Implementation and Testing of Turbulence Models for the F18-HARV Simulation

Jessie C. Yeager
Lockheed Martin Engineering & Sciences, Hampton, Virginia

March 1998
## Table of Contents

- **Introduction** ........................................................................................................... 1
- **Symbols** .................................................................................................................. 1
- **ACSL Test Program (GUSTMDL)** ............................................................................ 4
  - Equations .................................................................................................................. 4
    - Continuous Model ................................................................................................. 4
    - Tustin Model .......................................................................................................... 6
    - MIL STD Model ..................................................................................................... 8
  - Variable Definition ................................................................................................... 9
  - Code .......................................................................................................................... 11
- **Time History Plots** .................................................................................................. 30
- **Power Spectral Densities** ........................................................................................ 35
  - Equations ................................................................................................................ 35
  - Code ......................................................................................................................... 39
  - Plots .......................................................................................................................... 46
- **Measured Statistics** .................................................................................................. 53
  - Equations ................................................................................................................ 53
  - Results ....................................................................................................................... 53
- **Theoretical Statistics** ................................................................................................ 60
  - Equations ................................................................................................................ 60
  - Code .......................................................................................................................... 61
  - Autocorrelation ......................................................................................................... 69
    - Equations .............................................................................................................. 70
    - Code ....................................................................................................................... 71
    - Plots ......................................................................................................................... 77
- **Aircraft Simulation** .................................................................................................... 80
  - Code .......................................................................................................................... 80
  - Time History Plots .................................................................................................... 95
- **References** ............................................................................................................... 141
Abstract

This report presents three methods of implementing the Dryden power spectral density model for atmospheric turbulence. Included are the equations which define the three methods and computer source code written in Advanced Continuous Simulation Language to implement the equations. Time-history plots and sample statistics of simulated turbulence results from executing the code in a test program are also presented. Power spectral densities were computed for sample sequences of turbulence and are plotted for comparison with the Dryden spectra. The three model implementations were installed in a nonlinear six-degree-of-freedom simulation of the High Alpha Research Vehicle airplane. Aircraft simulation responses to turbulence generated with the three implementations are presented as plots.

Introduction

An important part of an aircraft simulation is the capability to simulate atmospheric turbulence. Such a capability is useful in evaluating flight control system performance and assessing control effector activity. The Dryden turbulence model of power spectral density as specified in MIL_STD_1797A (ref. 1) is frequently used as the basis for turbulence simulation. The turbulence model is also discussed in reference 2. This report presents source code for three different approaches to implementing the model in a six-degree-of-freedom aircraft simulation and presents results obtained with these implementations.

Symbols

Variable:

\( a \) \hspace{1cm} Intermediate variable defined by equation (44); used in equation (43)

\( a_i, a_j \) \hspace{1cm} Intermediate variable for the i-th or the j-th component of turbulence ; used in equations (52) and (53) as defined by equations (54); used in equations (56) as defined by equations (57)

\( b \) \hspace{1cm} Intermediate variable defined by equation (44)

\( b_w \) \hspace{1cm} Reference wing span

\( C_{BL} \) \hspace{1cm} Mapping constant used in the Tustin transform to relate frequencies in the digital filter response to frequencies in the analog response for the u-, v-, and w-components

\( C_{BL_i} \) \hspace{1cm} Mapping constant used in the Tustin transform to relate frequencies in the digital filter response to frequencies in the analog response for the i-th component, \( i = p, q, r \)

\( c \) \hspace{1cm} Intermediate variable defined by equation (44)

\( d \) \hspace{1cm} Intermediate variable defined by equation (44)

\( e \) \hspace{1cm} Intermediate variable defined by equation (44)

\( f \) \hspace{1cm} Intermediate variable defined by equation (44)

\( G_d(\omega) \) \hspace{1cm} Power spectral density of the i-th component of turbulence, discrete model

\( H_i(z) \) \hspace{1cm} Transfer function used to compute the i-th component of turbulence in the z-domain

\( H_i(s) \) \hspace{1cm} Transfer function used to compute the i-th component of turbulence in the s-domain
\( \tilde{H}_i(e^{j\omega T_v}) \) Transfer function used to compute the frequency response of \( H_i(z = e^{j\omega T_v}) \)

\( K_i \) Intermediate variable defined by equation (44)

\( k_i, k_j \) Coefficient in difference equation in the MIL STD model for calculating the i-th or the j-th component of turbulence; defined by equation (57)

\( L_i \) Scale length of i-th component of turbulence

\( M_{r^+}, M_{r^-} \) Intermediate variables defined by equation (51)

\( \hat{M}_g \) Intermediate variable defined by equation (60)

\( \hat{m}_{g_i}^{(j)} \) Sample mean of the j-th sample sequence of the i-th component of turbulence

\( m_g \) Mean, or expected value, of \( g \)

\( N \) Number of samples in turbulence sequence

\( R_{i-} \) Intermediate variable defined by equation (74)

\( R_{j-} \) Intermediate variable defined by equation (72)

