Nonlinear Motion Cueing Algorithm: Filtering at Pilot Station and Development of the Nonlinear Optimal Filters for Pitch and Roll

Kirill B. Zaychik and Frank M. Cardullo
State University of New York, Binghamton, New York
NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA’s STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA’s mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 443-757-5803

- Phone the NASA STI Information Desk at 443-757-5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320
Nonlinear Motion Cueing Algorithm: Filtering at Pilot Station and Development of the Nonlinear Optimal Filters for Pitch and Roll

Kirill B. Zaychik and Frank M. Cardullo
State University of New York, Binghamton, New York
The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.
I. Abstract

Telban and Cardullo [1] have developed and successfully implemented the non-linear optimal motion cueing algorithm at the Visual Motion Simulator (VMS) at the NASA Langley Research Center in 2005. The latest version of the non-linear algorithm performed filtering of motion cues in all degrees-of-freedom except for pitch and roll. This manuscript describes the development and implementation of the non-linear optimal motion cueing algorithm for the pitch and roll degrees of freedom. Presented results indicate improved cues in the specified channels as compared to the original design.

To further advance motion cueing in general, this manuscript describes modifications to the existing algorithm, which allow for filtering at the location of the pilot’s head as opposed to the centroid of the motion platform. The rational for such modification to the curing algorithms is that the location of the pilot’s vestibular system must be taken into account as opposed to the offset of the centroid of the cockpit relative to the center of rotation alone. Results provided in this report suggest improved performance of the motion cueing algorithm.
# II. Contents

I. Abstract ................................................................. i
II. Contents ........................................................................ iii
II. List of Tables .............................................................. v
II. List of Figures ............................................................. vi
III. Nomenclature ................................................................ x
1. Introduction ................................................................... 1
2. Background .................................................................... 1
   2.1. Simulator geometry and reference frames .................. 2
   2.2. Human perceptual system models ............................ 5
       2.2.1. Semicircular canals ........................................ 5
       2.2.2. Otoliths .......................................................... 6
       2.2.3. Human Vestibular model ................................ 6
   2.3. Linear Optimal Algorithm formulation .................... 8
   2.4. Non-linear Optimal Algorithm ............................... 9
3. Development of transformation equations for cues determined at pilot’s station .... 13
   3.1. Modifications of the original nonlinear optimal algorithm .... 13
       3.1.1. Filtering at Pilot’s Station ................................. 13
       3.1.2. Variable flow .................................................... 15
       3.1.3. Equations ....................................................... 17
   3.2. New algorithm evaluation ....................................... 17
   3.3. Discussions .......................................................... 19
4. Design and Development of Nonlinear Optimal Filters for Two Rotational Degrees of Freedom ...................................................... 23
   4.1. Problem description ............................................... 23
   4.2. Algorithm development ........................................... 23
       4.2.1. Pitch ............................................................... 23
       4.2.2. Roll ................................................................. 26
       4.2.3. Modifications to the on-line implementation code .... 27
   4.4. Algorithm tuning .................................................... 32
   4.5. Discussions .......................................................... 36
Appendix A. Programming updates ........................................ 38
1. Common variable listing ............................................... 38
   1.1. comint2.com ........................................................ 38
   1.2. optint3.com ........................................................ 38
   1.3. matrix1c.com ....................................................... 39
   1.4. nopt4.com ............................................................ 39
2. Program Listing .......................................................... 42
   2.1. gainopt4.f ............................................................. 42
   2.2. integ4.f ................................................................. 44
   2.3. invplf.f ................................................................. 47
   2.4. jackdrvr.f ............................................................. 50
   2.5. liba.f .................................................................. 54
Appendix B. Non-linear optimal algorithm: filtering at platform centroid (original) vs. filtering at PS (modified) ................................................................. 102
1. Pitch ............................................................................. 102
2. Roll ............................................................................. 109
3. Yaw ............................................................................. 115
4. Surge ......................................................................... 122
5. Sway ............................................................................ 128
6. Heave ........................................................................... 134

Appendix C. Non-linear optimal algorithm: original vs. augmented .................................. 141
1. Pitch ......................................................................... 141
2. Roll ......................................................................... 148
3. Yaw ......................................................................... 154
4. Surge ........................................................................ 161
5. Sway ........................................................................... 168
6. Heave ......................................................................... 174

Appendix D. Non-linear optimal algorithm: original applied at PS (filt. @ PS Original) vs. augmented (Augmented) ......................................................... 181
1. Pitch ........................................................................... 181
2. Roll ........................................................................... 188
3. Yaw ........................................................................... 195
4. Surge ........................................................................ 202
5. Sway ........................................................................... 209
6. Heave ........................................................................... 216

References ........................................................................ 223
II. List of Tables

Table 3.1. Upper and Lower ball joints coordinates .......................................................... 15
Table 3.2. Characteristics of the input signal for each degree of freedom ...................... 18
II. List of Figures

Figure 2.1. Vehicle simulator structure. Adopted from Telban and Cardullo [1] .......... 2
Figure 2.2. Reference frames used in the algorithm and their mutual orientation. Adopted from Telban and Cardullo [1] ................................................................. 3
Figure 2.3. VMS motion system geometry. Adapted from Telban and Cardullo [1] ........ 4
Figure 2.4. Linear Optimal Algorithm Structure. Adopted from Telban and Cardullo [1] 8
Figure 2.5. Optimal Algorithm Implementation for Longitudinal Mode. Adopted from Telban and Cardullo [1] ................................................................. 9
Figure 2.6. Non-linear Optimal Cueing Algorithm Structure. Adopted from Telban and Cardullo [1] ................................................................. 10
Figure 2.7. Nonlinear optimal algorithm implementation. Longitudinal mode. Adopted from Telban and Cardullo [1] ................................................................. 10
Figure 2.8. Nonlinear Algorithm Implementation with Unity-Gain Pitch Filter. Adopted from Telban and Cardullo [1] ................................................................. 11
Figure 3.1. Geometrical interpretation of the $F_{r_3}$ reference frame shift from $O_3$ (centroid of the upper motion platform) to $O_{PS}$ (pilot station) ........................................ 14
Figure 3.2. Vectors of the j-th actuator ................................................................. 14
Figure 3.3. Variables (accelerations) flow ............................................................ 16
Figure 3.4. The modified version of the online implementation of the nonlinear washout filter (longitudinal channel) ................................................................. 17
Figure 3.5. Nonlinear Algorithm Implementation for Yaw Mode.............................. 19
Figure 3.6. Aircraft and Platform accelerations at the centroid of the motion platform... 20
Figure 3.7. The XY plane of the $F_c$, when being placed at the centroid of the motion platform................................................................. 21
Figure 3.8. Tilt angular velocities for sway and surge channels............................... 22
Figure 3.9. Aircraft and Simulator sensed Specific Forces and Angular Rates ............ 22
Figure 4.1. Nonlinear Algorithm implementation for longitudinal mode. The dotted box in this figure encompasses the pitch channel. Adopted from Telban and Cardullo [1] .... 26
Figure 4.2. Flowchart of the augmented nonlinear washout algorithm. NFILP and NFILQ are the Riccati equation solvers for the roll and pitch channels respectively ........... 28
Figure 4.3. NFILQ subroutine flowchart ............................................................... 29
Figure 4.4. NFILP subroutine flowchart ............................................................... 30
Figure 4.5. STATE4 subroutine flowchart ............................................................. 31
Figure 4.6. Sensed specific force and angular rates for the pitch channel, with the tuned version of the nonlinear washout filter ................................................................. 32
Figure 4.7. Sensed specific force and angular rates for the pitch channel, with the tuned version of the nonlinear washout filter ................................................................. 36
Figure B.1. 1 ................................................................. 102
Figure B.1. 2 ................................................................. 103
Figure B.1. 3 ................................................................. 104
Figure B.1. 4 ................................................................. 105
Figure B.1. 5 ................................................................. 106
Figure B.1. 6 ................................................................. 107
Figure B.1. 7 ................................................................. 108
| Figure C.2. 1 | 109 |
| Figure C.2. 2 | 110 |
| Figure C.2. 3 | 111 |
| Figure C.2. 4 | 112 |
| Figure C.2. 5 | 113 |
| Figure C.2. 6 | 114 |
| Figure C.2. 7 | 115 |
| Figure B.3. 1 | 116 |
| Figure B.3. 2 | 117 |
| Figure B.3. 3 | 118 |
| Figure B.3. 4 | 119 |
| Figure B.3. 5 | 120 |
| Figure B.3. 6 | 121 |
| Figure B.3. 7 | 122 |
| Figure B.4. 1 | 123 |
| Figure B.4. 2 | 124 |
| Figure B.4. 3 | 125 |
| Figure B.4. 4 | 126 |
| Figure B.4. 5 | 127 |
| Figure B.4. 6 | 128 |
| Figure B.4. 7 | 129 |
| Figure B.5. 1 | 130 |
| Figure B.5. 2 | 131 |
| Figure B.5. 3 | 132 |
| Figure B.5. 4 | 133 |
| Figure B.5. 5 | 134 |
| Figure B.5. 6 | 135 |
| Figure B.5. 7 | 136 |
| Figure B.6. 1 | 137 |
| Figure B.6. 2 | 138 |
| Figure B.6. 3 | 139 |
| Figure B.6. 4 | 140 |
| Figure B.6. 5 | 141 |
| Figure C.1. 1 | 142 |
| Figure C.1. 2 | 143 |
| Figure C.1. 3 | 144 |
| Figure C.1. 4 | 145 |
| Figure C.1. 5 | 146 |
| Figure C.1. 6 | 147 |
| Figure C.1. 7 | 148 |
| Figure C.2. 1 | 149 |
| Figure C.2. 2 | 150 |
| Figure C.2. 3 | 151 |
Figure D.3. 1. ........................................................................................................... 195
Figure D.3. 2. ........................................................................................................... 196
Figure D.3. 3. ........................................................................................................... 197
Figure D.3. 4. ........................................................................................................... 198
Figure D.3. 5. ........................................................................................................... 199
Figure D.3. 6. ........................................................................................................... 200
Figure D.3. 7. ........................................................................................................... 201
Figure D.4. 1. ........................................................................................................... 202
Figure D.4. 2. ........................................................................................................... 203
Figure D.4. 3. ........................................................................................................... 204
Figure D.4. 4. ........................................................................................................... 205
Figure D.4. 5. ........................................................................................................... 206
Figure D.4. 6. ........................................................................................................... 207
Figure D.4. 7. ........................................................................................................... 208
Figure D.5. 1. ........................................................................................................... 209
Figure D.5. 2. ........................................................................................................... 210
Figure D.5. 3. ........................................................................................................... 211
Figure D.5. 4. ........................................................................................................... 212
Figure D.5. 5. ........................................................................................................... 213
Figure D.5. 6. ........................................................................................................... 214
Figure D.5. 7. ........................................................................................................... 215
Figure D.6. 1. ........................................................................................................... 216
Figure D.6. 2. ........................................................................................................... 217
Figure D.6. 3. ........................................................................................................... 218
Figure D.6. 4. ........................................................................................................... 219
Figure D.6. 5. ........................................................................................................... 220
Figure D.6. 6. ........................................................................................................... 221
Figure D.6. 7. ........................................................................................................... 222
III. Nomenclature

In order to be consistent with the original non-linear optimal algorithms developed report by Telban and Cardullo [1], similar nomenclature was adopted in the current report.

Symbols

\[ \mathbf{a} \text{ acceleration} \]
\[ \mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T \]

\[ \mathbf{A}_j \text{ coordinates of the upper bearing block of the j-th actuator} \]

\[ \mathbf{B}_j \text{ coordinates of the lower bearing block of the j-th actuator} \]

\[ \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{H} \text{ matrices of the state-space model of a control system} \]

\[ \mathbf{A}' \text{ system matrix of the standard form optimal control system} \]

\[ d, e, \delta, \gamma, \lambda \text{ NASA adaptive algorithm washout parameters} \]

\[ \mathbf{E} \text{ objective function or energy norm for neurocomputing approach} \]

\[ e \text{ pilot sensation error} \]

\[ \mathbf{Fr} \text{ reference frame} \]

\[ f \text{ specific force} \]

\[ \hat{f} \text{ sensed specific force} \]

\[ G_{o}, G_{s} \text{ gain sensitivities in the otolith and semicircular canals models} \]

\[ \mathbf{g} \text{ acceleration due to gravity} \]

\[ J \text{ system cost function} \]

\[ \mathbf{K} \text{ state feedback gain matrix} \]

\[ l_j \text{ length of the j-th motion platform actuator} \]

\[ \mathbf{L}_{SI} \text{ transformation matrix from simulator into inertial frame} \]
solution of the algebraic Riccati equation

weighting matrices in a cost function (tracking form)

weighting matrix for nonlinear algorithm control law

weighting matrices in a cost function (standard form)

radius vector

Laplace variable

transformation matrix from angular velocity to Euler angle rates

coefficients in the semicircular canal sensation model

input to a control system

input to the standard form optimal control system

time output of neurocomputing solver

white noise

optimal algorithm transfer function matrix

system state vector

desired state space system output

excitatory input signal for neurocomputing system

prescribed degree of nonlinearity for nonlinear algorithm

Euler angles $\beta = [\phi \ \theta \ \psi]^T$

pilot control input vector

filtered white noise break frequency

learning parameter for neurocomputing solver

time constants in the semicircular and otolith sensation models

density of the otoconial membrane
\(\omega\) angular velocity about the body frame \(\omega = [p \quad q \quad r]^T\)

\(\hat{\omega}\) sensed angular velocity

**Subscripts**

Subscripts indicate to what the main symbol is related.

- \(A\) aircraft
- \(CG\) center of gravity of aircraft
- \(d\) simulator states included in the cost function
- \(e\) sensation or perceptual error
- \(i\) inertial reference frame
- \(j\) j-th actuator of the motion platform
- \(n\) white noise input states
- \(OTO\) otolith model
- \(PS\) pilot station
- \(PA\) pilot in the aircraft
- \(S\) simulator
- \(SCC\) semicircular canals sensation model
- \(ST\) simulator tilt coordination channel
- \(VEST\) human vestibular system
- \(VIS\) human visual system
- \(x, y, z\) x, y, or z component
- \(a\) relates to system with nonlinearity
Superscripts

Superscripts indicate which reference frame the main symbol is in

( )^A in aircraft reference frame Fr_A
( )^I in inertial reference frame Fr_I
( )^S in simulator reference frame Fr_S
1. Introduction

This report documents the modifications to the NASA Non-Linear Optimal Motion Cueing Algorithm. The report consists of two major parts.

The first part describes modifications to the non-linear optimal algorithm, which are needed in order to perform filtering at the pilot station location as opposed to the original design of the algorithm where filtering was done at the centroid of the motion platform of the simulator. The essence of such a modification is in shifting the origin of the simulator attached reference frame from the centroid of the motion platform to wherever the location of the pilot station is. It could be the pilot’s head or pilot’s abdomen for instance. The new algorithm evaluation is also presented.

The second part of the report describes the development of the non-linear optimal filters for the additional two rotational degrees of freedom such as pitch and roll. In the original design of the algorithm only scaling and limiting was implemented for these rotational degrees of freedom. Note, that for the yaw channel the nonlinear washout filter was successfully implemented by Telban and Cardullo [1]. This report also delivers the FORTRAN code necessary for successful implementation of these algorithms on the NASA Langley Visual Motion Simulator (VMS).

It is assumed that the reader is familiar with the original work on development of the non-linear optimal algorithm performed by Telban and Cardullo [1]. For that reason some of the sections of the report are relatively concise.

2. Background

Figure 2.1 illustrates the basic vehicle simulator structure. As one can see, the motion cueing algorithm plays an essential part in the entire simulator architecture. The prime objective of any motion cueing algorithm is to provide a human operator with an array of cues, which will evoke behavior consistent with that in the real aircraft. It is obvious that due to some physical limitations none of the existing ground simulators are capable of delivering that 100%. Hence, motion cueing algorithms are designed to “trick”
a person into believing that he/she is experiencing cues similar to those in a real flight. The latest innovation in this area is the non-linear optimal algorithm designed by Robert Telban and Frank Cardullo [1]. This chapter is dedicated to describing basic concepts of the non-linear washout algorithm. However, for better understanding of the non-linear algorithm, the description of the linear optimal algorithm is given first. Some mathematical aspects of on-line implementation are addressed in this chapter along with the description of the human perceptual models utilized in the non-linear as well as lineal optimal algorithms.

Figure 2.1. Vehicle simulator structure. Adopted from Telban and Cardullo [1]

2.1. Simulator geometry and reference frames

There are four reference frames involved in algorithm design: aircraft center of gravity reference frame (RF), Fr_{CG}, aircraft RF, Fr_{A}, simulator RF, Fr_{S}, and the inertial RF, Fr_{I}. Figure 2.2 illustrates these RFs as they are oriented in space and with respect to each other. It can be seen that Fr_{CG} has its origin in the center of gravity of the aircraft. Fr_{S} is attached to the centroid of the upper motion platform of the simulator. Zs is directed downward and perpendicular to the plane of the motion platform. Xs is looking
forward, whereas $Y_S$ is pointing toward the pilot’s right hand side. $F_{r_A}$ is associated with the similar point in the aircraft cockpit as $F_{r_S}$ on the simulator platform. All three RFs are parallel to each other. The inertial RF $F_{r_I}$ is attached to the simulator motion system base. $F_{r_I}$ is oriented in such a way that $Z_I$ is parallel to gravity vector and $Y_I$ pointing to the right with respect to the simulator operator.

![Diagram of reference frames](image)

**Figure 2.2. Reference frames used in the algorithm and their mutual orientation. Adopted from Telban and Cardullo [1]**

The NASA Langley VMS motion system is a six degrees of freedom synergistic type device. The geometry of this motion system is shown in Figure 2.3.
Figure 2.3. VMS motion system geometry. Adapted from Telban and Cardullo [1]
2.2. Human perceptual system models

Another characteristic of this motion cueing algorithm is that it incorporates a model of the human vestibular system, with the new semicircular canal and otoliths models. A new integrated visual-vestibular perception model is also involved in the design. Note that the models of semicircular canals, otoliths and visual-vestibular interaction constitute the perceptual model of a pilot. Figure 2.6 illustrates how such a model falls into the entire concept of washout filters.

2.2.1. Semicircular canals

The semicircular canals are responsible for sensing angular motion. For the implementation into the linear optimal as well as nonlinear optimal algorithms the following mathematical model of the semicircular canals was presented [1]:

\[
\frac{\phi(s)}{\alpha(s)} = K_{scc} \left[ \frac{\tau_c s}{1 + \tau_c s} \right] \left[ \frac{1 + \tau_i s}{1 + \tau_i s} \right],
\]

where \(\phi(s)\) is the deflection of the cupula (a leaf-like structure in the semicircular canals, which deflects if the head is accelerated or decelerated) and \(\alpha(s)\) is the stimulus acceleration. This model takes into account both the semicircular canal dynamics and neural transduction dynamics. For implementation into the linear optimal and non-linear optimal cueing algorithms, angular velocity is employed as a stimulus, requiring the following transfer function:

\[
\frac{\hat{\omega}(s)}{\omega(s)} = 5.73 \frac{80s}{(1 + 80s)(1 + 5.73s)(1 + 0.005s)},
\]

where \(\omega(s)\) and \(\hat{\omega}(s)\) are stimulus and sensed angular velocities respectively.

Note that for the online implementation of the algorithm, a reduced form of the transfer function was used, which is given in equation (2.3)

\[
\frac{\hat{\omega}(s)}{\omega(s)} = 5.73 \frac{80s}{(1 + 80s)(1 + 5.73s)},
\]
2.2.2. Otoliths

The otolith organs are the elements of the vestibular system that provide linear motion sensation in humans and mammals. These organs are responsive to specific force, responding to both linear acceleration and tilting of the head with respect to the gravity vector. Telban and Cardullo [1] proposed the following otolith model, which provides the relationship between the sensed response and the specific force stimulus:

\[
\hat{f} = f = K_{OTO} \frac{(\tau_1 s + 1)}{(\tau_1 s + 1)(\tau_2 s + 1)},
\]

(2.4)

where \( K_{OTO} = 0.4, \tau_1 = 5 \text{ sec}, \tau_2 = 0.016 \text{ sec}, \) and \( \tau_L = 10 \text{ sec} \). For implementation into the motion cueing algorithms, Eq. 2.4 can be rewritten as

\[
\hat{f} = f = K'_{OTO} \frac{(s + A_0)}{(s + B_0)(s + B_1)}.
\]

(2.5)

where \( A_0 = 1/\tau_1, B_0 = 1/\tau_1, B_1 = 1/\tau_2, \) and \( K'_{OTO} = K_{OTO} \tau_1 \tau_2 / \tau_L \).

2.2.3. Human Vestibular model

The following section illustrates how the otolith and semicircular canals models are integrated together for further utilization in the nonlinear optimal washout algorithm. According to the formulation of the non-linear washout algorithm, which will be presented in the last section of this chapter, the human perceives the signal \( u \), comprised of both the angular velocity and translational accelerations:

\[
u = \begin{bmatrix} \dot{\theta} \\ a_s \\ u_1 \\ u_2 \end{bmatrix}.
\]

(2.6)

The semicircular canal model (Eq. (2.2)) can be rewritten in a more formal way:
\[
\dot{\theta} = \frac{G_{SCC} \tau_s \tau_a s^2 (1 + \tau_L s)}{(1 + \tau_a s)(1 + \tau_s s)(1 + \tau_L s)} u_t,
\]  \tag{2.7}

where values for semicircular canals time constants \( \tau_a, \tau_2, \tau_4, \) and \( \tau_L \) are given in Eq. 2.2, and \( G_{SCC} \) is the angular velocity threshold that scales the response to threshold units. Eq. 2.7, in turn, can be rewritten as

\[
\dot{\theta} = \frac{T_s s^3 + T_4 s^2}{s^3 + T_2 s^2 + T_4 s + T_0} u_t,
\]  \tag{2.8}

where:

\[
T_0 = \frac{1}{\tau_a \tau_2}, \quad T_1 = \frac{\tau_a + \tau_1 + \tau_2}{\tau_a \tau_1 \tau_2}, \quad T_2 = \frac{\tau_4 \tau_2 + \tau_a (\tau_1 + \tau_2)}{\tau_a \tau_1 \tau_2}, \quad T_3 = G_{SCC} / \tau_2, \quad \text{and} \quad T_4 = G_{SCC} \tau_L / \tau_2,
\]

and can be defined in state space notation as

\[
\dot{x}_{SCC} = A_{SCC} x_{SCC} + B_{SCC} u
\]  \tag{2.9}

which in observer canonical form is,

\[
A_{SCC} = \begin{bmatrix}
-T_2 & 1 & 0 \\
-T_1 & 0 & 1 \\
-T_0 & 0 & 0
\end{bmatrix}, \quad B_{SCC} = \begin{bmatrix}
T_3 - T_2 T_4 \\
-T_1 T_4 \\
-T_0 T_4
\end{bmatrix}, \quad C_{SCC} = [1 \ 0 \ 0], \quad \text{and} \quad D_{SCC} = [T_4 \ 0].
\]

On the other hand, the otolith model (Eq. 2.4) can be redefined in a state space notion as:

\[
\dot{x}_{OTO} = A_{OTO} x_{OTO} + B_{OTO} u
\]

\[
\hat{f}_x = C_{OTO} x_{OTO} + D_{OTO} u,
\]  \tag{2.10}

where \( x_{OTO} \) are the otoliths states, and

\[
A_{OTO} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
-b & -a & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & -b & -a
\end{bmatrix}, \quad B_{OTO} = \begin{bmatrix}
c & 0 \\
d - ac & 0 \\
e & 0 \\
0 & f \\
0 & h - af
\end{bmatrix},
\]

\[
C_{OTO} = [1 \ 0 \ 0 \ 1 \ 0], \quad D_{OTO} = [-G_{OTO} K_{OTO} R_{SC} \ 0].
\]
The representations in Eq. 2.9 and 2.10 can be combined to form a single representation for the human vestibular model:

\[
\begin{align*}
\dot{x}_v &= A_v x_v + B_v u \\
\hat{y}_v &= C_v x_v + D_v u,
\end{align*}
\]  

(2.11)

where \(x_v\) and \(\hat{y}_v\) are, respectively, the combined states and sensed responses, and \(A_v\), \(B_v\), \(C_v\), and \(D_v\) represent the vestibular models as one set of state equations:

\[
A_v = \begin{bmatrix} A_{scc} & 0 \\ 0 & A_{oro} \end{bmatrix}, \quad B_v = \begin{bmatrix} B_{scc} \\ B_{oro} \end{bmatrix}, \quad C_v = \begin{bmatrix} C_{scc} & 0 \\ 0 & C_{oro} \end{bmatrix}, \quad D_v = \begin{bmatrix} D_{scc} \\ D_{oro} \end{bmatrix}.
\]

2.3. Linear Optimal Algorithm formulation

Before getting to the description of the non-linear optimal washout algorithm, some background information on the linear optimal algorithm is presented. Note that both algorithms are designed based on the same principles and concepts. The major differences are in real-time implementation of washout filters.

