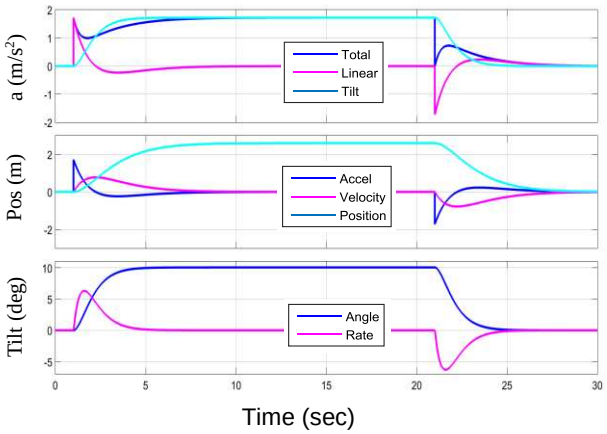


Figure 3. Classical Algorithm Tuned Response (Org)

Fischer et al. [FiSS16] introduced the notion of an Objective Motion Cueing Test (OMCT) for driving simulators based on work by Advani et al. for flight simulators [AdHP07]. The Bode plot of the Classical Algorithm, as initially tuned, is shown in Fig. 4 and is labelled as 'Org'. The red line in the figure shows the region below which Advani considers the open loop response of a motion base coupled with a motion cueing algorithm to have 'Low Fidelity' for a flight simulator [HoAd16]. The original tuning of the Classical Algorithm far exceeds the required performance identified by Advani (although we must keep in mind that the motion base performance must be added to calculate the total system response).

From Advani's amplitude guidelines in Fig. 4, one can come up with a new more general optimisation strategy. First the sagging must be less than 80% across all frequencies greater than 0.1 Hz. Reviewing additional work by Advani [HoAd16] the phase error should be less than 180 degrees. For the Classical Algorithm in Fig. 1, if the break frequency of the low pass filter is less than the break frequency of the high pass filter then there will be a large phase error at higher frequencies.

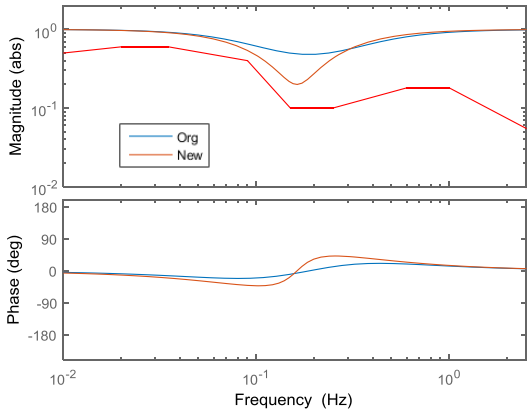


Figure 4. Classical Algorithm Bode Response

Also the break frequency of the low pass filter should be between 0.5 rad/sec and 2 rad/sec to meet the cross coupling requirements between surge and pitch as defined by Hosman et al. [HoAd16]. Therefore the new optimisation is (assuming a desired tilt rate of 5 deg/sec and the same motion base size as before):

Maximize	$\hat{a}^{x}_{T\,max}$
Subject to	$\dot{\theta}_{\text{TILT max}} \leq 5$ deg/sec
	$S^{x}_{I\,max} \leq 2.6$ meters
	$a_{P\,SAG} \leq 80$ % magnitude
	$\omega_l^{\theta} > \omega_h^{x}$
	$0.5 < \omega_l^{\theta} < 2$ rad/sec
Yields	$\hat{a}^{x}_{T\,max} = 1.92$ m/s^2
	ω_l^{θ} =1.212 rad/sec
	ω_h^{x} =0.8593 rad/sec

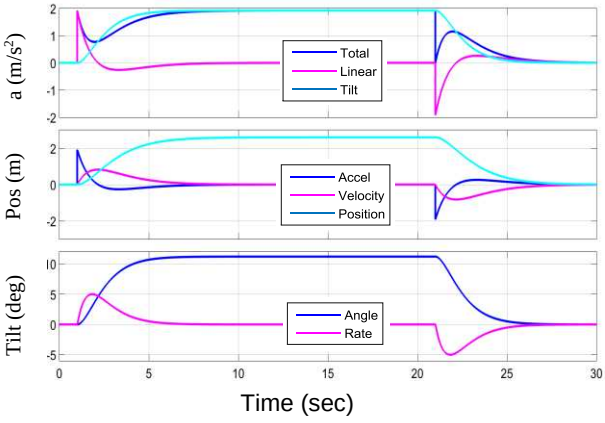


Figure 5. Classical Algorithm Tuned Response (New)

The response of the algorithm to the step input is given in Fig. 5 and the Bode response is given in Fig. 4 and labelled as 'New'. As can be seen in Fig. 4 the performance of the algorithm meets the amplitude and phase requirements of the OMCT. Note that if input scaling is desired, that this must be accounted for when selecting the sagging percentage in order to still meet the OMCT requirements.