Figure 2.4 contains the block diagram of the linear optimal algorithm structure.

![Linear Optimal Algorithm Structure](image)

Figure 2.4. Linear Optimal Algorithm Structure. Adopted from Telban and Cardullo [1]
Since the entire purpose of the washout filters is to minimize the sensation error, the transfer function matrix $W(s)$, which relates the desired simulator motion input to the aircraft input, is to be determined. In other words the elements of $W(s)$ are the coefficients of the washout filter. The linear optimal algorithm generates the desired transfer functions $W(s)$ by solving the Riccati equation by an off-line program, which are then implemented on-line. Figure 2.5 illustrates how the linear optimal washout filters are implemented on-line.

![Diagram](image)

**Figure 2.5. Optimal Algorithm Implementation for Longitudinal Mode. Adopted from Telban and Cardullo [1]**

### 2.4. Non-linear Optimal Algorithm

The non-linear optimal algorithm is formulated in a similar fashion to that of the linear optimal algorithm, except for the differences in computing the matrix $W(s)$. The structure of the algorithm is shown in Figure 2.6.

As can be seen the solution to the Riccati equation, which was obtained offline in the linear optimal algorithm, is now implemented in real time, resulting in the necessary matrix for computing the desired non-linear optimal filters.
Figure 2.6. Non-linear Optimal Cueing Algorithm Structure. Adopted from Telban and Cardullo [1]

Figure 2.7 illustrates how the non-linear washout filter is formulated for the longitudinal mode.

Figure 2.7. Nonlinear optimal algorithm implementation. Longitudinal mode. Adopted from Telban and Cardullo [1]
There are two separate filtering channels for translational and rotational degrees of freedom with the cross-feed path providing tilt coordination cues.

The aircraft acceleration is first transformed from the simulator attached RF to the inertial RF. The signal is then passed through the non-linear scaling and limiting block. The resulting signal then becomes an input to the “State Equations” block, from which the simulator translational acceleration is produced. This acceleration is integrated twice to produce the simulator translational position command $S_t$. Signals $S_t$ and $\dot{S}_t$ form a feedback loop and serve as inputs to the “Riccati Solver” block. The solution to the Riccati equation is the matrix $K(\alpha)$, which is fed back to “State Equations” block.

The aircraft angular velocity $\omega^A$ is transformed to the Euler angular rate ($\dot{\beta}_A$). Next it is limited and scaled. A separate set of State equations is employed along with the Riccati solver. The resulting signal is $\beta_S$ - the simulator angular position command. For the previous on-line implementation, however, the case of a unity-gain pitch (and roll) filter was implemented. Hence, Figure 2.7 can be redrawn as it is shown in Figure 2.8.

Figure 2.8. Nonlinear Algorithm Implementation with Unity-Gain Pitch Filter. Adopted from Telban and Cardullo [1].
The simulator translational $S_I$ and angular $\beta_S$ position commands are then transformed from degrees-of-freedom space to simulator actuator space. Actuator commands are then generated to achieve the desired simulator platform motion.

As one might already be aware, solving the Riccati equation in real time is a computationally challenging task. Conventionally a Newton-Raphson technique is utilized for that purpose. The main drawback is that it involves a matrix inversion, which can result in singular solutions for ill-conditioned systems. The non-linear optimal algorithm uses the structured neural network to solve the Riccati equation in real time. The main advantage of the neural computing approach over the Newton-Raphson is speed due to the fact that neither matrix inversion nor computation of the Jacobian matrix as a Kronecker product is required. Moreover, the problem of having a singular solution eliminates itself, since no matrix inversion is involved.
3. Development of transformation equations for cues determined at pilot’s station

3.1. Modifications of the original nonlinear optimal algorithm

Before continuing with this section of the report a new terminology shall be introduced. Pilot’s Station (PS) is the point in the simulator cockpit where washout filters are applied. In the original design of the non-linear optimal algorithm filtering was applied at the origin of the simulator reference frame Frs. This report proposes modification to the original non-linear algorithm, which position the PS at the location of the pilot’s head. The rational here is such that filtering must be performed at the location of the pilot’s vestibular apparatus.

3.1.1. Filtering at Pilot’s Station.

According to the theory of washout filter development the location where the actual “filtering” is being performed is associated with the location of the origin of the simulator reference frame. As has been previously mentioned, in the original design of the algorithm the location of the origin of the simulator attached RF is at the centroid of the upper motion platform. Hence the essence of applying non-linear washout filters at the PS location as opposed to the centroid of the upper joint bearings of the motion platform is, in fact, the shift of the simulator related reference frame Frs from the motion platform to wherever the PS is located. The given shift vector $R_{ss}$ (Figure 3.1) has the following coordinates in $Fr_{s}$: [0.0254, -0.653, -2.1946] (unique to the NASA Langley Visual Motion Simulator, VMS). The vectors $A_{s}$ connect the origin of the PS to each platform attach point. $B_{js}$ vectors will change as well. These vectors are different from the original implementation and are computed by means of a simple coordinate transformation. Table 3.1 contains coordinates of $A_{s}$ vectors before and after the transformation.
Figure 3.1. Geometrical interpretation of the $F_{Rs}$ reference frame shift from $O_S$ (centroid of the upper motion platform) to $O_{PS}$ (pilot station)

Figure 3.2. Vectors of the j-th actuator
Figure 3.2 demonstrates the relative location of the upper and lower ball joint bearings of the \( j \)-th actuator of the simulator. It is quite obvious that when the location of the \( Fr_s \) origin is shifted from \( O_s \) to \( O_{rs} \), \( R_t \) and \( A_j \) (along with \( B_j \)) are changed accordingly.

### Table 3.1. Upper and Lower ball joints coordinates

<table>
<thead>
<tr>
<th>Vector</th>
<th>Original coordinates</th>
<th>Vector</th>
<th>Modified coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
<td>([2.1117179, 0.0762, 0.0])</td>
<td>( A'_1 )</td>
<td>([2.0863179, 0.7112, 2.1946])</td>
</tr>
<tr>
<td>( A_2 )</td>
<td>([2.1117179, -0.0762, 0.0])</td>
<td>( A'_2 )</td>
<td>([2.0863179, 0.5588, 2.1946])</td>
</tr>
<tr>
<td>( A_3 )</td>
<td>([-0.98986594, -1.8669, 0.0])</td>
<td>( A'_3 )</td>
<td>([-1.01526594, -1.2319, 2.1946])</td>
</tr>
<tr>
<td>( A_4 )</td>
<td>([-1.12184942, -1.7907, 0.0])</td>
<td>( A'_4 )</td>
<td>([-1.14724942, -1.1557, 2.1946])</td>
</tr>
<tr>
<td>( A_5 )</td>
<td>([-1.12184942, 1.7907, 0.0])</td>
<td>( A'_5 )</td>
<td>([-1.14724942, 2.4257, 2.1946])</td>
</tr>
<tr>
<td>( A_6 )</td>
<td>([-0.98986594, 1.8669, 0.0])</td>
<td>( A'_6 )</td>
<td>([-1.01526594, 2.5019, 2.1946])</td>
</tr>
<tr>
<td>( B_1 )</td>
<td>([1.5021, 1.9812, 2.5806])</td>
<td>( B'_1 )</td>
<td>([1.4767, 2.6162, 4.77524])</td>
</tr>
<tr>
<td>( B_2 )</td>
<td>([1.5021, -1.9812, 2.5806])</td>
<td>( B'_2 )</td>
<td>([1.4767, -1.3462, 4.77524])</td>
</tr>
<tr>
<td>( B_3 )</td>
<td>([0.9647, -2.2914, 2.5806])</td>
<td>( B'_3 )</td>
<td>([0.9898, -1.6564, 4.77524])</td>
</tr>
<tr>
<td>( B_4 )</td>
<td>([-2.4668, -0.3102, 2.5806])</td>
<td>( B'_4 )</td>
<td>([-2.4922, 0.3247, 4.77524])</td>
</tr>
<tr>
<td>( B_5 )</td>
<td>([-2.4668, 0.3102, 2.5806])</td>
<td>( B'_5 )</td>
<td>([-2.4922, 0.9452, 4.77524])</td>
</tr>
<tr>
<td>( B_6 )</td>
<td>([0.9647, 2.9214, 2.5806])</td>
<td>( B'_6 )</td>
<td>([0.9393, 2.9264, 4.77524])</td>
</tr>
</tbody>
</table>

The reader should be aware that the data presented above are applicable solely to the VMS facility at NASA Langley.

### 3.1.2. Variable flow

According to the code available in the original NASA report by Telban and Cardullo [1] it is clear that the accelerations used as inputs to the non-linear washout filters are computed at the origin of the \( Fr_A \), i.e. at the centroid of the upper motion platform. The specific forces at the PS are then calculated utilizing the knowledge of the simulator cockpit geometry. Therefore, the variable flow (accelerations in particular) can be presented in a form of the following block diagram (Figure 3.3).
The following is a summary of changes to the original design of the algorithm that had been done in order to perform filtering at the PS.

- An auxiliary block had been introduced into the block diagram of the online implementation of the original non-linear washout filter. This block calculates the a/c acceleration at the location of the pilot’s station in the aircraft reference frame \(F_{r_A}\). The modified version of the online implementation block diagram is given in Figure 3.4. The geometrical location of the PS in the VMS cockpit is known and defined by the vector \(R_{ss} = [0.0254, -0.653, -2.1946]\) in meters.

- Vectors, connecting the origin of the \(F_{r_3}\) and the joints of the upper motion platform had been recalculated, taking into account the shift of \(F_{r_3}\) from its former location at the centroid of the cockpit/upper motion platform to the new location at the pilot’s head (see section 3.1.1.).
The modified version of the online implementation of the nonlinear washout filter (longitudinal channel)

### 3.1.3. Equations

The following are the equations for computing translational accelerations at the pilot station which are implemented in the “Translation to PS” block, in Figure 3.4.

\[
\begin{align*}
    a_{xPS}^A &= a_{xA}^A - R_{sxs} (q^2 + r^2) + R_{sxy} (pq - \dot{r}) + R_{szs} (pr + \dot{q}) \\
    a_{yPS}^A &= a_{yA}^A + R_{sxs} (pq + \dot{r}) - R_{syy} (p^2 + r^2) + R_{szs} (qr - \dot{p}) \\
    a_{zPS}^A &= a_{zA}^A + R_{sxs} (pr - \dot{q}) + R_{szy} (qr + \dot{p}) - R_{szs} (p^2 + q^2)
\end{align*}
\]

(3.1)

where \(a_{PS}^A = [a_{xPS}^A, a_{yPS}^A, a_{zPS}^A]\), \(a_A^A = [a_{xA}^A, a_{yA}^A, a_{zA}^A]\), \(R_{ss} = [R_{sxs}, R_{syy}, R_{szs}]\), \(\alpha_A^A = [p, r, q]\), and \(\dot{\alpha}_A^A = [\dot{p}, \dot{q}, \dot{r}]\) respectively.

### 3.2. New algorithm evaluation

This section of the report presents the evaluation of the modified non-linear optimal washout algorithm. The evaluation is done in a form of a comparative analysis.
against the original non-linear optimal algorithm. Comparison is performed for 6 degrees of freedom (3 translational and 3 rotational) and is followed by a discussion.

Table 3.2 contains information on the degree of freedom and the corresponding type and characteristic of the input signal. In each case, the inputs are the accelerations m/s² for the translational and rad/s² for rotational degrees of freedom) measured at the aircraft centroid.

Table 3.2. Characteristics of the input signal for each degree of freedom

<table>
<thead>
<tr>
<th>Degree of freedom</th>
<th>Input</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal (x-channel)</td>
<td>Ramp to step</td>
<td>Peak magnitude: 1 m/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slope: 3 m/s²/s</td>
</tr>
<tr>
<td>Lateral (y-channel)</td>
<td>Half sine</td>
<td>Peak magnitude: 3 m/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration: 5 s</td>
</tr>
<tr>
<td>Vertical (z-channel)</td>
<td>Pulse</td>
<td>Peak magnitude: 1 m/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration: 10 s</td>
</tr>
<tr>
<td>Pitch</td>
<td>Pulse doublet</td>
<td>Magnitude: 0.1 rad/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration: 5 s</td>
</tr>
<tr>
<td>Roll</td>
<td>Pulse doublet</td>
<td>Magnitude: 0.1 rad/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration: 5 s</td>
</tr>
<tr>
<td>Yaw</td>
<td>Pulse doublet</td>
<td>Magnitude: 0.1 rad/s²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration: 5 s</td>
</tr>
</tbody>
</table>

Appendix B contains a set of plots for each degree of freedom. Each set consists of the following graphs:

- Actuator extensions (6)
- Aircraft and Simulator Sensed Specific Force (3)
- Aircraft and Simulator Sensed Angular Rate (3)
- Platform Velocity in Inertial Coordinates (3)
- Desired and Actual Platform Displacement (3)
- Specific Force at Aircraft Pilot Head (3)
- Specific Force at Simulator Pilot Head (3)
- Aircraft Angular Rate (3)
- Platform Angular Rate (3)
- Aircraft Angular Position (3)
- Desired and Actual Platform Angular Position (3)
- Aircraft Acceleration at MB Centroid (3)
3.3. Discussions

In this section the discussion of the peculiarities of the motion platform behavior associated with applying of the washout filter at the PS will be presented. For that purpose the yaw channel was chosen. Figure 3.5 illustrates how the nonlinear filter for the yaw channel was implemented online.

![Figure 3.5. Nonlinear Algorithm Implementation for Yaw Mode](image.png)

The characteristic feature of the yaw channel is that when excited there is no gravity alignment issue, which is associated with the longitudinal or lateral channel. In other words, there is no need to tilt the motion platform to produce the sustained acceleration cues.

The original nonlinear washout algorithm developed by Telban and Cardullo [1] did a very good job in simulating cues, inherent to a pure yaw motion of the aircraft. However, when the filtering is performed at the PS a few differences (compared to the original algorithm) in the behavior of the platform can be observed. For example, Figure 3.6 contains graphs for the aircraft and platform accelerations at the motion platform centroid.
Figure 3.6. Aircraft and Platform accelerations at the centroid of the motion platform

It can easily be seen that accelerations in the X and Y channels are nonzero when filtering is done at the PS. If one looks at the physics of the yaw motion of the platform, it is possible to reason the presence of those accelerations. Figure 3.7 illustrates the geometry of the motion platform if observed from the top (Figure 3.1 if observed from the top). The Z axis of the $Fr_A$ is aimed away from the reader and is perpendicular to the drawing, given that the origin of the $Fr_A$ is placed at the centroid of the motion platform. Vector OA represents the XY component of the vector $R_{xx}$. 
Consider the clockwise rotation of the platform with the angular velocity \( \dot{\omega}_z \). At point “A” this motion will result in the tangential acceleration \( a_t \). In Fr\(_A\) vector \( a_t \) has components \( a_{tx} \) and \( a_{ty} \). If filtering is performed at the PS, then accelerations \( a_{tx} \) and \( a_{ty} \) automatically form inputs to the translational and lateral channels of the washout algorithm. Which, in turn, results in the sway and surge motion of the platform (Figure 3.8).

One should also note that accelerations and angular rates resulting from this minor surge and sway motion of the platform is either on or below the perceptual threshold. Moreover, the modified algorithm resulted in better (closer to the aircraft) reproduction of the specific force (Figure 3.9).
Figure 3.8. Tilt angular velocities for sway and surge channels

Figure 3.9. Aircraft and Simulator sensed Specific Forces and Angular Rates
4. Design and Development of Nonlinear Optimal Filters for Two Rotational Degrees of Freedom

4.1. Problem description

According to the original design of the nonlinear algorithm, filtering in the channels of the rotational degrees of freedom, such as pitch and roll, was confined to limiting and scaling only. The major task of this part of the project is to develop nonlinear filters for two rotational degrees of freedom: pitch and roll.

4.2. Algorithm development

4.2.1. Pitch

The derivation process is similar to that performed by Telban [1] when designing the nonlinear filter for the yaw channel.

The semicircular canals model described in a form of a transfer function is given in Eq. (4.1), which is the reduced form of the semicircular model cited in Eq. (2.7). One can refer to the original report by Telban [1] for a detailed explanation on how such reduction was made. It is worth mentioning, however, that the simplified formula cuts down the computational burden substantially, which is essential for real time applications.

\[ \hat{\theta} = \frac{G_{SCC}s^2}{s^3 + T_1s + T_0}u, \]  

(4.1)

where \( \hat{\theta} \) is the sensed angular velocity, \( G_{SCC} \) is the angular velocity threshold gain, which scales the response to the threshold units, and \( T_1 \) and \( T_0 \) relate to the semicircular canals time constants.

In state space form this model constitutes the state equations:

\[ \dot{x}_{SCC} = A_{SCC}x_{SCC} + B_{SCC}u, \]
\[ \dot{\theta} = C_{SCC}x_{SCC} + D_{SCC}u, \]

(4.2)

where \( A_{SCC} = \begin{bmatrix} -T_1 & 1 \\ T_0 & 0 \end{bmatrix} \), \( B_{SCC} = \begin{bmatrix} -G_{SCC}T_1 \\ -G_{SCC}T_0 \end{bmatrix} \), \( C_{SCC} = [1 \ 0] \), \( D_{SCC} = [G_{SCC} \ 0] \).
The additional state due to optokinetic influence must be added to these equations:

$$\dot{\theta}_{\text{OK}} = \frac{T_2}{s + T_2} \dot{\theta}_e,$$

where $$\dot{\theta}_e = \dot{\theta}_A - \dot{\theta}_S$$, and $$T_2$$ relates to the time constant $$\tau_{\text{OK}}$$.

The state equations become then:

$$\dot{x}_{\text{SCC}} = A_{\text{SCC}} x_{\text{SCC}} + B_{\text{SCC}} u$$

$$\dot{\theta}_{\text{PE}} = C_{\text{SCC}} x_{\text{SCC}} + D_{\text{SCC}} u,$$

where

$$A_{\text{SCC}} = \begin{bmatrix} -T_1 & 1 & 0 \\ -T_0 & 0 & 0 \\ -T_2 & 0 & -T_2 \end{bmatrix}, \quad B_{\text{SCC}} = \begin{bmatrix} -G_{\text{SCC}} T_1 \\ -G_{\text{SCC}} T_0 \\ -G_{\text{SCC}} T_2 \end{bmatrix}, \quad C_{\text{SCC}} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \quad D_{\text{SCC}} = \begin{bmatrix} G_{\text{SCC}} & 0 \end{bmatrix}.$$

The ultimate set of state equations should include the otolith model as well. In the case of pure pitch, however, the human otoliths are not engaged. Hence the final version of the state equation set will be as follows:

$$\dot{x}_V = A_v x_v + B_v u$$

$$y_{\text{PE}} = C_v x_v + D_v u,$$

where $$x_v$$, and $$y_{\text{PE}}$$ are, respectively, the combined states and perceived responses, whereas matrices

$$A_v = A_{\text{SCC}}, \quad B_v = B_{\text{SCC}}, \quad C_v = C_{\text{SCC}}, \quad D_v = D_{\text{SCC}}.$$

The next step is to add additional motion platform states $$x_d$$ and filtered white noise $$x_n$$.

$$\dot{x}_d = A_d x_d + B_d u,$$

where $$x_d = \left[ \int \dot{\theta} dt \quad \dot{\theta} \right]$$, and $$A_d = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$ and $$B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In turn, the aircraft input can be expressed as a filtered white noise.
\[
\begin{align*}
\dot{x}_n &= -\gamma x_n + \gamma w, \\
u_A &= x_n, \\
\end{align*}
\] (4.6)
where \(\gamma\) is the break frequency for a given degree of freedom.

The state equations given in Eq. 4.4, 4.5, and 4.6 can be combined to form the desired system equation

\[
\dot{x} = Ax + Bu_s + Hw
\]
\[
y = [e \quad x_d] = Cx + Du_s,
\] (4.7)
with

\[
A_{SCC} = \begin{bmatrix} A_v & 0 & -B_v \\ 0 & A_d & 0 \\ 0 & 0 & -\gamma \end{bmatrix}, \\
B = \begin{bmatrix} B_v \\ B_d \\ 0 \end{bmatrix}, \\
H = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
C = \begin{bmatrix} C_v & 0 & -D_v \\ 0 & I & 0 \end{bmatrix}, \\
D = \begin{bmatrix} D_v \\ 0 \end{bmatrix},
\]
where \(y\) is the desired output, and \(x = [x_e \quad x_d \quad x_n]^T\) represents the combined states.

The standard optimal control form is then applied to form a cost function \(J\), which is later enhanced by the additional term \(e^{2\alpha t}\) [2], where \(\alpha\) is a scalar representing a minimum degree of stability in the closed loop system, \(\alpha > 0\).

\[
J' = E\left\{ \int_0^t e^{2\alpha t} \left( x^T R_1' x + u^T R_2 u \right) dt \right\},
\] (4.8)
where \(R_1'\) is positive definite and \(R_2\) is positive semi-definite.

The cost function constrains both the sensation error and the motion platform states.

The essence of the washout algorithm is to compute the simulator control input so that the given cost function is minimized. The solution is sought in the following form:

\[
u_s = -K(\alpha)x,
\] (4.9)
where \(K(\alpha)\) is the feedback matrix, which depends upon the solution to the algebraic Riccati equation. For the sake of brevity however, the author will omit the derivation of the solution and will get down to the on-line implementation of the algorithm, the block diagram of which is given in Figure 4.1. It is worth mentioning that the variable \(\alpha\) is set to depend upon the motion platform states: \(\alpha = x_d^T Q_2 x_d\).
Figure 4.1. Nonlinear Algorithm implementation for longitudinal mode. The dotted box in this figure encompasses the pitch channel. Adopted from Telban and Cardullo [1]

4.2.2. Roll

For the roll channel, the filter development is analogous to the pitch channel. The sensed rotational motion $\dot{\theta}$ in Eq. 4.1 is replaced by $\dot{\phi}$. The remaining development is identical in form to Eqs. 4.2 to 4.9, resulting in a matrix of fifth-order transfer functions $W(s)$ for the lateral mode. The on-line implementation of this mode is identical to Figure 4.1.
4.2.3. *Modifications to the on-line implementation code*

This section contains block diagrams and flowcharts of the nonlinear washout algorithm augmented by the non-linear filters for pitch and roll channels. The format of presentation is similar to that used by Telban and Cardullo [1].

The flowchart for the augmented nonlinear algorithm subroutine NEWOPT4 is shown in Figure 4.2. The RESET/HOLD modes and transformation subroutines are identical to the optimal algorithm discussed in the preceding section.

Two new subroutines, NFILP and NFILQ, accomplish the task of solving the Riccati equation in real time for the pitch and roll mode. The feedback matrices and updated motion states for each mode are then computed in the subroutine STATE4, which was also modified to accommodate new filters for pitch and roll. The subroutine INTEG4, which computes the desired simulator displacements and attitudes, taking into account the tilt coordination limits had to be modified as well. Appropriate flowcharts are given later in the text. Note that STATE4 has to be computed six times as opposed to four times in the original design.
Figure 4.2. Flowchart of the augmented nonlinear washout algorithm. NFILP and NFILQ are the Riccati equation solvers for the roll and pitch channels respectively.
Figure 4.3. NFILQ subroutine flowchart
\[ \alpha = x^TQ_xx \]
\[ \Lambda' = \Lambda' + \alpha I \]

\[ p = Pz \]
\[ v = (\text{PSP} \cdot \Lambda'_zP - \text{PA} \cdot \Lambda'_z)z \]

\[ \Delta P = \Lambda'_zvz^T + vz^T \Lambda'_z - v^T \Lambda'_z P \]

\[ P(k+1) = P(k) + \frac{\mu}{2} \left[ \Delta P(k) + \Delta P^T(k) \right] \]
Figure 4.5. STATE4 subroutine flowchart

Riccati Solution

Prior Inputs and States

Matrices

\[ R_2, A_v, B_v, C_v, D_v \]

更新前的输入和状态

A2NO, BADNO, XZO, XRO, XPO, XQO

\[ K_1 = R_2^T \left[ B_v^T P_{11} + B_{d1}^T P_{21} + D_v^T Q C_v \right] \]

\[ K_2 = R_2^T \left[ B_v^T P_{12} + B_{d2}^T P_{22} \right] \]

\[ K_3 = R_2^T \left[ B_v^T P_{13} + B_{d3}^T P_{23} - D_v^T Q D_v \right] \]

\[ \dot{x}_e = \int \dot{x}_e \, dt \]

\[ x_e = \int \dot{x}_a \, dt \]

\[ \dot{x}_a = \int \dot{x}_d \, dt \]

2\textsuperscript{nd}-Order Runge-Kutta Integration of States

Compute Lateral, Vertical, Yaw, Roll and Pitch States

XY, XZ, XR, XP, XQ

更新前的输入和状态

A2NO, BADNO, XZO, XRO, XPO, XQO

\[ \dot{x}_e = (A_v - B_v K_1) x_e - B_v K_2 x_d - B_v (I + K_3) u_A \Delta t \]

\[ \dot{x}_d = -B_d K_1 x_e - (A_d - B_d K_2) x_d - B_d K_3 u_A \Delta t \]

XXI1, XXI2

XX

XY, XZ, XR, XP, XQ

更新前的输入和状态
4.4. Algorithm tuning

Figure 4.6 contains comparison graphs of augmented filtering algorithms versus original washout filters for the *pitch channel*. Sensed angular rate and specific forces at the PS are of particular interest here.

![Comparison graphs of augmented filtering algorithms versus original washout filters for the pitch channel.](image)

**Figure 4.6. Sensed specific force and angular rates for the pitch channel, with the tuned version of the nonlinear washout filter**

It can be seen that augmented filters respond in an “under-gained” manner. For example, sensed specific force at the PS appears to be less than with the original nonlinear washout filters. Such a tendency can be observed in the roll channel as well. In order to increase the performance of the augmented filters an auxiliary gain was introduced for the pitch and roll channels. Subroutine “integ4.f” (Telban and Cardullo [3]) was modified in the following manner to accommodate such a gain. Desired pitch and roll angles are “boosted” before being combined with the tilt signal (due to gravity align). The modified code is highlighted with yellow color.
SUBROUTINE INTEG4

INCLUDE 'optint3.com'

INCLUDE 'comint2.com'

INCLUDE 'wcom2.com'

INCLUDE 'matrix1c.com'

INCLUDE 'nopt4.com'

SURGE FILTER OUTPUT ASI(1) & BSDT(2)

ASI(1) = -K1X(2,1)*XX(1)-K1X(2,2)*XX(2)-K1X(2,3)*XX(3)
  -K1X(2,4)*XX(4)-K1X(2,5)*XX(5)-K1X(2,6)*XX(6)
  -K2X(2,1)*XX(7)-K2X(2,2)*XX(8)-K2X(2,3)*XX(9)
  -K3X(2)*A2N(1)

BSDT(2) = -K1X(1,1)*XX(1)-K1X(1,2)*XX(2)-K1X(1,3)*XX(3)
  -K1X(1,4)*XX(4)-K1X(1,5)*XX(5)-K1X(1,6)*XX(6)
  -K2X(1,1)*XX(7)-K2X(1,2)*XX(8)-K2X(1,3)*XX(9)
  -K3X(1)*A2N(1)

SWAY FILTER OUTPUT ASI(2) & BSDT(1)

ASI(2) = -K1Y(2,1)*XY(1)-K1Y(2,2)*XY(2)-K1Y(2,3)*XY(3)
  -K1Y(2,4)*XY(4)-K1Y(2,5)*XY(5)-K1Y(2,6)*XY(6)
  -K2Y(2,1)*XY(7)-K2Y(2,2)*XY(8)-K2Y(2,3)*XY(9)
  -K3Y(2)*A2N(2)

BSDT(1) = -K1Y(1,1)*XY(1)-K1Y(1,2)*XY(2)-K1Y(1,3)*XY(3)
  -K1Y(1,4)*XY(4)-K1Y(1,5)*XY(5)-K1Y(1,6)*XY(6)
  -K2Y(1,1)*XY(7)-K2Y(1,2)*XY(8)-K2Y(1,3)*XY(9)
  -K3Y(1)*A2N(2)

HEAVE FILTER OUTPUT ASI(3)

ASI(3) = (-K1Z(1)*XZ(1)-K1Z(2)*XZ(2)-K2Z(1)*XZ(3)
   -K2Z(2)*XZ(4)-K2Z(3)*XZ(5)-K3Z*A2N(3))

YAW FILTER OUTPUT BSDR(3)

BSDR(3) = -K1R(1)*XR(1)-K1R(2)*XR(2)-K1R(3)*XR(3)
  -K2R*XR(4)-K3R*BADN(3)
C PITCH FILTER OUTPUT BSDR(2)

BSDR(2)= -K1P(1)*XP(1) - K1P(2)*XP(2) - K1P(3)*XP(3)
X - K2P*XP(4) - K3P*BADN(2)

C ROLL FILTER OUTPUT BSDR(1)

BSDR(1)= -K1Q(1)*XQ(1) - K1Q(2)*XQ(2) - K1Q(3)*XQ(3)
X - K2Q*XQ(4) - K3Q*BADN(1)

C LIMIT THE ANGULAR RATE IN THE CROSS-OVER TILT CHANNEL.
THE REAL TILT POSITION WILL BE DETERMINED BY BOTH THE
DESIRED
POSITION AND THE DIFFERENCE BETWEEN THE DESIRED AND REAL
TILT POSITION.

DO K=1,2
  BETASR(K)=BETASRO(K) + DT*BSDRO(K)
  BETAST(K)=BETASTO(K) + DT*BSDTO(K)
  DIF(K)=0.005*(BETAST(K)-BETASTLO(K))+(BETAST(K)-
  BETASTO(K))
  BETASTL(K)=BETASTL(K)+MAX(-
  BDLIM*DT,MIN(BDLIM*DT,DIF(K)))
C COMPUTE THE TILT ANGULAR VELOCITY
C
BSDTL(K)=(BETASTL(K)-BETASTLO(K))/DT

BSDRO(K) = BSDR(K)
BetASRO(K)=BETASR(K)
BetASTO(K)=BETAST(K)
BetASTLO(K)=BETASTL(K)
BS DTO(K)=BSDT(K)
END DO

C COMBINE THE TILT AND ROTATIONAL CHANNELS TO OBTAIN
THE DESIRED ANGULAR POSITION

G1=3

BETAS1(1)=BETASTL(1)+G1*BETASR(1)
BETAS1(2)=BETASTL(2)+G1*BETASR(2)
BETAS1(3)=XR(4)
USE DIFFERENCE BETWEEN DESIRED BETAST AND REAL BETAST
TO GENERATE ADDITIONAL LINEAR RESPONSE AND ACHIEVE
COORDINATION BETWEEN THE LINEAR AND TILT CHANNELS

\[
\begin{align*}
\text{SSI}_1(1) &= XX(8) + \text{RRS}(3) \times (\text{BETAST}(2) - \text{BETASTL}(2)) \\
\text{SSI}_1(2) &= XY(8) - \text{RRS}(3) \times (\text{BETAST}(1) - \text{BETASTL}(1))
\end{align*}
\]

FOR ON-GROUND MOTION, ADD RUNWAY ROUGHNESS EFFECT
AMPLITUDE IS FAIRED UPON TOUCHDOWN OR TAKEOFF

\[
\text{SSI}_1(3) = XZ(4) + XKA \times \sin((\text{WB} + \text{XKG} \times \text{VGSPD}) \times T)
\]

Swap Variables To Match Modified Algorithm
For Input to JACKDRVR

XDD = ASI(1)
YDD = ASI(2)
ZDD = ASI(3)
XD = XX(9)
YD = XY(9)
ZD = XZ(5)
X = SSI1(1)
Y = SSI1(2)
Z = SSI1(3)
PHI = BETAS1(1)
THE = BETAS1(2)
PSI = BETAS1(3)
PHID = BSDTL(1) + BSDR(1)
THED = BSDTL(2) + BSDR(2)
PSID = BSDR(3)

RETURN
END

Trial and error method yields a value of \( G1 = 3 \), which provides best tuning of the algorithm. Figure 4.7 contains comparison graphs of the modified augmented filtering algorithms against the original washout filters. It can be seen that modified augmented filters result in better (at least the same order of magnitude) cueing at the PS.
4.5. Discussion

A discussion of the benefits of applying the nonlinear filters to all degrees of freedom (augmented filtering), is presented using the pitch channel as an example. Appendix C contains a comprehensive set of graphs, which compares the augmented filtering with the original set of non-linear optimal filters.

As can be seen, augmented filtering algorithms provide at least as much cueing as original filters. Figures C.1.5 and C.1.7 in Appendix C also clearly illustrate how nonlinear filters washout the simulator platform attitude, as opposed to the original filter design, which result in sustained cues.

Figure 4.7. Sensed specific force and angular rates for the pitch channel, with the tuned version of the nonlinear washout filter
Recall that for the pitch channel, the reference signal was the pulse doublet. Figure C.1.1 clearly illustrates how nonlinear filters provide cue onset with the following washout. The same observation holds true for the roll channel as well (Appendix C).

It can also be seen that introduction of non-linear washout filters to the pitch and roll channels had no effect on other degrees of freedom.

It is useful to compare the augmented filtering algorithms against the original nonlinear filters applied at the PS. The latter were developed in Sections 3 of the current report. Such comparison makes sense since both filtering algorithms clearly have certain advantages over the original nonlinear filters. Appendix D contains a complete set of comparison graphs.

Using the pitch channel as an example it is possible to draw a few conclusions, which hold true for other degrees of freedom. It can be seen (Figure D.1.1) that original nonlinear filters applied at the PS as opposed to the augmented filters produce a noticeable accelerations at the MB centroid, which is undesirable. Moreover, as it has been noted previously, augmented filters provide cue washout (Figure D.1.2). Figure D.1.3 demonstrates, that if expressed in terms of the platform angular rate, both approaches yielded comparable results. Specific forces at the pilot’s head location (Figure D.1.4) are of almost the same magnitude. The augmented filters, however, produced the desired washout. Moreover, the augmented filters resulted in smaller displacements of the platform, which can be seen in Figure D.1.7.
Appendix A. Programming updates

1. Common variable listing

Followed is the common variables listing, which was enhanced due to introduction of the pitch and roll nonlinear filters. The listing contents had been preserved intact from the original work by Telban and Cardullo [3], save for some additions, which are marked in yellow.

1.1. comint2.com

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, Y, Z</td>
<td>Desired Platform Displacements (m)</td>
</tr>
<tr>
<td>XD, YD, ZD</td>
<td>Desired Platform Velocities (m/s)</td>
</tr>
<tr>
<td>XDD, YDD, ZDD</td>
<td>Desired Platform Accelerations (m/s/s)</td>
</tr>
<tr>
<td>PHI, THE, PSI</td>
<td>Desired Platform Attitudes (rad)</td>
</tr>
<tr>
<td>PHID, THED, PSID</td>
<td>Desired Platform Angular Velocities (rad/s)</td>
</tr>
</tbody>
</table>

1.2. optint3.com

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>Time Step (Sec)</td>
</tr>
<tr>
<td>XX1(8), X1O(8)</td>
<td>Filter State Variable 1 (Current and Prior Time)</td>
</tr>
<tr>
<td>X2(8), X2O(8)</td>
<td>Filter State Variable 2 (Current and Prior Time)</td>
</tr>
<tr>
<td>X3(8), X3O(8)</td>
<td>Filter State Variable 3 (Current and Prior Time)</td>
</tr>
<tr>
<td>X4(8), X4O(8)</td>
<td>Filter State Variable 4 (Current and Prior Time)</td>
</tr>
<tr>
<td>X5(6), X5O(6)</td>
<td>Filter State Variable 5 (Current and Prior Time)</td>
</tr>
<tr>
<td>X6(6), X6O(6)</td>
<td>Filter State Variable 6 (Current and Prior Time)</td>
</tr>
<tr>
<td>ACA(3), ACAO(3)</td>
<td>Aircraft Body Acceleration Vector (m/s/s)</td>
</tr>
<tr>
<td>BETAA(3)</td>
<td>Aircraft Euler Angle Vector (rad)</td>
</tr>
<tr>
<td>BETAAD(3)</td>
<td>Aircraft Euler Rate Vector (rad/sec)</td>
</tr>
<tr>
<td>A2(3)</td>
<td>Aircraft Inertial Acceleration Vector (m/s/s)</td>
</tr>
<tr>
<td>A2N(3), A2NO(3)</td>
<td>Scaled Aircraft Inertial Acceleration Vector (m/s/s)</td>
</tr>
<tr>
<td>BADN(3), BADNO(3)</td>
<td>Scaled Aircraft Euler Rate Vector (rad/s)</td>
</tr>
<tr>
<td>ASI(3), ASIO(3)</td>
<td>Desired Platform Acceleration Cue (m/s/s)</td>
</tr>
<tr>
<td>VSI(3), VSIO(3)</td>
<td>Desired Platform Velocity (m/s)</td>
</tr>
<tr>
<td>SSIIU(3), SSII(3), SSIO(3)</td>
<td>Desired Platform Displacement (m)</td>
</tr>
<tr>
<td>WAA(3)</td>
<td>Aircraft Body Velocity Vector (rad/sec)</td>
</tr>
<tr>
<td>BSDT(2), BSDTO(2), BSDTL(2)</td>
<td>Platform Tilt Cue (Current, Prior, and Limited) (rad/sec)</td>
</tr>
<tr>
<td>BETASAI(3)</td>
<td>Desired Platform Angular Position (rad)</td>
</tr>
<tr>
<td>BDLIM</td>
<td>Platform Tilt Cue Limit (rad/sec)</td>
</tr>
<tr>
<td>BSDR(3), BSDRO(3)</td>
<td>Platform Rotational Cue (Current and Prior) (rad/sec)</td>
</tr>
<tr>
<td>BETASR(3), BETASRO(3)</td>
<td>Platform Rotational Angle (Current and Prior) (rad/sec)</td>
</tr>
</tbody>
</table>
BETAST(2), BETASTO(2) Platform Tilt Angle (Current and Prior) (rad)
BETASTL(2), BETASTLO(2) Platform Tilt Angle with Limit (Current and Prior) (rad)
DIF(2) Difference between Current and Limited Tilt Angles
XH(2), XO(2) Trim Filter State Variables
T1N1, T1DO, T2N1, T2DO Trim Filter Coefficients
XT, XTO, XT2, XT2O Augmented Turbulence Filter State Variables
ACZT Augmented Turbulence Acceleration
WGUST, WGUSTO Z-Axis Gust Velocity (m/sec)
G1D0, G1D1 Augmented Turbulence Filter Coefficients (Denominator)
G1N0, G1N1, G1N2 Augmented Turbulence Filter Coefficients (Numerator)

1.3. matrix1c.com

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAIS(3,6)</td>
<td>Motion Base Actuator Coordinates (m)</td>
</tr>
<tr>
<td>BBII(3,6)</td>
<td>Fixed Base Actuator Coordinates (m)</td>
</tr>
<tr>
<td>RRS(3)</td>
<td>Vector from Motion Base Centroid to Pilot Head (m)</td>
</tr>
<tr>
<td>LENEUT</td>
<td>Actuator Neutral Length (m)</td>
</tr>
<tr>
<td>LI(6)</td>
<td>Desired Actuator Extensions (m)</td>
</tr>
<tr>
<td>RLI(6), RLOI(6)</td>
<td>Actual Actuator Extensions (m)</td>
</tr>
<tr>
<td>LEGC(6)</td>
<td>Actual Actuator Extensions (in)</td>
</tr>
<tr>
<td>SSI(3)</td>
<td>Actual Simulator Displacements (m)</td>
</tr>
<tr>
<td>BETAS(3)</td>
<td>Actual Simulator Attitudes (rad)</td>
</tr>
<tr>
<td>BRAKE(6)</td>
<td>Actuator Braking Region Value</td>
</tr>
<tr>
<td>RATIO(6)</td>
<td>Actuator Braking Ratio</td>
</tr>
<tr>
<td>LAVAIL</td>
<td>Actuator Available Length (m)</td>
</tr>
<tr>
<td>RLID(6), RLIDO(6)</td>
<td>Actuator Velocity (m/s)</td>
</tr>
<tr>
<td>RLIDD(6)</td>
<td>Actuator Acceleration (m/s²)</td>
</tr>
<tr>
<td>FLAG(6)</td>
<td>Braking Region Flag (0 or 1)</td>
</tr>
<tr>
<td>IT2, NC2</td>
<td>Braking Recovery Indices</td>
</tr>
<tr>
<td>SUMFLAG</td>
<td>Sum of FLAG(6) Values</td>
</tr>
</tbody>
</table>

1.4. nopt4.com

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal Mode</td>
<td></td>
</tr>
<tr>
<td>ALPX, ALPXMAX</td>
<td>Prescribed Nonlinearity α</td>
</tr>
<tr>
<td>APX(11,11), APXO(11)</td>
<td>State Space System Matrix A'</td>
</tr>
<tr>
<td>BRBX(11,11), R1PX(11,11), R2IX(2)</td>
<td>System Weighting Matrix S, R₁ᵀ, R₂</td>
</tr>
<tr>
<td>AVX(6,6), BVX(6,2), DQCX(6)</td>
<td>Vestibular State Space Matrices Aᵥ, Bᵥ, Dᵥ, QCᵥ</td>
</tr>
<tr>
<td>K1X(2,6), K2X(2,3), K3X(2)</td>
<td>Feedback Matrices K₁, K₂, K₃</td>
</tr>
<tr>
<td>ZX(11,11)</td>
<td>Neurocomputing Solver Bi-Polar Vectors Z</td>
</tr>
<tr>
<td>PX(11,11), PXVEC(66)</td>
<td>Riccati Equation Solution P</td>
</tr>
<tr>
<td>XX(9), XXO(9)</td>
<td>State Vector x</td>
</tr>
</tbody>
</table>

| Lateral Mode | |
| ALPY, ALPYMAX | Prescribed Nonlinearity α |
| APY(11,11), APYO(11) | State Space System Matrix A' |
| BRBY(11,11), RIPX(11,11), R2IY(2) | System Weighting Matrix S, R₁ᵀ, R₂ |
### AVY(6,6), BVY(6,2), DQCY(6)
Vestibular State Space Matrices $A_V, B_V, D_V QC_V$

### K1Y(2,6), K2Y(2,3), K3Y(2)
Feedback Matrices $K_1, K_2, K_3$

### ZY(11,11)
Neurocomputing Solver Bi-Polar Vectors Z

### PY(11,11), PYVEC(66)
Riccati Equation Solution P

### XY(9), XYO(9)
State Vector $x$

---

#### Yaw Mode

**ALPR, ALPRMAX**
Prescribed Nonlinearity $\alpha$

**APR(5,5), APRO(5)**
State Space System Matrix $A'$

**BRBR(5,5), R1PR(5,5), R2IR**
System Weighting Matrix $S, R_1', R_2$

**AVR(3,3), BVR(3), DQCR(3), DQDR**
Vestibular State Space Matrices $A_V, B_V, D_V QC_V, D_V QD_V$

**K1R(3), K2R, K3R**
Feedback Matrices $K_1, K_2, K_3$

**ZR(5,5)**
Neurocomputing Solver Bi-Polar Vectors Z

**PR(5,5), PRVEC(15)**
Riccati Equation Solution P

**XR(4), XRO(4)**
State Vector $x$

---

#### Roll Mode

**ALPP, ALPPMAX**
Prescribed Nonlinearity $\alpha$

**APP(5,5), APPO(5)**
State Space System Matrix $A'$

**BRBP(5,5), R1PP(5,5), R2IP**
System Weighting Matrix $S, R_1', R_2$

**AVP(3,3), BVP(3), DQCP(3), DQDP**
Vestibular State Space Matrices $A_V, B_V, D_V QC_V, D_V QD_V$

**K1P(3), K2P, K3P**
Feedback Matrices $K_1, K_2, K_3$

**ZP(5,5)**
Neurocomputing Solver Bi-Polar Vectors Z

**PP(5,5), PPVEC(15)**
Riccati Equation Solution P

**XP(4), XPO(4)**
State Vector $x$

---

#### Pitch Mode

**ALPQ, ALPQMAX**
Prescribed Nonlinearity $\alpha$

**APQ(5,5), APQO(5)**
State Space System Matrix $A'$

**BRBQ(5,5), R1PQ(5,5), R2IQ**
System Weighting Matrix $S, R_1', R_2$

**AVQ(3,3), BVQ(3), DQCQ(3), DQDQ**
Vestibular State Space Matrices $A_V, B_V, D_V QC_V, D_V QD_V$

**K1Q(3), K2Q, K3Q**
Feedback Matrices $K_1, K_2, K_3$

**ZQ(5,5)**
Neurocomputing Solver Bi-Polar Vectors Z

**PQ(5,5), PQVEC(15)**
Riccati Equation Solution P

**XQ(4), XQO(4)**
State Vector $x$

---

#### Heave Mode

**ALPZ, ALPZMAX**
Prescribed Nonlinearity $\alpha$

**APZ(6,6), APZO(6)**
State Space System Matrix $A'$

**BRBZ(6,6), R1PZ(6,6)**
System Weighting Matrix $S, R_1'$

**AVZ(2,2), BVZ(2)**
Vestibular State Space Matrices $A_V, B_V$

**K1Z(2), K2Z(2), K3Z**
Feedback Matrices $K_1, K_2, K_3$

**ZZ(6,6)**
Neurocomputing Solver Bi-Polar Vectors Z

**PZ(6,6), PZVEC(21)**
Riccati Equation Solution P

**XZ(5), XZO(5)**
State Vector $x$

---

#### Nonlinear Gains and Turbulence

**GX4,GY4,GZ4,GP4,GQ4,GR4**
In-Flight Polynomial Scaling Coefficients

**AMX4,BMX4**
In-Flight Translational and Rotational Limits

**GZ40,GZ4S,AMX40,AMX4S**
On-Ground Scaling Coefficients and Limits

**GT4, G2D0, G2D1**
Augmented Turbulence Acceleration Gain

**Augmented Turbulence Filter Coefficients (Denominator)**
| G2N0, G2N1, G2N2 | Augmented Turbulence Filter Coefficients (Numerator) |
2. Program Listing

Followed is the program listing of the on-line implementation of the augmented nonlinear washout algorithm.