Summary

The Classical Algorithm of Reid and Nahon was reconfigured to allow rapid selection of the filter parameters. The Classical Algorithm was then automatically tuned using a simple optimisation method to compare its performance with an MPC algorithm as well as to generate an acceptable set of tuning coefficients that meet the requirements of the OMCT while maximising acceleration. Neither tuning ensures that the algorithm will be accepted by drivers in driving simulators as realistic. Instead the contribution is a rapid method of quantitatively tuning the Classical Algorithm to allow an easy comparison to other algorithms and standards. Towards that end, the MATLAB optimisation code and test harness is provided to ensure that the same starting point can be used for future comparisons. The code could be extended to consider the complete OMCT response shape.

References

ADVANI, S. K. ; HOSMAN, R. J. A. W. ; POTTER, MARIO: Objective Motion Fidelity Qualification in Flight Training Simulators. In: AIAA Modeling and Simulation Technologies Conference and Exhibit (2007)

CONRAD, B. ; SCHMIDT, S. F.: Motion drive signals for piloted flight simulators (1970)

FANG, ZHOU ; KEMENY, ANDRAS: A review and prospects of Renault's MPC based motion cueing algorithm for driving simulator. In: Driving Simulation Conference 2014 (2014)

FISCHER, MARTIN ; SEEFRIED, ANDREAS ; SEEHOF, CARSTEN: Objective Motion Cueing Test for Driving Simulators. In: Driving Simulation Conference 2016 (2016)

GROEN, ERIC ; HOSMAN, RUUD: Pilot's Perception and Control Behavior During Simulated Take Off. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Guidance, Navigation, and Control and Co-located Conferences : American Institute of Aeronautics and Astronautics (2003)

HOSMAN, R. ; ADVANI, S.: Design and evaluation of the objective motion cueing test and criterion. In: Aeronautical Journal Bd. 120 (2016), Nr. 1227, S. 873–891

REID, LD ; NAHON, MA: Flight Simulation Motion-base Drive Algorithimns: Part 1-Developing and Testing the Equations, 1985 — ISBN 1111111111

ROMANO, R A: Motion control logic for large excursion driving simulators, 1999

SCHMIDT, S.F. ; CONRAD, B. ; CENTER, AMES RESEARCH: A Study of Techniques for Calculating Motion Drive Signals for Flight Simulators. Bd. 5. Mountain View, CA, 1971

SINACORI, J.B.: A Practical Approach to Motion Simulation. In: Proceedings of the AIAA Visual and Motion Simulation Conference. New York, NY, 1973

MATLAB Code

```
MaxPosition=2.6; % meters
MaxTiltRate=5/180*3.1415; % rad/sec
MaxSag=.8; % amplitude ratio

%Scan accel values m/s/s
for MaxAccel=0.2:0.01:10.0

% Calculate filter parameters
  Wlpy=MaxTiltRate/MaxAccel*9.81*exp(1);
  Whpy=sqrt(MaxAccel/MaxPosition);

  % tf creates transfer function object
  syshp = tf( [1 0 0] , ...
    [1 2*Whpy Whpy*Whpy]);
  syslp = tf( [Wlpy*Wlpy] , ...
    [1 2*Wlpy Wlpy*Wlpy]);
  sys = syshp+syslp;

  [Y,T,W]=bode(sys);
  y=min(squeeze(Y(1,1,:)))
  sag=1-y;

  if((sag>MaxSag)||(Wlpy<Whpy))
    if(Wlpy<Whpy)
      % roll back one step
      MaxAccel=MaxAccel-0.01;
    Wlpy=MaxTiltRate/MaxAccel*9.81*exp(1);
      Whpy=sqrt(MaxAccel/MaxPosition);
      syshp = tf( [1 0 0] , ...
        [1 2*Whpy Whpy*Whpy]);
      syslp = tf( [Wlpy*Wlpy] , ...
        [1 2*Wlpy Wlpy*Wlpy]);
      sys = syshp+syslp;

      [Y,T]=step(sys);
      y=min(squeeze(Y(1,1,:)));
      sag=1-y;
    end

    step(sys*MaxAccel)
    figure
    fprintf('MaxAccel %f TiltRate %f LP %f
       HP %f Tilt %f\n', ...
       MaxAccel,MaxTiltRate*57.2958,Wlpy,Wh
       py,57.2958*MaxAccel/9.81);
    scale=1/3.1415*180/9.81; % deg/g
    bode(sys);
    break;
  end
end
```

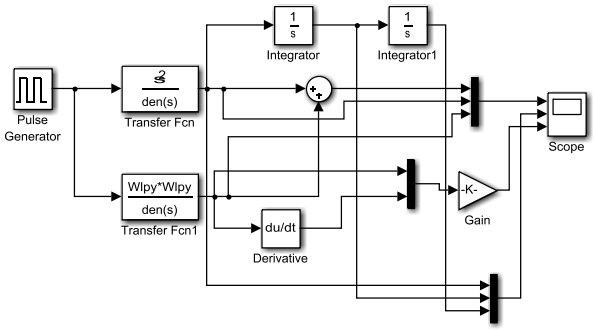


Figure 6. Test Harness