2.1. gainopt4.f

C******************************************************************************
C NONLINEAR ALGORITHM NONLINEAR GAIN SUBROUTINE.
C THE INPUT IS FIRST LIMITED AND THEN SCALED BY A POLYNOMIAL.
C******************************************************************************

SUBROUTINE GAINOPT4

INCLUDE 'optint3.com'

INCLUDE 'nopt4.com'

REAL AA4(3), BA4(3)

C

C Take Absolute Value of Input
C
AA4(1) = ABS(A2(1))
AA4(2) = ABS(A2(2))
AA4(3) = ABS(A2(3))
BA4(1) = ABS(BETAAD(1))
BA4(2) = ABS(BETAAD(2))
BA4(3) = ABS(BETAAD(3))

C

C Limit Translational and Rotational Inputs
C
AAM = MAX(AA4(1), AA4(2), AA4(3))
BAM = MAX(BA4(1), BA4(2), BA4(3))
IF (AAM.GT.AMX4) THEN
    RATIO = AMX4/AAM
    AA4(1) = AA4(1)*RATIO
    AA4(2) = AA4(2)*RATIO
    AA4(3) = AA4(3)*RATIO
END IF
IF (BAM.GT.BMX4) THEN
    RATIO = BMX4/BAM
    BA4(1) = BA4(1)*RATIO
    BA4(2) = BA4(2)*RATIO
    BA4(3) = BA4(3)*RATIO
END IF

C

C Perform Nonlinear Scaling of Inputs
C
A2N(1) = (GX4(1)*AA4(1)+GX4(2)*AA4(1)**2.+GX4(3)*AA4(1)**3.)*SIGN(1.,A2(1))
A2N(2) = (GY4(1)*AA4(2)+GY4(2)*AA4(2)**2.+GY4(3)*AA4(2)**3.)*SIGN(1.,A2(2))
A2N(3) = (GZ4(1)*AA4(3)+GZ4(2)*AA4(3)**2.+GZ4(3)*AA4(3)**3.)*SIGN(1.,A2(3))
\[ \text{BADN}(1) = (GP4(1) \times BA4(1) + GP4(2) \times BA4(1)^2 + GP4(3) \times BA4(1)^3) \times \text{SIGN}(1, \text{BETAAD}(1)) \]

\[ \text{BADN}(2) = (GQ4(1) \times BA4(2) + GQ4(2) \times BA4(2)^2 + GQ4(3) \times BA4(2)^3) \times \text{SIGN}(1, \text{BETAAD}(2)) \]

\[ \text{BADN}(3) = (GR4(1) \times BA4(3) + GR4(2) \times BA4(3)^2 + GR4(3) \times BA4(3)^3) \times \text{SIGN}(1, \text{BETAAD}(3)) \]

RETURN

END

C
2.2. integ4.f

SUBROUTINE INTEG4

INCLUDE 'optint3.com'

INCLUDE 'comint2.com'

INCLUDE 'wcom2.com'

INCLUDE 'matrix1c.com'

INCLUDE 'nopt4.com'

SURGE FILTER OUTPUT ASI(1) & BSDT(2)

ASI(1) = -K1X(2,1)*XX(1) - K1X(2,2)*XX(2) - K1X(2,3)*XX(3)
    x - K1X(2,4)*XX(4) - K1X(2,5)*XX(5) - K1X(2,6)*XX(6)
    x - K2X(2,1)*XX(7) - K2X(2,2)*XX(8) - K2X(2,3)*XX(9)
    x - K3X(2)*A2N(1)

BSDT(2) = -K1X(1,1)*XX(1) - K1X(1,2)*XX(2) - K1X(1,3)*XX(3)
    x - K1X(1,4)*XX(4) - K1X(1,5)*XX(5) - K1X(1,6)*XX(6)
    x - K2X(1,1)*XX(7) - K2X(1,2)*XX(8) - K2X(1,3)*XX(9)
    x - K3X(1)*A2N(1)

SWAY FILTER OUTPUT ASI(2) & BSDT(1)

ASI(2) = -K1Y(2,1)*XY(1) - K1Y(2,2)*XY(2) - K1Y(2,3)*XY(3)
    x - K1Y(2,4)*XY(4) - K1Y(2,5)*XY(5) - K1Y(2,6)*XY(6)
    x - K2Y(2,1)*XY(7) - K2Y(2,2)*XY(8) - K2Y(2,3)*XY(9)
    x - K3Y(2)*A2N(2)

BSDT(1) = -K1Y(1,1)*XY(1) - K1Y(1,2)*XY(2) - K1Y(1,3)*XY(3)
    x - K1Y(1,4)*XY(4) - K1Y(1,5)*XY(5) - K1Y(1,6)*XY(6)
    x - K2Y(1,1)*XY(7) - K2Y(1,2)*XY(8) - K2Y(1,3)*XY(9)
    x - K3Y(1)*A2N(2)

HEAVE FILTER OUTPUT ASI(3)

ASI(3) = -(K1Z(1)*XZ(1) - K1Z(2)*XZ(2) - K2Z(1)*XZ(3)
    x - K2Z(2)*XZ(4) - K2Z(3)*XZ(5) - K3Z*A2N(3))

YAW FILTER OUTPUT BSDR(3)

BSDR(3) = -K1R(1)*XR(1) - K1R(2)*XR(2) - K1R(3)*XR(3)
    x - K2R*XZ(4) - K3R*BADN(3)

PITCH FILTER OUTPUT BSDR(2)

BSDR(2) = -K1P(1)*XP(1) - K1P(2)*XP(2) - K1P(3)*XP(3)
    x - K2P*XP(4) - K3P*BADN(2)

ROLL FILTER OUTPUT BSDR(1)

BSDR(1) = -K1Q(1)*XQ(1) - K1Q(2)*XQ(2) - K1Q(3)*XQ(3)
LIMIT THE ANGULAR RATE IN THE CROSS-OVER TILT CHANNEL.

THE REAL TILT POSITION WILL BE DETERMINED BY BOTH THE DESIRED POSITION AND THE DIFFERENCE BETWEEN THE DESIRED AND REAL TILT POSITION.

DO K=1,2
   BETASR(K)=BETASRO(K)+DT*BSDRO(K)
   BETAST(K)=BETASTO(K)+DT*BSDTO(K)
   DIF(K)=0.005*(BETAST(K)-BETASTLO(K))+(BETAST(K)-BETASTO(K))
   BETASTL(K)=BETASTLO(K)+MAX(-BDLIM*DT,MIN(BDLIM*DT,DIF(K)))
END DO

COMPUTE THE TILT ANGULAR VELOCITY

BSDTL(K)=(BETASTL(K)-BETASTLO(K))/DT

BSDRO(K) = BSDR(K)
BETASRO(K)=BETASR(K)
BETASTO(K)=BETAST(K)
BETASTLO(K)=BETASTL(K)
BSDTO(K)=BSDT(K)

END DO

COMBINE THE TILT AND ROTATIONAL CHANNELS TO OBTAIN THE DESIRED ANGULAR POSITION

BETAS1(1)=BETASTL(1)+BETASR(1)
BETAS1(2)=BETASTL(2)+BETASR(2)
BETAS1(3)=XR(4)

USE DIFFERENCE BETWEEN DESIRED BETAST AND REAL BETAST TO GENERATE ADDITIONAL LINEAR RESPONSE AND ACHIEVE COORDINATION BETWEEN THE LINEAR AND TILT CHANNELS

SSI1(1)=XX(8)+RRS(3)*(BETAST(2)-BETASTL(2))
SSI1(2)=XY(8)-RRS(3)*(BETAST(1)-BETASTL(1))
SSI1(3)=XZ(4)+XKA*SIN((WB+XKG*VGSPD)*T)

FOR ON-GROUND MOTION, ADD RUNWAY ROUGHNESS EFFECT AMPLITUDE IS FAIRED UPON TOUCHDOWN OR TAKEOFF

SSI1(3)=XZ(4)+XKA*SIN((WB+XKG*VGSPD)*T)

Swap Variables To Match Modified Algorithm For Input to JACKDRVR

XDD = ASI(1)
YDD = ASI(2)
ZDD = ASI(3)
XD = XX(9)
YD = XY(9)
ZD = XZ(5)
X = SSI1(1)
Y = SSI1(2)
Z = SSI1(3)
PHI = BETAS1(1)
THE = BETAS1(2)
PSI = BETAS1(3)
PHID = BSDTL(1)+BSDR(1)
THED = BSDTL(2)+BSDR(2)
PSID = BSDR(3)

RETURN
END
2.3.  invplf

C  THIS SUBROUTINE WILL UNDERTAKE AN INVERSE TRANSFORMATION DEVELOPED
C  BY THE NEWTON-RAPHSON TECHNIQUE. LEG EXTENSIONS WILL BE TRANSFORMED
C  TO THE DEGREES OF FREEDOM. THIS INVERSE TRANSFORMATION IS PERFORMED
C  BY AN ITERATIVE METHOD DENOTED AS NEWTON-RAPHSON TECHNIQUE.
C  ITERATIONS ARE TERMINATED WHEN THE DIFFERENCE BETWEEN TWO SUBSEQUENT
C  ITERATIONS IS LESS THAN SOME ERROR CRITERION.

C  RLI IS THE LEG EXTENSIONS.
C  SSI IS THE TRANSLATIONAL DISPLACEMENT OF THE PLATFORM.
C  BETAS IS THE ANGULAR DISPLACEMENT OF THE PLATFORM.
C  XS,YM,ZM: COORDINATES OF THE FIXED ENDS OF THE LEGS.
C  XM,YM,ZM: COORDINATES OF THE MOVING ENDS OF THE LEGS.
C  AAIS,BBII: GEOMETRY OF THE MOTION SYSTEM.
C  SSIIN: INITIAL TRANSLATIONAL DISPLACEMENT.
C  RRS: VECTOR NOT USED BY THIS SUBROUTINE.
C  LENEUT: LENGTH OF LEGS IN NEUTRAL POSITION.

SUBROUTINE INVPLF
  REAL RAML(6),XS(6),YS(6),ZS(6),XM(6),YM(6),ZM(6)
  REAL F(6),PFX(6),PFY(6),PFZ(6),PFS(6),PFT(6),PFP(6),
    + A(3,3),ZEBRA(36),P
  INCLUDE 'matrix1c.com'
  DATA IFLAG/0/
  C***********************************************************************
  C INITIALIZE SIMULATOR POSITION.
  DATA X/0./,Y/0./,Z/0./,P/0./,T/0./,S/0./
  C***********************************************************************
  C SAVE
  IF(IFLAG.EQ.0) THEN
    DO JACK=1,6
      XM(JACK)=AAIS(1,JACK)
      YM(JACK)=AAIS(2,JACK)
      ZM(JACK)=AAIS(3,JACK)
      XS(JACK)=BBII(1,JACK)
      YS(JACK)=BBII(2,JACK)
      ZS(JACK)=BBII(3,JACK)
    END DO
    IFLAG=1
  END IF
  C***************************************************************
  C X=SSI(1)+SSIIN(1)
  C Y=SSI(2)+SSIIN(2)
  C Z=SSI(3)+SSIIN(3)
  C P=BETAS(1)
  C T=BETAS(2)
  C S=BETAS(3)
  C***************************************************************
  DO JACK=1,6
    RAML(JACK)=RLI(JACK)+LENEUT
  END DO
IT=0
CONTINUE
A(1,1)=COS(S)*COS(T)
A(1,2)=SIN(S)*COS(T)
A(1,3)=-SIN(T)
A(2,1)=COS(S)*SIN(T)*SIN(P)-SIN(S)*COS(P)
A(2,2)=SIN(S)*SIN(T)*SIN(P)+COS(S)*COS(P)
A(2,3)=COS(T)*SIN(P)
A(3,1)=COS(S)*SIN(T)*COS(P)+SIN(S)*SIN(P)
A(3,2)=SIN(S)*SIN(T)*COS(P)-COS(S)*SIN(P)
A(3,3)=COS(T)*COS(P)
DO 17 I=1,6
+2.*(X-XS(I))*(XM(I)*A(1,1)+YM(I)*A(2,1)+ZM(I)*A(3,1))
+2.*(Y-YS(I))*(XM(I)*A(1,2)+YM(I)*A(2,2)+ZM(I)*A(3,2))
+2.**(Z-ZS(I))*(XM(I)*A(1,3)+YM(I)*A(2,3)+ZM(I)*A(3,3))
+2.*(X**2.+Y**2.+Z**2.+RAML(I)**2.**(Z-ZS(I))
PFX(I)=2.*((X+XM(I)*A(1,1)+YM(I)*A(2,1)+ZM(I)*A(3,1)-XS(I))
PFY(I)=2.*((Y+XM(I)*A(1,2)+YM(I)*A(2,2)+ZM(I)*A(3,2)-YS(I))
PFZ(I)=2.*((Z+XM(I)*A(1,3)+YM(I)*A(2,3)+ZM(I)*A(3,3)-ZS(I))
PFS(I)=2.*(X-XS(I))*(XM(I)*A(1,1)+YM(I)*A(2,1)+ZM(I)*A(3,1))
+2.*(Y-YS(I))*(XM(I)*A(1,2)+YM(I)*A(2,2)+ZM(I)*A(3,2))
+2.*(Z-ZS(I))*(XM(I)*A(1,3)+YM(I)*A(2,3)+ZM(I)*A(3,3))
PFT(I)=2.*(X-XS(I))*(XM(I)*A(1,2)+YM(I)*A(2,2)+ZM(I)*A(3,2))
+2.*(Y-YS(I))*(XM(I)*A(1,1)+YM(I)*A(2,1)+ZM(I)*A(3,1))
+2.*(Z-ZS(I))*(XM(I)*A(1,3)+YM(I)*A(2,3)+ZM(I)*A(3,3))
+2.*(X**2.+Y**2.+Z**2.+RAML(I)**2.**(Z-ZS(I))
PFP(I)=2.*(X-XS(I))*(YM(I)*A(3,1)-ZM(I)*A(2,1))
+2.*(Y-YS(I))*(YM(I)*A(3,2)-ZM(I)*A(2,2))
+2.*(Z-ZS(I))*(YM(I)*A(3,3)-ZM(I)*A(2,3))
Continue
DO 1 N=1,6
ZEBRA(N) =PFX(N)
ZEBRA(N+6) =PFY(N)
ZEBRA(N+12) =PFZ(N)
ZEBRA(N+18) =PFS(N)
ZEBRA(N+24) =PFT(N)
ZEBRA(N+30) =PFP(N)
CONTINUE
N=6
CALL SIMQ(ZEBRA,F,N,KS)
IF(KS.EQ.1) THEN
WRITE(*,*) ' 1 MATRIX IS SINGULAR'
GOTO 22
END IF
IT=IT+1
IF(IT.EQ.51) GO TO 22
X=X-F(1)
Y=Y-F(2)
Z=Z-F(3)
S=S-F(4)
T=T-F(5)
P=P-F(6)
ZLIM1=0.01
ZLIM2=0.1/57.296
IF(MAX(ABS(F(1)),ABS(F(2)),ABS(F(3))).GT.ZLIM1) GO TO 9

48
IF(MAX(ABS(F(4)), ABS(F(5)), ABS(F(6))).GT.ZLIM2) GO TO 9

22   SSI(1)=X
     SSI(2)=Y
     SSI(3)=Z
     BETAS(1)=P
     BETAS(2)=T
     BETAS(3)=S
RETURN
END
2.4. jackdrvr.f

C COMPILE THE ACTUATOR EXTENSION COMMANDS BASED ON POSITION
C IN INERTIAL FRAME.
C
SUBROUTINE JACKDRVR

INCLUDE 'matrix1c.com'

INCLUDE 'comint2.com'

INCLUDE 'optint3.com'

REAL DUMMY31A(3,1),LLIS(3,3),L1(6),L2(6),L3(6),LENGTHTOT(6)

REAL LLIMH,LLIML

C
C********** Langley VMS motion system geometry **********

DATA LENEUT/3.2649/

DATA RRS/0.0254,-0.635,-2.1946/

DATA AAIS/2.1117179, 0.0762, 0.0,
     x 2.1117179, -0.0762, 0.0,
     x -0.98986594, -1.8669, 0.0,
     x -1.12184942, -1.7907, 0.0,
     x -1.12184942, 1.7907, 0.0,
     x -0.98986594, 1.8669, 0.0/

DATA BBII/ 1.5021179, 1.9812, 2.58064,
     x 1.5021179, -1.9812, 2.58064,
     x 0.96471232, -2.29147116, 2.58064,
     x -2.46682768, -0.31027116, 2.58064,
     x -2.46682768, 0.31027116, 2.58064,
     x 0.96471232, 2.29147116, 2.58064/

DATA RLI/6*0./, SSI/3*0.0/, BETAS/3*0.0/

C********************************************************
C
C     McFadden Actuator Stroke Limit
C     (When Used Replace LLIMH/LLIML with LLIM)

C     Langley VMS Actuator Stroke Limits
C
C     DATA LLIMH/0.7864/,LLIML/0.6487/

DATACMAX/0.7/

DATA FLAG/6*0/,FLAG2/0/,IT2/400/,NC2/400/,SUMFLAG/0/

DATA RLID/6*0./,RLIDD/6*0./,RLIO/6*0./,RLIDO/6*0./

C
C      EXACT ANGLE COMPUTATIONS
C
SINPHI = SIN(PHI)

SINTH  = SIN(THE)

SINPSI = SIN(PSI)

COSPHI = COS(PHI)

COSTH  = COS(THE)

COSPSI = COS(PSI)
FORM LLIS TRANSFORMATION MATRIX

\[
\begin{align*}
LLIS(1,1) &= \cos\psi \cos\theta \\
LLIS(2,1) &= \sin\psi \cos\theta \\
LLIS(3,1) &= -\sin\theta \\
LLIS(1,2) &= \cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi \\
LLIS(2,2) &= \sin\psi \sin\theta \sin\phi + \cos\psi \cos\phi \\
LLIS(3,2) &= \cos\theta \sin\phi \\
LLIS(1,3) &= \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \\
LLIS(2,3) &= \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi \\
LLIS(3,3) &= \cos\theta \cos\phi
\end{align*}
\]

Compute Leg Extensions

DO JACK = 1, 6
   CALL VMULT(LLIS, AAIS(1, JACK), DUMMY31A, 3, 3, 1)
   L1(JACK) = DUMMY31A(1, 1) + X - BBII(1, JACK)
   L2(JACK) = DUMMY31A(2, 1) + Y - BBII(2, JACK)
   L3(JACK) = DUMMY31A(3, 1) + Z - BBII(3, JACK)
   LENGHTOT(JACK) = SQRT(L1(JACK)**2 + L2(JACK)**2 + L3(JACK)**2)
   LI(JACK) = LENGHTOT(JACK) - LENEUT
END DO

C*************** JACK EXTENSION LIMITING ***********************

DO JACK = 1, 6
   IF(FLAG(JACK).EQ.1) GOTO 5
   Avail. Length for Same +/- Extension Limits
   LAVAIL = LLIM - RLI(JACK)*SIGN(1., RLID(JACK))
   Avail. Length for Different +/- Extension Limits
   IF (RLI(JACK).GT.0.) THEN
      LAVAIL = LLIMH - RLI(JACK)*SIGN(1., RLID(JACK))
   ELSE
      LAVAIL = LLIML - RLI(JACK)*SIGN(1., RLID(JACK))
   END IF
   BRAKE(JACK) = ABS(RLID(JACK))**2 - 1.98*ACMAX*LAVAIL
   IF(BRAKE(JACK).LT.0.) GOTO 5
   FLAG(JACK) = 1
   VLEAD = RLID(JACK)
   FLAG2 = 1
   DO JK = 1, 6
      IF(FLAG(JK).EQ.0) THEN
         RATIO(JK) = ABS(RLID(JK)/VLEAD)
      ELSE
         RATIO(JK) = 1.
      END IF
   END DO
5 END DO

SUMFLAG = FLAG(1) + FLAG(2) + FLAG(3) + FLAG(4) + FLAG(5) + FLAG(6)
When brake is set, determine if brake should be released

\[ \text{DXX} = \text{ABS}(X) - \text{ABS}(	ext{SSI}(1)) \]
\[ \text{DYY} = \text{ABS}(Y) - \text{ABS}(	ext{SSI}(2)) \]
\[ \text{DZZ} = \text{ABS}(Z) - \text{ABS}(	ext{SSI}(3)) \]
\[ \text{DPHI} = \text{ABS}(	ext{PHI}) - \text{ABS}(	ext{BETAS}(1)) \]
\[ \text{DTHE} = \text{ABS}(	ext{THE}) - \text{ABS}(	ext{BETAS}(2)) \]
\[ \text{DPSI} = \text{ABS}(	ext{PSI}) - \text{ABS}(	ext{BETAS}(3)) \]
\[ \text{DMAX} = \max(\text{DXX}, \text{DYY}, \text{DZZ}, \text{DPHI}, \text{DTHE}, \text{DPSI}) \]
\[ \text{DMIN} = \min(\text{DXX}, \text{DYY}, \text{DZZ}, \text{DPHI}, \text{DTHE}, \text{DPSI}) \]

\[ \text{IF}((\text{DMAX} \leq 1.\text{E}-5) \text{.AND.} (\text{DMIN} \leq -0.01)) \ THEN \]
\[ \text{ID} = 0 \]
ELSE
\[ \text{ID} = 1 \]
END IF

\[ \text{DO} \ \text{JACK}=1,6 \]
\[ \text{IF}((\text{FLAG(JACK)}).EQ.0) \ \text{GOTO} \ 20 \]
\[ \text{RLIDD(JACK)} = -\text{ACMAX} \times \text{SIGN}(1., \text{RLID(JACK)}) \]
\[ \text{RLID(JACK)} = \text{RLID(JACK)} + \text{RLIDD(JACK)} \times H \]
\[ \text{GOTO} \ 30 \]
20 \[ \text{IF}((\text{SUMFLAG}).EQ.0) \ \text{GOTO} \ 50 \]
\[ \text{IF}((\text{ABS}(\text{RLID(JACK)})).GT.0.001) \ \text{THEN} \]
\[ \text{RLIDD(JACK)} = -\text{ACMAX} \times \text{SIGN}(1., \text{RLID(JACK)}) \times \text{RATIO(JACK)} \]
\[ \text{RLID(JACK)} = \text{RLID(JACK)} + \text{RLIDD(JACK)} \times H \]
\[ \text{END IF} \]
30 \[ \text{IF}((\text{ABS}(\text{RLID(JACK)})).GT.(\text{ACMAX} \times H)) \ \text{THEN} \]
\[ \text{RLI(JACK)} = \text{RLI(JACK)} + \text{RLID(JACK)} \times H \]
ELSE
\[ \text{RLID(JACK)} = 0. \]
\[ \text{FLAG(JACK)} = 0 \]
END IF
\[ \text{GOTO} \ 70 \]
50 \[ \text{IF}((\text{FLAG2}).EQ.0) \ \text{GOTO} \ 60 \]
\[ \text{IF}((\text{ID}.NE.0) \ \text{GOTO} \ 70 \]
\[ \text{FLAG2} = 0 \]
\[ \text{IT2} = 0 \]
60 \[ \text{RLI(JACK)} = \text{RLI(JACK)} + \]
\[ + (1. - \cos(3.1416/2.*\text{IT2}/\text{NC2})) \times (\text{LI(JACK)} - \text{RLI(JACK)}) \]
\[ \text{IF}((\text{JACK}.EQ.6).AND.(\text{IT2}.LT.\text{NC2})) \ \text{IT2} = \text{IT2} + 1 \]
70 \[ \text{END DO} \]

\[ \text{DO} \ \text{JACK}=1,6 \]

\[ \text{Same +/- Extension limits} \]
\[ \text{RLI(JACK)} = \text{MIN}(0.999*\text{LLIM}, \text{MAX}(-0.999*\text{LLIM}, \text{RLI(JACK)})) \]

\[ \text{Different +/- Extension Limits} \]
\[ \text{RLI(JACK)} = \text{MIN}(0.999*\text{LLIMH}, \text{MAX}(-0.999*\text{LLIML}, \text{RLI(JACK)})) \]
\[ \text{RLIDD(JACK)} = (\text{RLI(JACK)} - \text{RLIO(JACK)}) / H \]
\[ \text{RLIDO(JACK)} = (\text{RLID(JACK)} - \text{RLIDO(JACK)}) / H \]
\[ \text{RLIO(JACK)} = \text{RLI(JACK)} \]
\[ \text{RLIDO(JACK)} = \text{RLID(JACK)} \]
END DO
2.5.  liba.f

C
C COMPUTE THE TRANSFORMATION MATRICES LIA AND TS
C
SUBROUTINE LIBA

INCLUDE 'optint3.com'

REAL LIA(3,3), TA(3,3)

DATA TA/1.,3*0.0,1.,3*0.0,1./

C
C    EXACT ANGLE COMPUTATIONS
C
SINPHI = SIN(BETAA(1))
SINTH  = SIN(BETAA(2))
SINPSI = SIN(BETAA(3))
COSPHI = COS(BETAA(1))
COSTH  = COS(BETAA(2))
COSPSI = COS(BETAA(3))
TANTH  = SINTH/COSTH

C
C    FORM LIA TRANSFORMATION MATRIX
C
LIA(1,1) = COSPSI*COSTH
LIA(2,1) = SINPSI*COSTH
LIA(3,1) = -SINTH
LIA(1,2) = COSPSI*SINTH*SINPHI - SINPSI*COSPHI
LIA(2,2) = SINPSI*SINTH*SINPHI + COSPSI*COSPHI
LIA(3,2) = COSTH*SINPHI
LIA(1,3) = COSPSI*SINTH*COSPHI + SINPSI*SINPHI
LIA(2,3) = SINPSI*SINTH*COSPHI - COSPSI*SINPHI
LIA(3,3) = COSTH*COSPHI

C
C    FORM TA TRANSFORMATION MATRIX
C
TA(1,2)=SINPHI*TANTH
TA(1,3)=COSPHI*TANTH
TA(2,2)=COSPHI
TA(2,3)=-SINPHI
TA(3,2)=SINPHI/COSTH
TA(3,3)=COSPHI/COSTH

C
C    Compute Inertial Accleration A2
C
CALL VMULT(LIA,ACA,A2,3,3,1)

C
C    Compute euler Rates BETAAD
C
CALL VMULT(TA,WAA,BETAAD,3,3,1)

RETURN
END
2.6. newopt4.f

C***** SUBROUTINE NEWOPT4.F *****
C
C NONLINEAR WASHOUT ALGORITHM: ANG. VELOCITY DEVELOPMENT.
C GIVEN A/C ACCELS ACA AND EULER RATES BETAAD,
C COMPUTE SIMULATOR INERTIAL DISPLACEMENT AND EULER ANGLES.
C
SUBROUTINE NEWOPT4(MODE)
INCLUDE 'comint2.com'
INCLUDE 'wcom2.com'
INCLUDE 'optint3.com'
INCLUDE 'nopt4.com'
INCLUDE 'matrix1c.com'
C
DATA IRESET/0/,IHOLD/0/
C
C Compute Fairing Parameters
C
SQWASH=SQWASHP*EA+SQWASHI*(1.0-EA)
DELSQ=MAX(MIN(SQWASH-SQWASHP,1.0),-1.0)
SQWASHP=SQWASH+DELSQ
A=1.0-SQWASHP
AA=1.0-SQWASHI
C
C Fairing of Heave Nonlinear Gain and Limit
C
GZ4(1)=AA*GZ40(1)+SQWASHI*GZ4S(1)
GZ4(2)=AA*GZ40(2)+SQWASHI*GZ4S(2)
GZ4(3)=AA*GZ40(3)+SQWASHI*GZ4S(3)
AMX4=AA*AMX40+SQWASHI*AMX4S
C
C Fairing of Runway Roughness Amplitude
C
XKA=A*XKA0+SQWASHP*XKAS
C
IF(MODE.EQ.1) THEN
H = DT
C
C Set "old" variables for future use in HOLD and OPERATE modes
C
DO I=1,3
   A2NO(I)=0.
   BADNO(I)=0.
END DO
DO I=1,9
   XXO(I)=0.
   XYO(I)=0.
END DO

DO J=1,11
   DO I=1,11
      IF (I.GE.J) THEN
         PX(I,J)=PXVEC((J-1)*11-J*(J-1)/2+I)
         PX(J,I)=PX(I,J)
         PY(I,J)=PYVEC((J-1)*11-J*(J-1)/2+I)
         PY(J,I)=PY(I,J)
      END IF
   END DO
END DO

DO I=1,4
   XRO(I)=0.
   XPO(I)=0.
   XQO(I)=0.
END DO

DO J=1,5
   DO I=1,5
      IF (I.GE.J) THEN
         PR(I,J)=PRVEC((J-1)*5-J*(J-1)/2+I)
         PR(J,I)=PR(I,J)
         PP(I,J)=PPVEC((J-1)*5-J*(J-1)/2+I)
         PP(J,I)=PP(I,J)
         PQ(I,J)=PQVEC((J-1)*5-J*(J-1)/2+I)
         PQ(J,I)=PQ(I,J)
      END IF
   END DO
END DO

DO I=1,5
   XZO(I)=0.
END DO

DO J=1,6
   DO I=1,6
      IF (I.GE.J) THEN
         PZ(I,J)=PZVEC((J-1)*6-J*(J-1)/2+I)
         PZ(J,I)=PZ(I,J)
      END IF
   END DO
END DO

CALL RESETC2

C Set "old" variables for future use in HOLD and OPERATE modes
C
BETASRO(1)=PHI
BETASRO(2)=THE
BSDRO(1)=PHID
BSDRO(2)=THED

DO I=1,2
   XHO(I)=0.
ACAO(I)=0.
BSDTO(I)=0.
BETASTO(I)=0.
BETASTLO(I)=0.
XTO=0.
XT2O=0.
WGUSTO=0.
ACZT=0.
END DO

GO TO 1
END IF

IF(MODE.EQ.2) THEN
CALL WTRIM3
END IF

C

Compute Augmented Acceleration from W-Gust
C

IF(MODE.EQ.3) THEN
WGAV=0.5*(WGUST+WGUSTO)
XT=XTO+DT*(-G2D1*XTO+XT2O+(G2N1-G2D1*G2N2)*WGAV)
XTAV=0.5*(XT+XTO)
XT2=XT2O+DT*(-G2D0*XTAV+(G2N0-G2D0*G2N2)*WGAV)
ACZT=XT+G2N2*WGUST
XTO=XT
XT2O=XT2
WGUSTO=WGUST
C

(first-order turbulence model no longer used)
C

XT=XTO+DT*(-G1D0*XTO+(G1N0-G1D0*G1N1)*WGUSTO)
C
XTO=XT
C
WGUSTO=WGUST
C
ACZT=XT+G1N1*WGUST
END IF

CALL LIBA
CALL GAINOPT4
A2N(3)=A2N(3)+GT4*ACZT

IF(MODE.EQ.3) THEN
CALL NFILX
CALL NFILY
CALL NFILZ
CALL NFILR
CALL NFILP
CALL NFILQ
CALL STATE4
ELSE IF(MODE.EQ.2) THEN
CALL STATE4
CALL STATE4
CALL STATE4
CALL STATE4
CALL STATE4
CALL STATE4
END IF
IF(MODE.EQ.3) CALL INTEG4

1 RETURN
END
C
2.7. *nfilr.f*

Yaw Filter Neurocomputing Solver for the Riccati Equation

SUBROUTINE NFILR

INCLUDE 'comint2.com'

INCLUDE 'nopt4.com'

REAL SPR(5,5), SUMER, SUMPR, SUMPR
REAL PAPR(5,5), UPBR(5,5), UAPR(5,5), APUR(5,5)
REAL EUR(5,5), UR(5), PZR(5)

Compute Prescribed Nonlinearity Alpha

ALPR = PSI*Q2R*PSI
IF(ALPR.GE.ALPRMAX) ALPR = ALPRMAX
DO I=1,5
    APR(I,I) = APR(I) + ALPR
END DO

Start Training Iterations & Initialize Variables

DO L=1,3
    DO I=1,5
        DO J=1,5
            SPR(I,J) = 0.
            UPBR(I,J) = 0.
            PAPR(I,J) = 0.
            UAPR(I,J) = 0.
            APUR(I,J) = 0.
        END DO
    END DO
END DO

Compute Matrix Products SP and PA

DO I=1,5
    DO J=1,5
        DO K=1,5
            SPR(I,J) = SPR(I,J) + BRBR(I,K)*PR(K,J)
        END DO
    END DO
    DO K=1,5
        PAPR(I,1) = PAPR(I,1) + PR(I,K)*APR(K,1)
        PAPR(I,3) = PAPR(I,3) + PR(I,K)*APR(K,3)
        PAPR(I,5) = PAPR(I,5) + PR(I,K)*APR(K,5)
        PAPR(I,2) = PR(I,1)*APR(1,2) + PR(I,2)*APR(2,2)
        PAPR(I,4) = PR(I,4)*APR(4,4)
    END DO
END DO

Compute Error Signal v, Vector p and Matrix Product pz

Note: Matrix Product A'P is transpose of PA by symmetry of P
C
DO I=1,5
   UR(I)=0.
PZR(I)=0.
DO J=1,5
   IF(I.LE.J) THEN
      SUMER=0.
      DO K=1,5
         SUMER=SUMER+PR(I,K)*SPR(K,J)
      END DO
      EUR(I,J)=SUMER-PAPR(I,J)-PAPR(J,I)-R1PR(I,J)
      EUR(J,I)=EUR(I,J)
   END IF
   UR(I)=UR(I)+EUR(I,J)*ZR(L,J)
PZR(I)=PZR(I)+PR(I,J)*ZR(L,J)
END DO
END DO
C
C Compute Matrix Product Avz
C
DO I=1,2
   DO J=1,5
      DO K=1,3
         APUR(I,J)=APUR(I,J)+APR(I,K)*UR(K)*ZR(L,J)
      END DO
      APUR(I,J)=APUR(I,J)+APR(I,5)*UR(5)*ZR(L,J)
   END DO
END DO
DO I=3,4
   DO J=1,5
      APUR(I,J)=APUR(I,J)+APR(I,1)*UR(1)*ZR(L,J)
      DO K=3,5
         APUR(I,J)=APUR(I,J)+APR(I,K)*UR(K)*ZR(L,J)
      END DO
   END DO
END DO
DO J=1,5
   APUR(5,J)=APR(5,5)*UR(5)*ZR(L,J)
END DO
C
C Compute Matrix Product vzA
C
DO I=1,5
   DO K=1,5
      UAPR(I,1)=UAPR(I,1)+UR(I)*ZR(L,K)*APR(K,1)
      UAPR(I,3)=UAPR(I,3)+UR(I)*ZR(L,K)*APR(K,3)
      UAPR(I,5)=UAPR(I,5)+UR(I)*ZR(L,K)*APR(K,5)
   END DO
   UAPR(I,2)=UR(I)*ZR(L,2)*APR(2,2)
   UAPR(I,4)=UR(I)*ZR(L,4)*APR(4,4)
END DO
C
C Compute Matrix Product vp'S
C
DO I=1,5
   DO J=1,4
      \text{...}
   END DO
END DO
DO K=1,4
   UPBR(I,J)=UPBR(I,J)+UR(I)*PZR(K)*BRBR(K,J)
END DO
END DO
END DO

C Sum Avz + vzA + v'S and integrate to update Riccati Solution P
C
DO I=1,5
   DO J=1,5
      IF(I.LE.J) THEN
         SUMPR=APUR(I,J)+UAPR(I,J)-UPBR(I,J)
         IF(I.NE.J) THEN
            SUMPR=APUR(J,I)+UAPR(J,I)-UPBR(J,I)
         ELSE
            SUMPR=SUMPR
         END IF
      END IF
      PR(I,J)=PR(I,J)+MUR*(SUMPR+SUMPR)
      PR(J,I)=PR(I,J)
   END DO
END DO
END DO

RETURN
END
2.8. nfilp.f

SUBROUTINE NFILP

INCLUDE 'comint2.com'
INCLUDE 'nopt4.com'

REAL SPP(5,5),SUMEP,SUMPP,SUMPPT
REAL PAPP(5,5),UPBP(5,5),UAPP(5,5),APUP(5,5)
REAL EUP(5,5),UP(5),PZP(5)

Compute Prescribed Nonlinearity Alpha
ALPP=PHI*Q2P*PHI
IF(ALPP.GE.ALPPMAX) ALPP=ALPPMAX
DO I=1,5
  APP(I,I)=APP0(I)+ALPP
END DO

Start Training Iterations & Initialize Variables
DO L=1,3
  DO I=1,5
    DO J=1,5
      SPP(I,J)=0.
      UPBP(I,J)=0.
      PAPP(I,J)=0.
      UAPP(I,J)=0.
      APUP(I,J)=0.
    END DO
  END DO
  PAPP(I,2)=PP(I,1)*APP(1,2)+PP(I,2)*APP(2,2)
  PAPP(I,4)=PP(I,4)*APP(4,4)
END DO

Compute Matrix Products SP and PA
DO I=1,5
  DO J=1,5
    DO K=1,5
      SPP(I,J)=SPP(I,J)+BRBP(I,K)*PP(K,J)
    END DO
  END DO
  DO K=1,5
    PAPP(I,1)=PAPP(I,1)+PP(I,K)*APP(K,1)
    PAPP(I,3)=PAPP(I,3)+PP(I,K)*APP(K,3)
    PAPP(I,5)=PAPP(I,5)+PP(I,K)*APP(K,5)
  END DO
  PAPP(I,2)=PP(I,1)*APP(1,2)+PP(I,2)*APP(2,2)
  PAPP(I,4)=PP(I,4)*APP(4,4)
END DO

Compute Error Signal v, Vector p and Matrix Product pz
Note: Matrix Product A'P is transpose of PA by symmetry of P
DO I=1,5
   UP(I)=0.
   PZP(I)=0.
   DO J=1,5
      IF(I.LE.J) THEN
         SUMEP=0.
         DO K=1,5
            SUMEP=SUMEP+PP(I,K)*SPP(K,J)
         END DO
         EUP(I,J)=SUMEP-PAPP(I,J)-PAPP(J,I)-R1PP(I,J)
      END IF
      UP(I)=UP(I)+EUP(I,J)*ZP(L,J)
      PZP(I)=PZP(I)+PP(I,J)*ZP(L,J)
   END DO
END DO

Compute Matrix Product Avz

DO I=1,2
   DO J=1,5
      DO K=1,3
         APUP(I,J)=APUP(I,J)+APP(I,K)*UP(K)*ZP(L,J)
      END DO
      APUP(I,J)=APUP(I,J)+APP(I,5)*UP(5)*ZP(L,J)
   END DO
END DO

DO I=3,4
   DO J=1,5
      DO K=3,5
         APUP(I,J)=APUP(I,J)+APP(I,K)*UP(K)*ZP(L,J)
      END DO
      APUP(5,J)=APP(5,5)*UP(5)*ZP(L,J)
   END DO
END DO

Compute Matrix Product vzA

DO I=1,5
   DO K=1,5
      UAPP(I,1)=UAPP(I,1)+UP(I)*ZP(L,K)*APP(K,1)
      UAPP(I,3)=UAPP(I,3)+UP(I)*ZP(L,K)*APP(K,3)
      UAPP(I,5)=UAPP(I,5)+UP(I)*ZP(L,K)*APP(K,5)
   END DO
   UAPP(I,2)=UP(I)*ZP(L,2)*APP(2,2)
   UAPP(I,4)=UP(I)*ZP(L,4)*APP(4,4)
   END DO

Compute Matrix Product vp'S

DO I=1,5
   DO J=1,4
   END DO
   END DO

63
DO K=1,4
    UPBP(I,J)=UPBP(I,J)+UP(I)*PZP(K)*BRBP(K,J)
END DO
END DO
END DO

C Sum Avz + vzA + vp'S and integrate to update Riccati Solution P
C
DO I=1,5
    DO J=1,5
        IF(I.LE.J) THEN
            SUMPP=APUP(I,J)+UAPP(I,J)-UPBP(I,J)
        IF(I.NE.J) THEN
            SUMPPT=APUP(J,I)+UAPP(J,I)-UPBP(J,I)
        ELSE
            SUMPPT=SUMPP
        END IF
        PP(I,J)=PP(I,J)+MUP*(SUMPP+SUMPPT)
        PP(J,I)=PP(I,J)
    END IF
END DO
END DO
END DO
RETURN
END
2.9. *nfilq.f*

C Pitch Filter Neurocomputing Solver for the Riccati Equation
C
SUBROUTINE NFILQ

INCLUDE 'comint2.com'
INCLUDE 'nopt4.com'

REAL SPQ(5,5),SUMEQ,SUMPQ,SUMPQT
REAL PAPQ(5,5),UPBQ(5,5),UAPQ(5,5),APUQ(5,5)
REAL EUQ(5,5),UQ(5),PZQ(5)

C Compute Prescribed Nonlinearity Alpha
C
ALPQ=THE*Q2Q*THE
IF(ALPQ.GE.ALQMAX) ALPQ=ALPQMAX
DO I=1,5
   APQ(I,I)=APQO(I)+ALPQ
END DO

C Start Training Iterations & Initialize Variables
C
DO L=1,3
   DO I=1,5
      DO J=1,5
         SPQ(I,J)=0.
         UPBQ(I,J)=0.
         PAPQ(I,J)=0.
         UAPQ(I,J)=0.
         APUQ(I,J)=0.
      END DO
   END DO
   DO K=1,5
      PAPQ(I,1)=PAPQ(I,1)+PQ(I,K)*APQ(K,1)
      PAPQ(I,3)=PAPQ(I,3)+PQ(I,K)*APQ(K,3)
      PAPQ(I,5)=PAPQ(I,5)+PQ(I,K)*APQ(K,5)
      PAPQ(I,2)=PQ(I,1)*APQ(1,2)+PQ(I,2)*APQ(2,2)
      PAPQ(I,4)=PQ(I,4)*APQ(4,4)
   END DO
C
C Compute Matrix Products SP and PA
C
DO I=1,5
   DO J=1,5
      DO K=1,5
         SPQ(I,J)=SPQ(I,J)+BRBQ(I,K)*PQ(K,J)
      END DO
   END DO
C
C Compute Error Signal \( v \), Vector \( p \) and Matrix Product \( p_2 \)
Note: Matrix Product A'P is transpose of PA by symmetry of P

DO I=1,5
   UQ(I)=0.
   PZQ(I)=0.
   DO J=1,5
      IF(I.LE.J) THEN
         SUMEQ=0.
         DO K=1,5
            SUMEQ=SUMEQ+PQ(I,K)*SPQ(K,J)
         END DO
         EUQ(I,J)=SUMEQ-PAPQ(I,J)-PAPQ(J,I)-R1PQ(I,J)
         EUQ(J,I)=EUQ(I,J)
      END IF
      UQ(I)=UQ(I)+EUQ(I,J)*ZQ(L,J)
      PZQ(I)=PZQ(I)+PQ(I,J)*ZQ(L,J)
   END DO
END DO

C Compute Matrix Product Avz

DO I=1,5
   DO J=1,5
      DO K=1,3
         APUQ(I,J)=APUQ(I,J)+APQ(I,K)*UQ(K)*ZQ(L,J)
      END DO
      APUQ(I,J)=APUQ(I,J)+APQ(I,5)*UQ(5)*ZQ(L,J)
   END DO
END DO
DO I=3,4
   DO J=1,5
      DO K=3,5
         APUQ(I,J)=APUQ(I,J)+APQ(I,K)*UQ(K)*ZQ(L,J)
      END DO
      APUQ(I,J)=APUQ(I,J)+APQ(I,5)*UQ(5)*ZQ(L,J)
   END DO
END DO
DO J=1,5
   APUQ(5,J)=APQ(5,5)*UQ(5)*ZQ(L,J)
END DO

C Compute Matrix Product vzA

DO I=1,5
   DO K=1,5
      UAPQ(I,1)=UAPQ(I,1)+UQ(I)*ZQ(L,K)*APQ(K,1)
      UAPQ(I,3)=UAPQ(I,3)+UQ(I)*ZQ(L,K)*APQ(K,3)
      UAPQ(I,5)=UAPQ(I,5)+UQ(I)*ZQ(L,K)*APQ(K,5)
   END DO
   UAPQ(I,2)=UQ(I)*ZQ(L,1)*APQ(1,2)
   x
   UAPQ(I,4)=UQ(I)*ZQ(L,4)*APQ(4,4)
END DO

C Compute Matrix Product vp'S

DO I=1,5
END DO
DO J=1,4
  DO K=1,4
    UPBQ(I,J)=UPBQ(I,J)+UQ(I)*PZQ(K)*BRBQ(K,J)
  END DO
END DO
END DO

C  C
 C Sum Avz + vzA + vp'S and integrate to update Riccati Solution P
 C
  DO I=1,5
    DO J=1,5
      IF(I.LE.J) THEN
        SUMPQ=APUQ(I,J)+UAPQ(I,J)-UPBQ(I,J)
      END IF
      IF(I.NE.J) THEN
        SUMPQT=APUQ(J,I)+UAPQ(J,I)-UPBQ(J,I)
      ELSE
        SUMPQT=SUMPQ
      END IF
      PQ(I,J)=PQ(I,J)+MUQ*(SUMPQ+SUMPQT)
      PQ(J,I)=PQ(I,J)
    END IF
  END DO
END DO

END DO
RETURN
END
2.10. *nfilx.f*

**Surge Filter Neurocomputing Solver for the Riccati Equation**

SUBROUTINE NFILX

INCLUDE 'comint2.com'

INCLUDE 'nopt4.com'

REAL SPX(11,11),SUMEX,SUMPX,SUMPXT
REAL PAPX(11,11),UPBX(11,11),UAPX(11,11),APUX(11,11)
REAL EUX(11,11),UX(11),PZX(11)

Compute Prescribed Nonlinearity Alpha

ALPX=X*Q2X(1)*X+XD*Q2X(2)*XD
IF(ALPX.GT.ALPMAX) ALPX=ALPMAX
DO I=1,11
   APX(I,I)=APXO(I)+ALPX
END DO

Start Training Iterations & Initialize Variables

DO L=1,3
   DO I=1,11
      DO J=1,11
         SPX(I,J)=0.0
         PAPX(I,J)=0.0
         UPBX(I,J)=0.0
         UAPX(I,J)=0.0
         APUX(I,J)=0.0
      END DO
   END DO
   DO J=1,11
      DO K=1,6
         SPX(I,J)=SPX(I,J)+BRBX(I,K)*PX(K,J)
      END DO
      SPX(I,J)=SPX(I,J)+BRBX(I,9)*PX(9,J)
   END DO
   DO J=1,11
      DO K=1,6
         SPX(9,J)=SPX(9,J)+BRBX(9,K)*PX(K,J)
      END DO
      SPX(9,J)=SPX(9,J)+BRBX(9,9)*PX(9,J)
   END DO
END DO

Compute Matrix Product SP

DO I=1,6
   DO J=1,11
      DO K=1,6
         SPX(I,J)=SPX(I,J)+BRBX(I,K)*PX(K,J)
      END DO
   END DO
END DO

Compute Matrix Product PA
DO I=1,11
  DO J=1,6
    DO K=1,6
      PAPX(I,J)=PAPX(I,J)+PX(I,K)*APX(K,J)
    END DO
  END DO
  PAPX(I,7)=PX(I,7)*APX(7,7)
  PAPX(I,8)=PX(I,7)*APX(7,8)+PX(I,8)*APX(8,8)
  PAPX(I,9)=PX(I,8)*APX(8,9)+PX(I,9)*APX(9,9)
  DO K=1,6
    PAPX(I,10)=PAPX(I,10)+PX(I,K)*APX(K,10)
  END DO
  PAPX(I,10)=PAPX(I,10)+PX(I,10)*APX(10,10)
  DO K=1,6
    PAPX(I,11)=PAPX(I,11)+PX(I,K)*APX(K,11)
  END DO
  PAPX(I,11)=PAPX(I,11)+PX(I,11)*APX(11,11)
END DO

C

Compute Error Signal v, Vector p and Matrix Product vz

Note: Matrix Product A'P is transpose of PA by symmetry of P

DO I=1,11
  UX(I)=0.
  PZX(I)=0.
  DO J=1,11
    IF(I.LE.J) THEN
      SUMEX=0.
      DO K=1,11
        SUMEX=SUMEX+PX(I,K)*SPX(K,J)
      END DO
      EUX(I,J)=SUMEX-PAPX(J,I)-PAPX(I,J)-R1PX(I,J)
      EUX(J,I)=EUX(I,J)
    END IF
    UX(I)=UX(I)+EUX(I,J)*ZX(L,J)
    PZX(I)=PZX(I)+PX(I,J)*ZX(L,J)
  END DO
END DO

C

Compute Matrix Product Avz

DO I=1,6
  DO J=1,11
    DO K=1,6
      APUX(I,J)=APUX(I,J)+APX(I,K)*UX(K)*ZX(L,J)
    END DO
    APUX(I,J)=APUX(I,J)+APX(I,10)*UX(10)*ZX(L,J)
    APUX(I,J)=APUX(I,J)+APX(I,11)*UX(11)*ZX(L,J)
    APUX(I,J)=APUX(I,J)+APX(I,11)*UX(11)*ZX(L,J)
  END DO
END DO

DO J=1,11
  APUX(7,J)=APX(7,7)*UX(7)*ZX(L,J)
  APUX(7,J)=APX(7,7)*UX(7)*ZX(L,J)
  APUX(8,J)=APX(8,8)*UX(8)*ZX(L,J)
  APUX(8,J)=APX(8,8)*UX(8)*ZX(L,J)
  APUX(9,J)=APX(9,9)*UX(9)*ZX(L,J)
  APUX(9,J)=APX(9,9)*UX(9)*ZX(L,J)
APUX(10,J)=APX(10,10)*UX(10)*ZX(L,J)
APUX(11,J)=APX(11,11)*UX(11)*ZX(L,J)
END DO

C
C Compute Matrix Product vzA
C
DO I=1,11
  DO J=1,6
    DO K=1,6
      UAPX(I,J)=UAPX(I,J)+UX(I)*ZX(L,K)*APX(K,J)
    END DO
  END DO
UAPX(I,7)=UX(I)*ZX(L,7)*APX(7,7)
UAPX(I,8)=UX(I)*ZX(L,7)*APX(7,8)
  DO K=1,6
      UAPX(I,8)=UAPX(I,8)+UX(I)*ZX(L,K)*APX(K,8)
  END DO
UAPX(I,9)=UX(I)*ZX(L,8)*APX(8,9)
  DO K=1,6
      UAPX(I,9)=UAPX(I,9)+UX(I)*ZX(L,K)*APX(K,9)
  END DO
UAPX(I,10)=UAPX(I,10)+UX(I)*ZX(L,10)*APX(10,10)
  DO K=1,6
      UAPX(I,10)=UAPX(I,10)+UX(I)*ZX(L,K)*APX(K,10)
  END DO
END DO

C
C Compute Matrix Product vp'S
C
DO I=1,11
  DO J=1,6
    DO K=1,6
      UPBX(I,J)=UPBX(I,J)+UX(I)*PZX(K)*BRBX(K,J)
    END DO
    UPBX(I,J)=UPBX(I,J)+UX(I)*PZX(9)*BRBX(9,J)
  END DO
  DO K=1,6
      UPBX(I,9)=UPBX(I,9)+UX(I)*PZX(K)*BRBX(K,9)
  END DO
    UPBX(I,9)=UPBX(I,9)+UX(I)*PZX(9)*BRBX(9,9)
  END DO
END DO

C
C Sum Avz + vzA + vp'S and integrate to update Riccati Solution P
C
DO I=1,11
  IF(I.LE.J) THEN
    SUMPX=APUX(I,J)+UAPX(I,J)-UPBX(I,J)
  ELSEIF(I.NE.J) THEN
    SUMPXT=APUX(J,I)+UAPX(J,I)-UPBX(J,I)
  ELSE
    SUMPXT=SUMPX
  END IF
  PX(I,J)=PX(I,J)+MUX*(SUMPX+SUMPXT)
  PX(J,I)=PX(I,J)
END IF
END DO
END DO
END DO

RETURN
END
2.11.  nfily.f

C Sway Filter Neurocomputing Solver for the Riccati Equation
C
SUBROUTINE NFILY

INCLUDE 'comint2.com'

INCLUDE 'nopt4.com'

REAL SPY(11,11),SUMEY,SUMPY,SUMPYT
REAL PAPY(11,11),UPBY(11,11),UAPY(11,11),APUY(11,11)
REAL EUY(11,11),UY(11),PZY(11)

C Compute Prescribed Nonlinearity Alpha
C
ALPY=Y*Q2Y(1)*Y+YD*Q2Y(2)*YD
IF(ALPY.GT.ALPHYMAX) ALPY=ALPHYMAX
DO I=1,11
   APY(I,I)=APYO(I)+ALPY
END DO

C Start Training Iterations & Initialize Variables
C
DO L=1,3
   DO I=1,11
      DO J=1,11
         SPY(I,J)=0.0
         PAPY(I,J)=0.0
         UPBY(I,J)=0.0
         UAPY(I,J)=0.0
         APUY(I,J)=0.0
      END DO
   END DO

C Compute Matrix Product SP
C
DO I=1,6
   DO J=1,11
      DO K=1,6
         SPY(I,J)=SPY(I,J)+BRBY(I,K)*PY(K,J)
      END DO
   END DO

C Compute Matrix Product PA
C
DO I=1,11
  DO J=1,6
    DO K=1,6
      PAPY(I,J)=PAPY(I,J)+PY(I,K)*APY(K,J)
    END DO
  END DO
PAPY(I,7)=PY(I,7)*APY(7,7)
PAPY(I,8)=PY(I,7)*APY(7,8)+PY(I,8)*APY(8,8)
PAPY(I,9)=PY(I,8)*APY(8,9)+PY(I,9)*APY(9,9)
  DO K=1,6
    PAPY(I,10)=PAPY(I,10)+PY(I,K)*APY(K,10)
  END DO
PAPY(I,10)=PAPY(I,10)+PY(I,10)*APY(10,10)
  DO K=1,6
    PAPY(I,11)=PAPY(I,11)+PY(I,K)*APY(K,11)
  END DO
PAPY(I,11)=PAPY(I,11)+PY(I,11)*APY(11,11)
END DO
C
C Compute Error Signal v, Vector p and Matrix Product vz
C Note: Matrix Product A'P is transpose of PA by symmetry of P
C
DO I=1,11
  UY(I)=0.
  PZY(I)=0.
  DO J=1,11
    IF(I.LE.J) THEN
      SUMEY=0.
      DO K=1,11
        SUMEY=SUMEY+PY(I,K)*SPY(K,J)
      END DO
      EUY(I,J)=SUMEY-PAPY(J,I)-PAPY(I,J)-R1PY(I,J)
      EUY(J,I)=EUY(I,J)
    END IF
    UY(I)=UY(I)+EUY(I,J)*ZY(L,J)
    PZY(I)=PZY(I)+PY(I,J)*ZY(L,J)
  END DO
END DO
C
C Compute Matrix Product Avz
C
DO I=1,6
  DO J=1,11
    DO K=1,6
      APUY(I,J)=APUY(I,J)+PY(I,K)*UY(K)*ZY(L,J)
    END DO
    APUY(I,J)=APUY(I,J)+PY(I,10)*UY(10)*ZY(L,J)
    x
    APUY(I,J)=APUY(I,J)+PY(I,11)*UY(11)*ZY(L,J)
  END DO
APUY(7,J)=PY(7,7)*UY(7)*ZY(L,J)
  x
  +PY(7,8)*UY(8)*ZY(L,J)
APUY(8,J)=PY(8,8)*UY(8)*ZY(L,J)
  x
  +PY(8,9)*UY(9)*ZY(L,J)
APUY(9,J) = APUY(9,9) * UY(9) * ZY(L,J)
APUY(10,J) = APUY(10,9) + APUY(10,10) * UY(10) * ZY(L,J)
APUY(11,J) = APUY(11,9) + APUY(11,10) * UY(11) * ZY(L,J)

END DO

C
C Compute Matrix Product vzA
C
DO I=1,11
  DO J=1,6
    DO K=1,6
      UAPY(I,J) = UAPY(I,J) + UY(I) * ZY(L,K) * APUY(K,J)
    END DO
  END DO
  UAPY(I,7) = UY(I) * ZY(L,7) * APUY(7,7)
  UAPY(I,8) = UY(I) * ZY(L,7) * APUY(7,8)
  UAPY(I,9) = UY(I) * ZY(L,8) * APUY(8,9)
  UAPY(I,10) = UAPY(I,10) + UY(I) * ZY(L,K) * APUY(K,10)
END DO

C
C Compute Matrix Product vp'S
C
DO I=1,11
  DO J=1,6
    DO K=1,6
      UPBY(I,J) = UPBY(I,J) + UY(I) * PZY(K) * BRBY(K,J)
    END DO
  END DO
  UPBY(I,9) = UPBY(I,9) + UY(I) * PZY(9) * BRBY(9,9)
END DO

C
C Sum Avz + vzA + vp'S and integrate to update Riccati Solution P
C
DO I=1,11
  DO J=1,11
    IF(I.LE.J) THEN
      SUMPY = APUY(I,J) + UAPY(I,J) - UPBY(I,J)
    IF(I.NE.J) THEN
      SUMPTY = APUY(J,I) + UAPY(J,I) - UPBY(J,I)
      ELSE
        SUMPTY = SUMPY
      END IF
      PY(I,J) = PY(I,J) + MUY * (SUMPY + SUMPTY)
    END IF
END DO
END DO
END DO
END DO
RETURN
END
2.12.  *nfilt.f*

Heave Filter Neurocomputing Solver for the Riccati Equation

SUBROUTINE NFILZ

INCLUDE 'comint2.com'

INCLUDE 'nopt4.com'

REAL SPZ(6,6),SUMEZ,SUMPZ,SUMPZT
REAL PAPZ(6,6),UPBZ(6,6),UAPZ(6,6),APUZ(6,6)
REAL EUZ(6,6),UZ(6),PZ(6)

C

C Compute Prescribed Nonlinearity Alpha
C
ALPZ=Z*Q2Z(1)*Z+ZD*Q2Z(2)*ZD
IF(ALPZ.GT.ALPHZMAX) ALPZ=ALPHZMAX
DO I=1,6
    APZ(I,I)=APZ0(I)+ALPZ
END DO

C

C Start Training Iterations & Initialize Variables
C
DO L=1,3
    DO I=1,6
        DO J=1,6
            SPZ(I,J)=0.
            PAPZ(I,J)=0.
            UPBZ(I,J)=0.
            UAPZ(I,J)=0.
            APUZ(I,J)=0.
        END DO
    END DO

C

C Compute Matrix Product SP
C
DO I=1,2
    DO J=1,6
        SPZ(I,J)=BRBZ(I,1)*PZ(1,J)
END DO
END DO
DO J=1,6
    SPZ(5,J)=BRBZ(5,1)*PZ(1,J)
END DO

C

C Compute Matrix Product PA
C
DO I=1,6
    PAPZ(I,1)=PZ(I,1)*APZ(1,1)+PZ(I,2)*APZ(2,1)
PAPZ(I,2) = PZ(I,1) * APZ(1,2) + PZ(I,2) * APZ(2,2)
PAPZ(I,3) = PZ(I,3) * APZ(3,3)
PAPZ(I,4) = PZ(I,3) * APZ(3,4) + PZ(I,4) * APZ(4,4)
PAPZ(I,5) = PZ(I,4) * APZ(4,5) + PZ(I,5) * APZ(5,5)
PAPZ(I,6) = PZ(I,1) * APZ(1,6) + PZ(I,2) * APZ(2,6)

END DO

C Compute Error Signal v, Vector p and Matrix Product vz

DO I=1,6
  UZ(I) = 0.
  PZZ(I) = 0.
  DO J=1,6
    IF (I.LE.J) THEN
      SUMEZ = 0.
      DO K=1,6
        SUMEZ = SUMEZ + PZ(I, K) * SPZ(K, J)
      END DO
      EUZ(I, J) = SUMEZ - PAPZ(J, I) - PAPZ(I, J) - R1PZ(I, J)
      EUZ(J, I) = EUZ(I, J)
    END IF
    UZ(I) = UZ(I) + EUZ(I, J) * ZZ(L, J)
    PZZ(I) = PZZ(I) + PZ(I, J) * ZZ(L, J)
  END DO
END DO

C Compute Matrix Product Avz

DO I=1,2
  DO J=1,6
    APUZ(I, J) = APZ(I, 1) * UZ(1) * ZZ(L, J) + APZ(I, 2) * UZ(2) * ZZ(L, J) + APZ(I, 6) * UZ(6) * ZZ(L, J)
  END DO
END DO

DO J=1,6
  APUZ(3, J) = APZ(3, 3) * UZ(3) * ZZ(L, J) + APZ(3, 4) * UZ(4) * ZZ(L, J)
  APUZ(4, J) = APZ(4, 4) * UZ(4) * ZZ(L, J) + APZ(4, 5) * UZ(5) * ZZ(L, J)
  APUZ(5, J) = APZ(5, 5) * UZ(5) * ZZ(L, J)
  APUZ(6, J) = APZ(6, 6) * UZ(6) * ZZ(L, J)
END DO

C Compute Matrix Product vzA

DO I=1,6
  UAPZ(I, 1) = UZ(I) * ZZ(L, 1) * APZ(1, 1) + UZ(I) * ZZ(L, 2) * APZ(2, 1)
  UAPZ(I, 2) = UZ(I) * ZZ(L, 1) * APZ(1, 2) + UZ(I) * ZZ(L, 2) * APZ(2, 2)
  UAPZ(I, 3) = UZ(I) * ZZ(L, 3) * APZ(3, 3)
  UAPZ(I, 4) = UZ(I) * ZZ(L, 3) * APZ(3, 4) + UZ(I) * ZZ(L, 4) * APZ(4, 4)
  UAPZ(I, 5) = UZ(I) * ZZ(L, 4) * APZ(4, 5) + UZ(I) * ZZ(L, 5) * APZ(5, 5)
  UAPZ(I, 6) = UZ(I) * ZZ(L, 1) * APZ(1, 6) + UZ(I) * ZZ(L, 2) * APZ(2, 6)
END DO
END DO

C Compute Matrix Product vp'S
C
DO I=1,6
  DO J=1,2
    UPBZ(I,J)=UZ(I)*PZZ(1)*BRBZ(1,J)
    x +UZ(I)*PZZ(2)*BRBZ(2,J)+UZ(I)*PZZ(5)*BRBZ(5,J)
  END DO
  UPBZ(I,5)=UZ(I)*PZZ(1)*BRBZ(1,5)
  x +UZ(I)*PZZ(2)*BRBZ(2,5)+UZ(I)*PZZ(5)*BRBZ(5,5)
END DO

C Sum Avz + vzA + vp'S and integrate to update Riccati Solution P
C
DO I=1,6
  DO J=1,6
    IF(I.LE.J) THEN
      SUMPZ=APUZ(I,J)+UAPZ(I,J)-UPBZ(I,J)
    IF(I.NE.J) THEN
      SUMPZT=APUZ(J,I)+UAPZ(J,I)-UPBZ(J,I)
    ELSE
      SUMPZT=SUMPZ
    END IF
    PZ(I,J)=PZ(I,J)+MUZ*(SUMPZ+SUMPZT)
    PZ(J,I)=PZ(I,J)
  END IF
END DO
END DO

RETURN
END
2.13. \textit{resetc2.f}

SUBROUTINE RESETC2

C
C     THIS ROUTINE:
C
C     (1) DOES T=0 INITIALIZATION OF MOTION VARIABLES
C
C     (2) USES A SECOND ORDER SCHEME TO DRIVE TO THE NEUTRAL
C        POSITION. (OVERDAMPED OSCILLATOR; ZETA = 1.5, OMEGAN = 1.0 )
C
INCLUDE 'comint2.com'
INCLUDE 'wcom2.com'
INCLUDE 'matrix1c.com'

C**********************************************************************
C
C      GAINS, ACCELERATION AND VELOCITY LIMITS FOR SECOND ORDER.
C
C      VALUES FOR ACCELERATION AND VELOCITY ARE METERS/SEC**2
C      AND METERS/SEC.  ROTATIONS AND ROTATIONAL VELOCITIES
C      ARE IN RADIANS AND RADIANS/SECOND.
C
C**********************************************************************
C
C      .03492 RADIANS/SEC**2 == 2.0  DEG/SEC**2
C      .294  METERS/SEC**2  ==  .03 G
C
PARAMETER (A_ACCLIM = .03491)
PARAMETER (T_ACCLIM = .294  )

C      THESE PARAMETERS ARE SET TO THE PERFORMANCE LIMIT OF THE BASE
C
C      .2617 RAD/SEC == 15 DEG/SEC
C      .610  METERS/SEC
C
PARAMETER (A_VELLIM = .2617 )
PARAMETER (T_VELLIM = .610  )

C      VALUES FOR X FILTER
C
DATA XDDLIM /  T_ACCLIM /
DATA XDLIM  /  T_VELLIM /

C      VALUES FOR Y FILTER
C
DATA YDDLIM /  T_ACCLIM /
DATA YDLIM  /  T_VELLIM /

C      VALUES FOR Z FILTER
C
DATA ZDDLIM /  T_ACCLIM /
DATA ZDLIM  /  T_VELLIM /
VALUES FOR PSI FILTER

DATA PSIDDLIM / A_ACCLIM /
DATA PSIDLIM / A_VELLIM /

VALUES FOR THETA FILTER

DATA THEDDLIM / A_ACCLIM /
DATA THEDLIM / A_VELLIM /

VALUES FOR PHI FILTER

DATA PHIDDLIM / A_ACCLIM /
DATA PHIDLIM / A_VELLIM /
DATA TWOZOMGN/ 3.0 /

local functions
CLIMIT(X,XMIN,XMAX) = MIN( MAX( X, XMIN ), XMAX )

If last operate run ended braked, reset braking algorithm to unbraked state

IF (FLAG2.EQ.1) THEN
  X = SSI(1)
  Y = SSI(2)
  Z = SSI(3)
  PHI = BETAS(1)
  THE = BETAS(2)
  PSI = BETAS(3)
  DO IJ=1,6
      RLID(IJ)=0.
      RLIDO(IJ)=0.
  END DO
  IT2=0
  FLAG2=0
END IF

T = 0 INITIALIZATION;

T   = 0.
INT  = 0
DXDLX = 0.
DXDLXD = 0.
DXDDX = 0.
DXDDXD = 0.
DTHDDX = 0.
LAMX  = LAMX0
DELEX = DELX0
DYDLY = 0.
DYDLXD = 0.
DYDDY = 0.
DYDDYD = 0.
DPHDDY = 0.
LAMY  = LAMY0
DELY  = DELY0
DZDEZ  = 0.
DZDEZD = 0.
ETAZ  = ETA20
DPSDE = 0.
ETAPS = ETAPS0

FADE TO THE NEUTRAL POSITION USING SECOND ORDER FILTER TO PROVIDE
VELOCITY AND ACCELERATION CONTROL.

DRIVE X TO NEUTRAL POSITION (X = 0)

XDDF  = -X - TWOZOMGN*XD
XDD   = CLIMIT(XDDF,-XDDLIM,XDDLIM)
XDF   = XD + XDD*H
XD    = CLIMIT(XDF,-XDLIM,XDLIM)
X     = X + XD*H

DRIVE Y TO NEUTRAL POSITION (Y = 0)

YDDF  = -Y - TWOZOMGN*YD
YDD   = CLIMIT(YDDF,-YDDLIM,YDDLIM)
YDF   = YD + YDD*H
YD    = CLIMIT(YDF,-YDLIM,YDLIM)
Y     = Y + YD*H

DRIVE Z TO NEUTRAL POSITION (Z = 0)

ZDDF  = -Z - TWOZOMGN*ZD
ZDD   = CLIMIT(ZDDF,-ZDDLIM,ZDDLIM)
ZDF   = ZD + ZDD*H
ZD    = CLIMIT(ZDF,-ZDLIM,ZDLIM)
Z     = Z + ZD*H

DRIVE PSI TO NEUTRAL POSITION (PSI = 0)

PSIDDF = -PSI - TWOZOMGN*PSID
PSIDDFL = CLIMIT(PSIDDF,-PSIDDLIM,PSIDDLIM)
PSIDF  = PSID + PSIDDFL*H
PSID   = CLIMIT(PSIDF,-PSIDLIM,PSIDLIM)
PSI    = PSI + PSID*H

DRIVE THETA TO NEUTRAL POSITION (THETA = 0)

THEDDF = -THE - TWOZOMGN*THED
THEDDFL = CLIMIT(THEDDF,-THEDDLIM,THEDDLIM)
THEDF  = THED + THEDDFL*H
THED   = CLIMIT(THEDF,-THEDLIM,THEDLIM)
THE    = THE + THED*H

DRIVE PHI TO NEUTRAL POSITION (PHI = 0)

PHIDDF = -PHI - TWOZOMGN*PHID
PHIDDFL = CLIMIT(PHIDDF,-PHIDDLIM,PHIDDLIM)
PHIDF  = PHID + PHIDDFL*H
PHID   = CLIMIT(PHIDF,-PHIDLIM,PHIDLIM)
PHI = PHI + PHID*H

C
DUMMY INTEGRATIONS
C
XD1 = XD
DXDLXD1 = DXDLXD
DXDDXD1 = DXDDXD
YD1 = YD
DYDLYD1 = DYDLYD
DYDDYD1 = DYDDYD
ZD1 = ZD
DZDEZD1 = DZDEZD
RETURN
END
2.14. \textit{simq.f}

SUBROUTINE SIMQ(A,B,N,KS)
DIMENSION A(36),B(6)
TOL=0.
KS=0
JJ=-N
DO 65 J=1,N
     JY=J+1
     JJ=JJ+N+1
     BIGA=0.
     IT=JJ-J
     DO 30 I=J,N
          IJ=IT+I
          IF(ABS(BIGA)-ABS(A(IJ))) 20,30,30
          20     BIGA=A(IJ)
          IMAX=I
     30     CONTINUE
     IF(ABS(BIGA)-TOL) 35,35,40
     35     KS=1
     RETURN
     40     I1=J+N*(J-2)
     IT=IMAX-J
     DO 50 K=J,N
          I1=I1+N
          I2=I1+IT
          SAVE1=A(I1)
          A(I1)=A(I2)
          A(I2)=SAVE1
          50     A(I1)=A(I1)/BIGA
          SAVE1=B(IMAX)
          B(IMAX)=B(J)
          B(J)=SAVE1/BIGA
     IF(J-N) 55,70,55
     55     IQS=N*(J-1)
     DO 65 IX=JY,N
          IXJ=IQS+IX
          IT=J-IX
          DO 60 JX=JY,N
               IXJX=IXJX+IT
               60       A(IXJX)=A(IXJX)-(A(IXJ)*A(JJX))
          65     B(IX)=B(IX)-(B(J)*A(IXJ))
     NY=N-1
     IT=N*N
     DO 80 J=1,NY
          IA=IT-J
          IB=N-J
          IC=N
     DO 80 K=1,J
          B(IB)=B(IB)-A(IA)*B(IC)
          IA=IA-N
     80     IC=IC-1
     RETURN
END
2.15. \textit{state4.f}

NONLINEAR STATE SPACE FILTERS (ALL IN INERTIAL FRAME):

SUBROUTINE STATE4

INCLUDE 'optint3.com'

INCLUDE 'nopt4.com'

REAL ABK, ABK1Y(6,6), BK2X(6,3), BK3X(6), XXI1(9), XXI2(9)
REAL ABK1Y(6,6), BK2Y(6,3), BK3Y(6), XYI1(9), XYI2(9)
REAL XZI1(5), XZI2(5), XRI1(4), XRI2(4)
REAL XPI1(4), XPI2(4), XQI1(4), XQI2(4)

ROLL AND PITCH UNITY GAIN FILTERS

BSDR(1)=BADNO(1)
BSDR(2)=BADNO(2)

9TH ORDER SURGE/PITCH STATE SPACE FILTER

DO I=1,6
  DO J=1,6
    K1X(1,J)=R2IX(1)*
      (BVX(1,1)*PX(1,J)+BVX(2,1)*PX(2,J)+BVX(3,1)*PX(3,J)
       +BVX(4,1)*PX(4,J)+BVX(5,1)*PX(5,J)+BVX(6,1)*PX(6,J)
       +DQCX(J))
    K1X(2,J)=R2IX(2)*
      (BVX(1,2)*PX(1,J)+BVX(2,2)*PX(2,J)+BVX(3,2)*PX(3,J)
       +BVX(4,2)*PX(4,J)+BVX(5,2)*PX(5,J)+BVX(6,2)*PX(6,J)
       +PX(9,J))
    ABK1X(I,J)=
      AVX(I,J)-BVX(I,1)*K1X(1,J)-BVX(I,2)*K1X(2,J)
  END DO
END DO

DO I=1,6
  DO J=1,3
    K2X(1,J)=R2IX(1)*
      (BVX(1,1)*PX(1,J+6)+BVX(2,1)*PX(2,J+6)+BVX(3,1)*PX(3,J+6)
       +BVX(4,1)*PX(4,J+6)+BVX(5,1)*PX(5,J+6)+BVX(6,1)*PX(6,J+6))
    K2X(2,J)=R2IX(2)*
      (BVX(1,2)*PX(1,J+6)+BVX(2,2)*PX(2,J+6)+BVX(3,2)*PX(3,J+6)
       +BVX(4,2)*PX(4,J+6)+BVX(5,2)*PX(5,J+6)+BVX(6,2)*PX(6,J+6)
       +PX(9,J+6))
    BK2X(I,J)=BVX(I,1)*K2X(1,J)+BVX(I,2)*K2X(2,J)
  END DO
END DO

K3X(1)=R2IX(1)*
x  (BVX(1,1)*PX(1,11)+BVX(2,1)*PX(2,11)+BVX(3,1)*PX(3,11)
x +BVX(4,1)*PX(4,11)+BVX(5,1)*PX(5,11)+BVX(6,1)*PX(6,11))

K3X(2)=R2IX(2)*
x  (BVX(1,2)*PX(1,11)+BVX(2,2)*PX(2,11)+BVX(3,2)*PX(3,11)
x +BVX(4,2)*PX(4,11)+BVX(5,2)*PX(5,11)+BVX(6,2)*PX(6,11)
x +PX(9,11))

DO I=1,6
  BK3X(I)=BVX(I,1)*K3X(1)+BVX(I,2)*(1.0+K3X(2))
END DO

DO I=1,6
  XXI1(I)=DT*
x  (ABK1X(I,1)*XXO(1)+ABK1X(I,2)*XXO(2)+ABK1X(I,3)*XXO(3)
x +ABK1X(I,4)*XXO(4)+ABK1X(I,5)*XXO(5)+ABK1X(I,6)*XXO(6)
x -BK2X(I,1)*XXO(7)-BK2X(I,2)*XXO(8)-BK2X(I,3)*XXO(9)
x -BK3X(I)*A2NO(1))
END DO

XXI1(7)=DT*XXO(8)
XXI1(8)=DT*XXO(9)
XXI1(9)=DT*
x  (-K1X(2,1)*XXO(1)-K1X(2,2)*XXO(2)-K1X(2,3)*XXO(3)
x -K1X(2,4)*XXO(4)-K1X(2,5)*XXO(5)-K1X(2,6)*XXO(6)
x -K2X(2,1)*XXO(7)-K2X(2,2)*XXO(8)-K2X(2,3)*XXO(9)
x -K3X(2)*A2NO(1))

DO I=1,6
  XXI2(I)=DT*
x  (ABK1X(I,1)*XXO(1)+XXI1(1))+ABK1X(I,2)*XXO(2)+ABK1X(I,3)*XXO(3)
x +ABK1X(I,4)*XXO(4)+ABK1X(I,5)*XXO(5)+ABK1X(I,6)*XXO(6)
x -BK2X(I,1)*(XXO(7)+XXI1(7))-BK2X(I,2)*(XXO(8)+XXI1(8))
x -BK3X(I,3)*(XXO(9)+XXI1(9))-BK3X(I)*A2N(1))
END DO

XXI2(7)=DT*(XXO(8)+XXI1(8))
XXI2(8)=DT*(XXO(9)+XXI1(9))
XXI2(9)=DT*(-K1X(2,1)*XXO(1)+XXI1(1))
x  (-K1X(2,2)*(XXO(2)+XXI1(2))-K1X(2,3)*(XXO(3)+XXI1(3))
x -K1X(2,4)*(XXO(4)+XXI1(4))-K1X(2,5)*(XXO(5)+XXI1(5))
x -K1X(2,6)*(XXO(6)+XXI1(6))-K2X(2,1)*(XXO(7)+XXI1(7))
x -K2X(2,2)*(XXO(8)+XXI1(8))-K2X(2,3)*(XXO(9)+XXI1(9))
x -K3X(2)*A2N(1))

DO I=1,9
  XX(I)=XXO(I)+0.5*(XXI1(I)+XXI2(I))
END DO

C 9TH ORDER SWAY/ROLL STATE SPACE FILTER

DO I=1,6
  K1Y(I,1,J)=R2IY(1)*
x  (BVY(1,1)*PY(1,1)+BVY(2,1)*PY(2,1)+BVY(3,1)*PY(3,1)
x +BVY(4,1)*PY(4,1)+BVY(5,1)*PY(5,1)+BVY(6,1)*PY(6,1)
x +DQCY(J))

85
\[ K_{1Y}(2, J) = R_{2IY}(2) \]
\[ \times (B_{VY}(1, 2) \times P_{Y}(1, J) + B_{VY}(2, 2) \times P_{Y}(2, J) + B_{VY}(3, 2) \times P_{Y}(3, J) \]
\[ + B_{VY}(4, 2) \times P_{Y}(4, J) + B_{VY}(5, 2) \times P_{Y}(5, J) + B_{VY}(6, 2) \times P_{Y}(6, J) \]
\[ + P_{Y}(9, J)) \]

\[ A_{BK1Y}(I, J) = \]
\[ A_{VY}(I, J) - B_{VY}(I, 1) \times K_{1Y}(1, J) - B_{VY}(I, 2) \times K_{1Y}(2, J) \]
\[ \text{END DO} \]

\[ \text{END DO} \]

\[ \text{DO } I=1, 6 \]
\[ \text{DO } J=1, 3 \]
\[ K_{2Y}(1, J) = R_{2IY}(1) \]
\[ \times (B_{VY}(1, 1) \times P_{Y}(1, J+6) + B_{VY}(2, 1) \times P_{Y}(2, J+6) + B_{VY}(3, 1) \times P_{Y}(3, J+6) \]
\[ + B_{VY}(4, 1) \times P_{Y}(4, J+6) + B_{VY}(5, 1) \times P_{Y}(5, J+6) + B_{VY}(6, 1) \times P_{Y}(6, J+6) \]
\[ K_{2Y}(2, J) = R_{2IY}(2) \]
\[ \times (B_{VY}(1, 2) \times P_{Y}(1, J+6) + B_{VY}(2, 2) \times P_{Y}(2, J+6) + B_{VY}(3, 2) \times P_{Y}(3, J+6) \]
\[ + B_{VY}(4, 2) \times P_{Y}(4, J+6) + B_{VY}(5, 2) \times P_{Y}(5, J+6) + B_{VY}(6, 2) \times P_{Y}(6, J+6) \]
\[ + P_{Y}(9, J+6) \]

\[ B_{K2Y}(I, J) = B_{VY}(I, 1) \times K_{2Y}(1, J) + B_{VY}(I, 2) \times K_{2Y}(2, J) \]
\[ \text{END DO} \]

\[ \text{END DO} \]

\[ K_{3Y}(1) = R_{2IY}(1) \]
\[ \times (B_{VY}(1, 1) \times P_{Y}(1, 11) + B_{VY}(2, 1) \times P_{Y}(2, 11) + B_{VY}(3, 1) \times P_{Y}(3, 11) \]
\[ + B_{VY}(4, 1) \times P_{Y}(4, 11) + B_{VY}(5, 1) \times P_{Y}(5, 11) + B_{VY}(6, 1) \times P_{Y}(6, 11) \]

\[ K_{3Y}(2) = R_{2IY}(2) \]
\[ \times (B_{VY}(1, 2) \times P_{Y}(1, 11) + B_{VY}(2, 2) \times P_{Y}(2, 11) + B_{VY}(3, 2) \times P_{Y}(3, 11) \]
\[ + B_{VY}(4, 2) \times P_{Y}(4, 11) + B_{VY}(5, 2) \times P_{Y}(5, 11) + B_{VY}(6, 2) \times P_{Y}(6, 11) \]
\[ + P_{Y}(9, 11) \]

\[ \text{DO } I=1, 6 \]
\[ B_{K3Y}(I) = B_{VY}(I, 1) \times K_{3Y}(1) + B_{VY}(I, 2) \times (1.0 + K_{3Y}(2)) \]
\[ \text{END DO} \]

\[ \text{DO } I=1, 6 \]
\[ X_{YI1}(I) = D_{T} \]
\[ \times (A_{BK1Y}(I, 1) \times X_{YO}(1) + A_{BK1Y}(I, 2) \times X_{YO}(2) + A_{BK1Y}(I, 3) \times X_{YO}(3) \]
\[ + A_{BK1Y}(I, 4) \times X_{YO}(4) + A_{BK1Y}(I, 5) \times X_{YO}(5) + A_{BK1Y}(I, 6) \times X_{YO}(6) \]
\[ - B_{K2Y}(I, 1) \times X_{YO}(7) - B_{K2Y}(I, 2) \times X_{YO}(8) - B_{K2Y}(I, 3) \times X_{YO}(9) \]
\[ - B_{K3Y}(I) \times A_{2NO}(2) \]
\[ \text{END DO} \]

\[ X_{YI1}(7) = D_{T} \times X_{YO}(8) \]
\[ X_{YI1}(8) = D_{T} \times X_{YO}(9) \]
\[ X_{YI1}(9) = D_{T} \]
\[ \times (-K_{1Y}(2, 1) \times X_{YO}(1) - K_{1Y}(2, 2) \times X_{YO}(2) - K_{1Y}(2, 3) \times X_{YO}(3) \]
\[ - K_{1Y}(2, 4) \times X_{YO}(4) - K_{1Y}(2, 5) \times X_{YO}(5) - K_{1Y}(2, 6) \times X_{YO}(6) \]
\[ - K_{2Y}(2, 1) \times X_{YO}(7) - K_{2Y}(2, 2) \times X_{YO}(8) - K_{2Y}(2, 3) \times X_{YO}(9) \]
\[ - K_{3Y}(2) \times A_{2NO}(2) \]

\[ \text{DO } I=1, 6 \]
\[ X_{YI2}(I) = D_{T} \]
\[ \times (A_{BK1Y}(I, 1) \times (X_{YO}(1) + X_{YI1}(1)) + A_{BK1Y}(I, 2) \times (X_{YO}(2) + X_{YI1}(2)) \]
\[ + A_{BK1Y}(I, 3) \times (X_{YO}(3) + X_{YI1}(3)) + A_{BK1Y}(I, 4) \times (X_{YO}(4) + X_{YI1}(4)) \]
END DO
XYI2(7)=DT*(XYO(8)+XYI1(8))
XYI2(8)=DT*(XYO(9)+XYI1(9))
XYI2(9)=DT*(-K1Y(2,1)*(XYO(1)+XYI1(1)))
- K1Y(2,2)*(XYO(2)+XYI1(2))-K1Y(2,3)*(XYO(3)+XYI1(3))
- K1Y(2,4)*(XYO(4)+XYI1(4))-K1Y(2,5)*(XYO(5)+XYI1(5))
- K1Y(2,6)*(XYO(6)+XYI1(6))-K2Y(2,1)*(XYO(7)+XYI1(7))
- K2Y(2,2)*(XYO(8)+XYI1(8))-K2Y(2,3)*(XYO(9)+XYI1(9))
- K3Y(2)*A2N(2))
DO I=1,9
  XY(I)=XYO(I)+0.5*(XYI1(I)+XYI2(I))
END DO

5TH ORDER HEAVE STATE SPACE FILTER

K1Z(1)=BVZ(1)*PZ(1,1)+BVZ(2)*PZ(2,1)+PZ(5,1)
K1Z(2)=BVZ(1)*PZ(1,2)+BVZ(2)*PZ(2,2)+PZ(5,2)
K2Z(1)=BVZ(1)*PZ(1,3)+BVZ(2)*PZ(2,3)+PZ(5,3)
K2Z(2)=BVZ(1)*PZ(1,4)+BVZ(2)*PZ(2,4)+PZ(5,4)
K2Z(3)=BVZ(1)*PZ(1,5)+BVZ(2)*PZ(2,5)+PZ(5,5)
K3Z=BVZ(1)*PZ(1,6)+BVZ(2)*PZ(2,6)+PZ(5,6)
DO I=1,2
XZI1(I)=DT*
  ((AVZ(I,1)-BVZ(I)*K1Z(1))*XZO(1)
+ (AVZ(I,2)-BVZ(I)*K1Z(2))*XZO(2)
- BVZ(I)*K2Z(1)*XZO(3)+K2Z(2)*XZO(4)+K2Z(3)*XZO(5))
- BVZ(I)*(1+K3Z)*A2NO(3))
END DO
XZI1(3)=DT*XZO(4)
XZI1(4)=DT*XZO(5)
XZI1(5)=DT*(-K1Z(1)*XZO(1)-K1Z(2)*XZO(2)-K2Z(1)*XZO(3)
- K2Z(2)*XZO(4)-K2Z(3)*XZO(5)-K3Z*A2NO(3))
DO I=1,2
XZI2(I)=DT*
  ((AVZ(I,1)-BVZ(I)*K1Z(1))*XZO(1)+XZI1(1))
+ (AVZ(I,2)-BVZ(I)*K1Z(2))*XZO(2)+XZI1(2))
- BVZ(I)*K2Z(1)*(XZO(3)+XZI1(3))+K2Z(2)*(XZO(4)+XZI1(4))
+ K2Z(3)*(XZO(5)+XZI1(5)))
- BVZ(I)*(1+K3Z)*A2N(3))
END DO
XZI2(3)=DT*(XZO(4)+XZI1(4))
XZI2(4)=DT*(XZO(5)+XZI1(5))
XZI2(5)=DT*(-K1Z(1)*(XZO(1)+XZI1(1))
- K1Z(2)*(XZO(2)+XZI1(2))
- K2Z(1)*(XZO(3)+XZI1(3))
- K2Z(2)*(XZO(4)+XZI1(4))
- K2Z(3)*(XZO(5)+XZI1(5))-K3Z*A2N(3))
DO I=1,5
  XZ(I)=XZO(I)+0.5*(XZI1(I)+XZI2(I))
END DO
4TH ORDER YAW STATE SPACE FILTER

DO I=1,3
  K1R(I)=R2IR*
  x (BVR(1)*PR(1,I)+BVR(2)*PR(2,I)+BVR(3)*PR(3,I)+PR(4,I)+DQCR(I))
END DO

K2R=R2IR*
  x (BVR(1)*PR(1,4)+BVR(2)*PR(2,4)+BVR(3)*PR(3,4)+PR(4,4))
K3R=R2IR*
  x (BVR(1)*PR(1,5)+BVR(2)*PR(2,5)+BVR(3)*PR(3,5)+PR(4,5)-DQDR)

DO I=1,3
  XRI1(I)=DT*
  x ((AVR(I,1)-BVR(I)*K1R(1))*XRO(1)
  + (AVR(I,2)-BVR(I)*K1R(2))*XRO(2)
  + (AVR(I,3)-BVR(I)*K1R(3))*XRO(3)
  -BVR(I)*K2R*XRO(4)
  -BVR(I)*(1+K3R)*BADNO(3))
END DO
XRI1(4)=DT*(-K1R(1)*XRO(1)-K1R(2)*XRO(2)
  -K1R(3)*XRO(3)-K2R*XRO(4)-K3R*BADNO(3))

DO I=1,3
  XRI2(I)=DT*
  x ((AVR(I,1)-BVR(I)*K1R(1))*XRO(1)+XRI1(I))
  + (AVR(I,2)-BVR(I)*K1R(2))*XRO(2)+XRI1(2))
  + (AVR(I,3)-BVR(I)*K1R(3))*XRO(3)+XRI1(3))
  -BVR(I)*K2R*(XRO(4)+XRI1(4))
  -BVR(I)*(1+K3R)*BADNO(3))
END DO
XRI2(4)=DT*(-K1R(1)*(XRO(1)+XRI1(1))
  -K1R(2)*XRO(2)+XRI1(2))
  -K1R(3)*XRO(3)+XRI1(3))
  -K2R*(XRO(4)+XRI1(4))-K3R*BADNO(3))

DO I=1,4
  XR(I)=XRO(I)+0.5*(XRI1(I)+XRI2(I))
END DO

4TH ORDER PITCH STATE SPACE FILTER

DO I=1,3
  K1P(I)=R2IP*
  x (BVP(1)*PP(1,I)+BVP(2)*PP(2,I)+BVP(3)*PP(3,I)+PP(4,I)+DQCP(I))
END DO

K2P=R2IP*
  x (BVP(1)*PP(1,4)+BVP(2)*PP(2,4)+BVP(3)*PP(3,4)+PP(4,4))
K3P=R2IP*
  x (BVP(1)*PP(1,5)+BVP(2)*PP(2,5)+BVP(3)*PP(3,5)+PP(4,5)-DQDP)

DO I=1,3
  XPI1(I)=DT*
  x ((AVR(I,1)-BVP(I)*K1P(1))*XPO(1)
END DO
XPI1(4)=DT*(-K1P(1)*XPO(1)-K1P(2)*XPO(2)
X-K1P(3)*XPO(3)-K2P*XPO(4)-K3P*BADNO(2))

DO I=1,3
    XP1(I)=XPO(I)+0.5*(XPI1(I)+XP12(I))
END DO

4TH ORDER ROLL STATE SPACE FILTER

DO I=1,3
    K1Q(I)=R2IQ*
    (BVQ(1)*PQ(1,I)+BVQ(2)*PQ(2,I)+BVQ(3)*PQ(3,I)+PQ(4,I)+DQCQ(I))
END DO
K2Q=R2IQ*
    (BVQ(1)*PQ(1,4)+BVQ(2)*PQ(2,4)+BVQ(3)*PQ(3,4)+PQ(4,4))
K3Q=R2IQ*
    (BVQ(1)*PQ(1,5)+BVQ(2)*PQ(2,5)+BVQ(3)*PQ(3,5)+PQ(4,5)-DQDQ)

DO I=1,3
    XQ11(I)=DT*
    ((AVQ(I,1)-BVQ(I)*K1Q(1))*XQO(I)
    + (AVQ(I,2)-BVQ(I)*K1Q(2))*XQO(2)
    + (AVQ(I,3)-BVQ(I)*K1Q(3))*XQO(3)
    -BVQ(I)*K2Q*XQO(4)
    -BVQ(I)*(1+K3Q)*BADNO(1))
END DO
XQ11(4)=DT*(-K1Q(1)*XQO(1)-K1Q(2)*XQO(2)
X-K1Q(3)*XQO(3)-K2Q*XQO(4)-K3Q*BADNO(1))

DO I=1,3
    XQ12(I)=DT*
    ((AVQ(I,1)-BVQ(I)*K1Q(1))*XQO(I)+XQ11(I))
    + (AVQ(I,2)-BVQ(I)*K1Q(2))*XQO(2)+XQ11(2))
    + (AVQ(I,3)-BVQ(I)*K1Q(3))*XQO(3)+XQ11(3))
    -BVQ(I)*K2Q*(XQO(4)+XQ11(4))
    -BVQ(I)*(1+K3Q)*BADN(1))
END DO
XQ12(4)=DT*(-K1Q(1)*(XQO(1)+XQ11(1)))
DO I=1,4
   XQ(I)=XQO(I)+0.5*(XQI1(I)+XQI2(I))
END DO

C************** UPDATE ALL THE DUMMY VARIABLES: ******************
C
DO I=1,9
   XXO(I)=XX(I)
   XYO(I)=XY(I)
END DO

DO I=1,5
   XZO(I)=XZ(I)
END DO

DO I=1,4
   XRO(I)=XR(I)
   XPO(I)=XP(I)
   XQO(I)=XQ(I)
END DO

A2NO(1) = A2N(1)
A2NO(2) = A2N(2)
A2NO(3) = A2N(3)
BADNO(1) = BADN(1)
BADNO(2) = BADN(2)
BADNO(3) = BADN(3)

RETURN
END
2.16. *vmult.f*

```fortran
C**********************************************************************
C       Subroutine VMULT : MATRIX MULTIPLICATION.
C**********************************************************************
C
SUBROUTINE VMULT(A,B,C,K,L,M)
DIMENSION A(K,L),B(K,L),C(K,M)
DO 20 KK = 1,K
   DO 20 MM = 1,M
      C(KK,MM) = 0.0
   DO 20 LL = 1,L
      C(KK,MM) = C(KK,MM) + A(KK,LL)*B(LL,MM)
20  CONTINUE
RETURN
END
```

2.17. *winit4.f*

```fortran
SUBROUTINE WINIT4
C
C     THIS ROUTINE LOADS THE INITIAL VALUES INTO THE WASHOUT
C     PARAMETER ARRAYS.
C
INCLUDE 'nopt4.com'

REAL BRBXVEC(66),R1PXVEC(66),ZXVEC(66)
REAL BRBYVEC(66),R1PYVEC(66),ZYVEC(66)
REAL BRBRVEC(15),R1PRVEC(15),ZRVEC(15)
REAL BRBPVEC(15),R1PPVEC(15),ZPVEC(15)
REAL BRBQVEC(15),R1PQVEC(15),ZQVEC(15)
REAL BRBZVEC(21),R1PZVEC(21),ZZVEC(21)

C
C Initialization of Nonlinear Algorithm Inputs
C
DATA XXO/9*0./XYO/9*0./,XRO/4*0./,XPO/4*0./,XQO/4*0./,XZO/5*0./
C
C Parameters for nonlinear roll/sway channel filters.
C
DATA ALPY /0.0/, ALPYMAX /1.0/, Q2Y /0.0,0.8/, MUY /4.0E-6 /
```

DATA APY /
+ -0.48601433, 1.50095561, 0.43295640, -2.00166583, 2.30852675,
+ 0.93783312, 0.0000, 0.0000, 0.0000, 0.0000,
+ -0.22785440, 1.23627334, 0.43548518, -2.00721978, 2.32816934,
+ 0.92719057, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.00000000, 0.00000000, 0.00000000, -0.50000000, 0.50000000,
+ -0.00000000, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.17852148, -0.42684297, 0.00091112, -0.01882892, 0.19024153,
+ 0.74839400, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.0000,
+ 0.84518815, 0.23802370, 0.00091112, 0.18117108, -0.00975847,
+ 0.08172733, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.47047268, -2.17119788, -0.42654860, 1.84882203, -2.39757925,
DO J=1,11
  APYO(J)=APY(J,J)
DO I=1,11
  IF(I.GE.J) THEN
+  0.2873278, -0.3474845,  0.0, -0.0806039, -0.0806039, -
  0.6884752/
DATA BVY /
  +  1.684826, -16.777107, -3.924000,  17.749601, -21.314831, -
  8.698809,
  +  1.333333,  1.333333, -0.000000,  0.200000, -0.200000, -
1.333333/
DATA DQCY /
  +  2641.408443,  2656.836237, -0.000000,
  +      5.558626,      5.558626, -2602.315320/
DATA ZYVEC /
  +       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
  -       1.0,
DATA PYVEC /
  +  8.78919914,
  +  8.64675152,  2.04454782, -5.18310331,  3.53876667, -7.88399752,
  +  8.38739821, -21.0214997, -25.10394095,  0.01775090,
  +  9.07316301,  2.21477279, -4.18934594,  4.64570745, -7.69883299,
  +  8.39486263, -21.05959150, -25.21567898,  0.01774509,
  +  0.71635788, -1.20003026,  0.98991751, -1.32889642,
  + -1.74100084, -8.39486263, -21.05959150, -25.21567898,  0.01774509,
  +  5.44431336,  0.1951117,  3.77151434,
  +  4.13378917,  10.88195613,  13.90068610, -0.02161095,  3.56872382,
  +  4.49770413, -3.84403097,
  + -4.01767838, -10.33185268, -12.56415459,  0.02169120, -3.19600160,
  +  8.54553179,
  +  8.27250466,  20.4679345,  23.69772741, -0.01783242,  6.00808010,
  +  23.81195147,  33.64457162,  29.33323432, -0.00748064,  6.63816656,
  + 71.85128707,  71.56090170, -0.03103290, 16.77905545,
  + 82.30634518, -0.06573714, 19.98961531,
  +  0.49999450, -0.01797396,
  +  5.06794969/

DO J=1,11
  APYO(J)=APY(J,J)
DO I=1,11
  IF(I.GE.J) THEN

93
BRBY(I,J)=BRBYVEC((J-1)*11-J*(J-1)/2+I)
BRBY(J,I)=BRBY(I,J)
R1PY(I,J)=R1PYVEC((J-1)*11-J*(J-1)/2+I)
R1PY(J,I)=R1PY(I,J)
PY(I,J)=PYVEC((J-1)*11-J*(J-1)/2+I)
PY(J,I)=PY(I,J)
ZY(I,J)=ZYVEC((J-1)*11-J*(J-1)/2+I)
ZY(J,I)=ZY(I,J)
END IF
END DO
END DO

c Parameters for nonlinear pitch/surge channel filters.
c
DATA ALPX /0.0/, ALPXMAX /1.0/, Q2X /0.0,0.6/, MUX /4.0E-6/
DATA APX /
+ 0.35214308, -1.72273247, 0.43037039, -2.38099014, 1.90345806,
+ -1.44348410, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.60755620, -1.99016155, 0.42784160, -2.36134755, 1.89792591,
+ -1.45137984, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.00000000, 0.00000000, 0.00000000, -0.50000000, 0.50000000,
+ -0.00000000, 0.00000000, 0.00000000, 0.0000, 0.0000,
+ 0.17753181, -0.42963264, -0.00091112, -0.00975847, 0.18117108,
+ 0.75422005, 0.0000, 0.0000, 0.0000, 0.0000,
+ 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,
+ -1.33333333, -1.33333333, 0.00000000, -0.20000000, 0.20000000,
+ 1.33333333, 0.0000, 0.0000, 0.0000, -3.14159265/
DATA BRBXVEC /
+ 1.77925517, 1.77466884, 0.00097480, 0.26137162, -0.26225730,
+ 1.78099759, 0.0000, 0.0000, 1.33333333, 0.0000, 0.0000,
+ 1.78432002, -0.00205131, 0.27780923, -0.27594547,
+ 1.77100221, 0.0000, 0.0000, 1.33333333, 0.0000, 0.0000,
+ 0.00064319, -0.00349374, 0.00290936,
+ 0.00024842, 0.00052276, -0.00016391, 0.00089035, -0.00074143,
+ 0.00054140, 0.0000, 0.0000, -1.000000, 0.0000,
+ -1.33333333, -1.33333333, 0.00000000, -0.20000000, 0.20000000,
+ 1.33333333, 0.0000, 0.0000, 0.0000, -3.14159265/
DATA R1PXVEC /
95
\[
\begin{align*}
+ & \quad 1.0, \quad -1.0, \quad -1.0, \quad -1.0, \\
+ & \quad 1.0, \quad -1.0, \quad -1.0, \\
+ & \quad 1.0, \quad -1.0, \\
+ & \quad 1.0 /
\end{align*}
\]

```
DO J=1,11
   APXO(J)=APX(J,J)
DO I=1,11
   IF(I.GE.J) THEN
      BRBX(I,J)=BRBXVEC((J-1)*11-J*(J-1)/2+I)
      BRBX(J,I)=BRBX(I,J)
      R1PX(I,J)=R1PXVEC((J-1)*11-J*(J-1)/2+I)
      R1PX(J,I)=R1PX(I,J)
      PX(I,J)=PXVEC((J-1)*11-J*(J-1)/2+I)
      PX(J,I)=PX(I,J)
      ZX(I,J)=ZXVEC((J-1)*11-J*(J-1)/2+I)
      ZX(J,I)=ZX(I,J)
   END IF
END DO
END DO
```

```
c Parameters for nonlinear yaw channel filters.
c
DATA ALPR /0.0/, ALPRMAX /1.0/, Q2R /120.0/, MUR /2.0E-6 /
```

```
DATA APR /
+1.45728460E-004,-1.69985373E-006,-4.89593935E-004,
+2.79035083E-002, 0.0,
+1.0, 0.0, 0.0, 0.0, 0.0,
+1.86874341E-001,2.17980102E-003,-4.89593935E-004,
+2.79035083E-002,0.00000000E+000,
+0.0, 0.0, 0.0, 0.0, 0.0,
+5.21851607E-003,6.08715277E-005,1.75322913E-002,
+9.99220787E-001, -1.0/
```

```
DATA BRBRVEC /
+3.49492524E-002,4.07666539E-004,1.17416612E-001,-5.21851607E-003,
+0.0,
+4.75523782E-006,1.36960937E-03,-6.08715277E-005, 0.0,
```

96
+3.94476555E-001, -1.75322913E-002, 0.0, 
+7.79212948E-004, 0.0, 
+ 0.0/
DATA R1PRVEC /
+7.79212948E-004, 0.0, 7.79212948E-004, 0.0, -2.79035083E-002,
+0.00000000E+000, 0.0, 0.00000000E+000, 0.0, 
+7.79212948E-004, 0.0, -2.79035083E-002,
+2.00000000E+002, 0.0, 
+9.92207878E-001/
DATA R2IR / 7.79212948E-004/
DATA AVR /
+ -1.87020070E-001, -2.18150087E-003, -6.28318531E-001, 
+1.00000000E+000, 0.00000000E+000, 0.00000000E+000, 
+0.00000000E+000, 0.00000000E+000, -6.28318531E-001/
DATA BVR / -6.69716293E+000, -7.81192456E-002, -2.25000000E+001/
DATA DQCR / 35.80986220, 0.00000000, 35.80986220/
DATA DQDR / 1.28234623E+003/
DATA ZRVEC /
+ 1.0, -1.0, -1.0, -1.0, -1.0, 
+ 1.0, -1.0, -1.0, -1.0, 
+ 1.0, -1.0, -1.0, 
+ 1.0, 
+ 1.0/
DATA PRVEC /
+ 0.52254103, 0.13331932, 0.53598671, -7.30258353, -6.33348013, 
+ 5.01432503, 1.07135777, 6.89175648, 5.57001009, 
+ 0.73962239, -5.97893648, -5.25519323, 
+ 323.73106725, 186.47187109, 
+ 139.87165376/

DO J=1,5
APRO(J)=APR(J, J)
DO I=1,5
IF(I.GE.J) THEN
BRBR(I, J)=BRBRVEC((J-1)*5-J*(J-1)/2+I)
BRBR(J, I)=BRBR(I, J)
R1PR(I, J)=R1PRVEC((J-1)*5-J*(J-1)/2+I)
R1PR(J, I)=R1PR(I, J)
PR(I, J)=PRVEC((J-1)*5-J*(J-1)/2+I)
PR(J, I)=PR(I, J)
ZR(I, J)=ZRVEC((J-1)*5-J*(J-1)/2+I)
ZR(J, I)=ZR(I, J)
END IF
END DO
END DO

Parameters for nonlinear pitch channel filters.

DATA ALPQ /0.0/, ALPQMAX /1.0/, Q2Q /200.0/, MUQ /2.0E-6 /
DATA APQ/
+0.0, 0.0, 0.0, 
+0.0, 0.0,
+ 1.0, 0.0, 0.0, 0.0, 0.0,
+ 5.86874341E-001, 7.17980102E-003, -2.89593935E-004,
+ -2.79035083E-002, 0.00000000E+000,
+ 0.0, 0.0, 0.0, 0.0, 0.0,
+ 5.21851607E-003, 6.08715277E-005, 7.17980102E-003,
+ 9.99220787E-001, -1.0/
DATA BRBQVEC /
+ 3.49492524E-002, 4.07666539E-004, 1.17416612E-001,
+ -5.21851607E-003, 0.0, 0.0, 0.0, 0.0,
+ 4.75523782E-006, 1.36960937E-003, -2.79035083E-002,
+ 9.99220787E-001, -1.0/
DATA R1PQVEC /
+ 7.79212948E-004, 0.0, 0.0, 0.0, 0.0,
+ 0.00000000E+000, 0.0, 0.00000000E+000, 0.0,
+ 0.0, 0.0, 0.0, 0.0, 0.0,
+ 7.79212948E-004, 0.0,
+ 0.0/
DATA R2IQ /
+ 7.79212948E-004/
DATA AVQ /
+ 1.87020070E-001, -2.18150087E-003, 6.28318531E-001,
+ 1.00000000E+000, 0.00000000E+000, 0.00000000E+000,
+ 0.00000000E+000, 0.00000000E+000, 0.00000000E+000,
+ 6.28318531E-001/
DATA BVQ /
+ -6.69716293E+000, -7.81192456E+002, -2.25000000E+001/
DATA DQCQ /
+ 35.80986220, 0.00000000, 35.80986220/
DATA DQDQ /
+ 1.28234623E+003/
DATA ZQVEC /
+ 1.0, -1.0, -1.0, -1.0, -1.0,
+ 1.0, -1.0, -1.0, -1.0,
+ 1.0, -1.0, -1.0,
+ 1.0,
+ 1.0/
DATA PQVEC /
+ 0.52254103, 0.13331932, 0.53598671, -7.30258353, -6.33348013,
+ 5.01432503, 1.07135777, 6.89175648, 5.57001008,
+ 0.73962239, -5.97893648, -5.25519323,
+ 323.73106725, 186.47187109, 139.87165376/
DATA PQVEC /
+ 0.52254103, 0.13331932, 0.53598671, -7.30258353, -6.33348013,
+ 5.01432503, 1.07135777, 6.89175648, 5.57001008,
+ 0.73962239, -5.97893648, -5.25519323,
+ 323.73106725, 186.47187109, 139.87165376/

do j=1,5
apqo(j)=apq(j,j)
do i=1,5
if (i.ge.j) then
  brbq(i,j)=brbqvec((j-1)*5-j*(j-1)/2+i)
brbq(j,i)=brbq(i,j)
r1pq(i,j)=r1pqvec((j-1)*5-j*(j-1)/2+i)
r1pq(j,i)=r1pq(i,j)
pq(i,j)=pqvec((j-1)*5-j*(j-1)/2+i)
pq(j,i)=pq(i,j)
zq(i,j)=zqvec((j-1)*5-j*(j-1)/2+i)
zq(j,i)=zq(i,j)
end if
end do
end do

Parameters for nonlinear roll channel filters.
DATA ALPP /0.0/, ALPPMAX /1.0/, Q2P /120.0/, MUP /2.0E-6 /

DATA APP /
+1.45728460E-004, -1.69985373E-006, -4.89593935E-004,
+2.79035083E-002, 0.0,
+1.0, 0.0, 0.0, 0.0, 0.0,
+1.86874341E-001, 2.17980102E-003, -4.89593935E-004,
+2.79035083E-002, 0.00000000E+000,
+0.0, 0.0, 0.0, 0.0, 0.0,
+5.21851607E-003, 6.08715277E-005, 1.75322913E-002,
+9.99220787E-001, -1.0/

DATA BRBPVEC /
+3.49492524E-002, 4.07666539E-004, 1.17416612E-001,
+5.21851607E-003, 0.0,
+4.75523782E-006, 1.36960937E-003,
+6.08715277E-005, 0.0,
+3.94476555E-001, -1.75322913E-002,
+7.79212948E-004, 0.0,
+0.0/

DATA R1PPVEC /
+7.79212948E-004, 0.0, 7.79212948E-004, 0.0,
+2.79035083E-002, 0.00000000E+000, 0.0,
+9.99220787E-001, 0.0/

DATA R2IP /
+7.79212948E-004/

DATA AVP /
+1.87020070E-001, -2.18150087E-003, -6.28318531E-001,
+1.00000000E-000, 0.00000000E+000, 0.00000000E+000,
+0.00000000E+000, 0.00000000E+000, -6.28318531E-001/

DATA BVP /
+6.69716293E+000, -7.81192456E-002, -2.25000000E+001/

DATA DQCP / 35.80986220, 0.00000000, 35.80986220/

DATA DQDP / 1.28234623E+003/

DATA ZPVEC /
+1.0, -1.0, -1.0, -1.0, -1.0,
+1.0, -1.0, -1.0, -1.0,
+1.0, -1.0, -1.0,
+1.0, -1.0,
+1.0/

DATA PPVEC /
+0.52254103, 0.13331932, 0.53598361, -7.30258353, -6.3338013,
+5.01432503, 1.07135777, 6.89175648, 5.57001008,
+0.73962239, -5.97893648, -5.25519323,
+323.73106725, 186.47187109,
+139.87165376/

DO J=1,5
APPO(J)=APP(J,J)
DO I=1,5
IF(I.GE.J) THEN
BRBP(I,J)=BRBPVEC((J-1)*5-J*(J-1)/2+I)
BRBP(J,I)=BRBP(I,J)
R1PP(I,J)=R1PPVEC((J-1)*5-J*(J-1)/2+I)
R1PP(J,I)=R1PP(I,J)
PP(I,J)=PPVEC((J-1)*5-J*(J-1)/2+I)
PP(J,I)=PP(I,J)
ZP(I,J)=ZPVEC((J-1)*5-J*(J-1)/2+I)
END
\[
ZP(J,I)=ZP(I,J)
\]

\[
\text{END IF}
\]

\[
\text{END DO}
\]

\[
\text{c Parameters for nonlinear heave channel filters.}
\]

\[
\text{DATA ALPZ }/0.0/,\ ALPZMAX /0.2/,\ Q2Z /1.0,2.0/,\ MUZ /1.0E-7 /
\]

\[
\text{DATA APZ }/
+0.060606,\ 0.139394,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.567713,\ -0.767713,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.000000,\ 0.000000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.000000,\ 0.000000,\ 1.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.000000,\ 0.000000,\ 0.0000,\ 1.0000,\ 0.0000,\ 0.0000,
+1.717157,\ -2.282843,\ 0.0000,\ 0.0000,\ 0.0000,\ -62.831853/
\]

\[
\text{DATA BRBZVEC }/
+2.948629,\ 3.920000,\ 0.0000,\ 0.0000,\ 1.717157,\ 0.0000,
+5.211371,\ 0.0000,\ 0.0000,\ 2.282843,\ 0.0000,\ 0.0000,
+0.000000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.000000,\ 0.0000,\ 0.0000,\ 1.0000,\ 0.0000,\ 0.0000,
+0.000000/
\]

\[
\text{DATA R1PZVEC }/
+200.0000,\ 200.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+200.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+40.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+400.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+40.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.0000,
+0.0000/
\]

\[
\text{DATA AVZ }/-0.060606,\ 0.139394,\ -0.567713,\ -0.767713/
\]

\[
\text{DATA BVZ }/1.717157,\ 2.282843/
\]

\[
\text{DATA ZZVEC }/
+1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,
+1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,
+1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,
+1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,
+1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,\ -1.0,
+1.0/
\]

\[
\text{DATA PZVEC }/
+910.190970,\ -315.435226,\ -1.160668,\ -79.558688,\ -844.169063,
+13.404542,\ 327.670260,\ -60.259856,\ -315.797740,\ -198.948877,
+3.146942,\ 193.813146,\ 269.544193,\ 145.881379,\ 2.202307,
+1160.148818,\ 888.175899,\ 13.591950,\ 1946.358732,\ 30.388540,
+0.480397/
\]
DO J=1,6
    APZO(J)=APZ(J,J)
    DO I=1,6
        IF(I.GE.J) THEN
            BRBZ(I,J)=BRBZVEC((J-1)*6-J*(J-1)/2+I)
            BRBZ(J,I)=BRBZ(I,J)
            R1PZ(I,J)=R1PZVEC((J-1)*6-J*(J-1)/2+I)
            R1PZ(J,I)=R1PZ(I,J)
            PZ(I,J)=PZVEC((J-1)*6-J*(J-1)/2+I)
            PZ(J,I)=PZ(I,J)
            ZZ(I,J)=ZZVEC((J-1)*6-J*(J-1)/2+I)
            ZZ(J,I)=ZZ(I,J)
        END IF
    END DO
END DO

Nonlinear Scaling Coefficients
DATA GX4/0.5,-0.05,0.002/
DATA GY4/0.4,-0.035,0.001/
DATA GZ40/0.6,-0.082,0.0038/
DATA GZ4S/2.0,-0.05,0.0/
DATA GP4/0.3,-0.3,0.1/
DATA GQ4/0.3,-0.3,0.1/
DATA GR4/1.1,-1.46,0.64/

Translational and Rotational Limits
DATA AMX40/10./,BMX4/1./,AMX4S/20./

Augmented Turbulence Parameters
DATA G2D0,G2D1,G2N0,G2N1,G2N2+/25.0,12.5,2.5,12.0,14.4/
DATA GT4/1.2/
RETURN
END
Appendix B. Non-linear optimal algorithm: filtering at platform centroid (original) vs. filtering at PS (modified)

Nonlinear optimal algorithm when filtered at the centroid of the motion platform (original) versus filtering at the pilot station (modified).

General note: if only one or two curves are visible on the plot, assume that the remaining curves are underneath the visible ones.

1. Pitch

![Figure B.1. 1.](image-url)

Figure B.1. 1.
Figure B.1. 2.
Figure B.1. 3.
Figure B.1. 4.
Figure B.1. 5.
Figure B.1.6.
Figure B.1. 7.
2. Roll

Figure B.2. 1.
Figure B.2. 2.
Figure B.2. 3.

Figure B.2. 4.
Figure B.2. 5.
Figure B.2.6.
Figure B.2. 7.
3. Yaw

Figure B.3. 1.
Figure B.3. 2.
Figure B.3. 3.
Figure B.3. 4.
Figure B.3. 5.
Figure B.3. 6.
Figure B.3. 7.
4. Surge

![Graphs showing specific forces and angular rates](image)

Figure B.4.1.
Figure B.4. 2.
Figure B.4. 3.

Figure B.4. 4.
Figure B.4. 5.
Figure B.4. 6.
Figure B.4. 7.
5. Sway

Figure B.5. 1.
Figure B.5.2.
Figure B.5. 3.

Figure B.5. 4.
Figure B.5. 5.
Figure B.5. 6.
Figure B.5. 7.
6. Heave

Figure B.6. 1.
Figure B.6. 2.
Figure B.6.3.
Figure B.6.4.
Figure B.6. 5.
Figure B.6. 6.
Figure B.6. 7.
Appendix C. Non-linear optimal algorithm: original vs. augmented

General note: if only one or two curves are visible on the plot, assume that the remaining curves are underneath the visible ones.

1. Pitch

![Plot of specific force and angular rate](image)

Figure C.1. 1.
Figure C.1. 2.
Figure C.1.3.
Figure C.1. 4.
Figure C.1.5.
Figure C.1. 6.
Figure C.1. 7.
2. Roll

Figure C.2. 1.
Figure C.2. 2.
Figure C.2.3.

Specific Force at A/C Pilot Head

AF-X (m/s^2)

AF-Y (m/s^2)

AF-Z (m/s^2)

Specific Force at Simu. Pilot Head

SF-X (m/s^2)

SF-Y (m/s^2)

SF-Z (m/s^2)

Aircraft Angular Rate

AW-p (deg/s)

AW-q (deg/s)

AW-r (deg/s)

Platform Angular Rate

SW-p (deg/s)

SW-q (deg/s)

SW-r (deg/s)
Figure C.2. 4.

Figure C.2. 5.
Figure C.2.6.
Figure C.2. 7.
3. Yaw

Figure C.3. 1.
Figure C.3. 2.
Figure C.3. 3.
Figure C.3. 4.
Figure C.3. 5.
Figure C.3. 6.
Figure C.3. 7.
4. Surge

![Surge Diagram]

Figure C.4. 1.
Figure C.4. 2.
Figure C.4.3.
Figure C.4.4.
Figure C.4. 5.
Figure C.4. 6.
Figure C.4. 7.
5. Sway

Figure C.5. 1.
Figure C.5. 2.
Figure C.5. 3.
Figure C.5. 4.

Figure C.5. 5.
Figure C.5. 6.
Figure C.5. 7.
6. Heave

Figure C.6. 1.
Figure C.6. 2.
Figure C.6. 3.
Figure C.6. 4.
Figure C.6. 5.
Figure C.6. 6.
Figure C.6. 7.
Appendix D. Non-linear optimal algorithm: original applied at PS (filt. @ PS Original) vs. augmented (Augmented)

General note: if only one or two curves are visible on the plot, assume that the remaining curves are underneath the visible ones.

1. Pitch

![Graphs showing A/C Accel. at MB Centroid and Platform Accel. at MB Centroid with filters and augmented versions.]

Figure D.1.1.
Figure D.1. 2.
Figure D.1. 3.
Figure D.1. 4.
Figure D.1. 5.
Figure D.1. 6.
Figure D.1. 7.
2. Roll

![Graphs showing A/C Accel. at MB Centroid and Platform Accel. at MB Centroid](image)

Figure D.2. 1.
Figure D.2. 2.
Figure D.2.3.
Figure D.2. 4.
Figure D.2.5.
Figure D.2. 6.
Figure D.2. 7.
3. Yaw

Figure D.3. 1.
Figure D.3. 2.
Figure D.3. 3.
Figure D.3. 4.
Figure D.3. 5.
Figure D.3. 6.
Figure D.3. 7.
4. Surge

![Figure D.4. 1.]

Figure D.4. 1.
Figure D.4. 2.
Figure D.4. 3.
Figure D.4. 4.
Figure D.4. 5.
Figure D.4. 6.
Figure D.4. 7.
5. Sway

Figure D.5.1.
Figure D.5. 2.
Figure D.5. 3.
Figure D.5. 4.
Figure D.5. 5.
Figure D.5. 6.
Figure D.5. 7.
6. Heave

Figure D.6.1.
Figure D.6. 2.
Figure D.6. 3.
Figure D.6. 4.
Figure D.6. 5.
Figure D.6. 6.
Figure D.6. 7.
References


1. REPORT DATE (DD-MM-YYYY) 01-05-2012
2. REPORT TYPE Contractor Report
3. DATES COVERED (From - To)
4. TITLE AND SUBTITLE Nonlinear Motion Cueing Algorithm: Filtering at Pilot Station and Development of the Nonlinear Optimal Filters for Pitch and Roll
5a. CONTRACT NUMBER NNL06AA74T
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER
6. AUTHOR(S) Zaychik, Kirill B.; Cardullo, Frank, M.
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, Virginia 23681-2199
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001
10. SPONSOR/MONITOR'S ACRONYM(S) NASA
11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/CR-2012-217567
12. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 54 Availability: NASA CASI (443) 757-5802
13. SUPPLEMENTARY NOTES This report was prepared by State University of New York, Binghamton, NY, under NASA contract NNL06AA74T with UNISYS Corporation, Reston, VA. Langley Technical Monitor: Jacob A. Houck
14. ABSTRACT Telban and Cardullo have developed and successfully implemented the non-linear optimal motion cueing algorithm in the Visual Motion Simulator (VMS) at the NASA Langley Research Center in 2005. The latest version of the non-linear algorithm performed filtering of motion cues in all degrees-of-freedom except for pitch and roll. This manuscript describes the development and implementation of the non-linear optimal motion cueing algorithm for the pitch and roll degrees of freedom. Presented results indicate improved cues in the designated channels as compared to the original design. To further advance motion cueing in general, this manuscript describes modifications to the existing algorithm, which allow for filtering at the location of the pilot’s head as opposed to the centroid of the motion platform. The rational for such modification to the cueing algorithms is that the location of the pilot’s vestibular system must be taken into account as opposed to the off-set of the centroid of the cockpit relative to the center of rotation alone. Results provided in this report suggest improved performance of the motion cueing algorithm.
15. SUBJECT TERMS Cueing Algorithms; Flight Simulator; Motion Systems
16. SECURITY CLASSIFICATION OF:
a. REPORT U
b. ABSTRACT U
c. THIS PAGE U
17. LIMITATION OF ABSTRACT UU
18. NUMBER OF PAGES 241
19a. NAME OF RESPONSIBLE PERSON STI Help Desk (email: help@sti.nasa.gov)
19b. TELEPHONE NUMBER (Include area code) (443) 757-5802