\( R_{i+} \) Intermediate variable defined by equation (77)

\( R_{j+} \) Intermediate variable defined by equation (75)

\( R_{j-, \tau} \) Intermediate variable defined by equation (73)

\( R_{j+, \tau} \) Intermediate variable defined by equation (76)

\( R_i(\tau) \) Autocorrelation function of the i-th component of turbulence

\( s \) Laplace transform variable

\( S_i(\omega) \) Two-sided power spectral density of i-component of turbulence, continuous model

\( T \) Length of turbulence sequence \( (T = NT_v) \)

\( T_v \) Sample interval in digital simulation of turbulence

\( V \) Total airspeed

\( X \) Intermediate variable defined by equation (57)

\( x_i \) Intermediate variable used in computation of i-th component of turbulence

\( Y \) Intermediate variable defined by equation (57)

\( z \) z-transform variable

\( v_i, v_i(k) \) Random white noise sequence used as inputs to the differential equations or difference equations for computation of the i-th component of turbulence

\( v_i(z) \) z-transform of \( v_i(k) \)

\( \xi_i, \xi_i(k) \) Random sequence representing the i-th component of turbulence

\( \xi_i(z) \) z-transform of \( \xi_i(k) \)

\( \pi \) \( \pi \) ( = \( \text{arc} \cos(-1) \))

\( \Phi_i(\omega) \) Single-sided power spectral density of i-th component of turbulence, continuous model
\( \tau_i \)  
Time constant of the i-th component of turbulence

\( \sigma_i \)  
Desired root-mean-square value of the i-th component of turbulence

\( \sigma_{ji}^{(i)} \)  
Sample standard deviation of the j-th sample sequence of the i-th component of turbulence

\( \hat{\Sigma}_g \)  
Standard deviation of \( g \)

\( \Sigma(\hat{\sigma}_i) \)  
Standard deviation of the sample standard deviation of \( i \)  
\[ = \sqrt{E\left\{ (\hat{\sigma}_i - E[\hat{\sigma}_i])^2 \right\}} \]

\( \Sigma(\hat{m}_i) \)  
Standard deviation of the sample mean of \( i \)  
\[ = \sqrt{E\left\{ (\hat{m}_i - E[\hat{m}_i])^2 \right\}} \]

\( \omega \)  
Angular frequency

\( \omega_w \)  
Reciprocal of time constant for the w-component of turbulence \( (= 1/\tau_w) \)

\( \omega_i \)  
Intermediate variable defined by equation (44) and equation (53)

Subscripts:

\( i \)  
Denotes turbulence component (u, v, w, p, q, or r)

\( j \)  
Denotes turbulence component (u, v, w, p, q, or r)

\( u \)  
Denotes linear velocity or turbulence component along x-axis (body axis)

\( v \)  
Denotes linear velocity or turbulence component along y-axis (body axis)

\( w \)  
Denotes linear velocity or turbulence component along z-axis (body axis)

\( p \)  
Denotes rotational velocity or turbulence component about x-axis (body axis)

\( q \)  
Denotes rotational velocity or turbulence component about y-axis (body axis)

\( r \)  
Denotes rotational velocity or turbulence component about z-axis (body axis)

Superscripts:

\( (j) \)  
Denotes j-th sample sequence

Operators and Notation:

\( E\{ \cdot \} \)  
Denotes expected value of argument

\( \mathcal{F}^{-1}\{ \cdot \} \)  
Denotes inverse Fourier transform of argument

\[ \sum_{k=1}^{N} [\cdot] \]  
Denotes summation of argument over \( k \) from 1 to \( N \)

\( |\cdot| \)  
Denotes absolute value of argument

Abbreviations:

ACSL  
Advanced Continuous Simulation Language

HARV  
High Alpha Research Vehicle

MIL STD  
Military Standard
PSD  power spectral density
rms  root-mean-square

ACSL Test Program (GUSTMDL)

A test program named GUSTMDL was developed to evaluate implementations of the three models prior to installation in the aircraft simulation. GUSTMDL contains algorithms to simulate linear (u, v, w) and rotational (p, q, r) velocity components of turbulence using the three models: continuous, Tustin transform, and Military Standard (MIL STD) algorithms. The GUSTMDL program was written using the Advanced Continuous Simulation Language (ACSL) to be easily implemented into the F18 High Alpha Research Vehicle (HARV) simulation (refs. 3 and 4) used by Dynamics Control Branch (DCB) researchers.

Equations

The algebraic, differential, and difference equations describing the Dryden turbulence model are presented in this section. These equations were developed from the Dryden spectral model contained in reference 1 and were furnished by the Government. The equations were transformed into ACSL source code and implemented in the GUSTMDL program. Comments were included in the program to correlate code with the equations in the following section by equation number.

The equations are separated according to each of the three turbulence models. The continuous model equations were implemented in the ACSL DERIVATIVE BLOCK of the GUSTMDL program. The Tustin model and MIL STD equations were implemented in the ACSL DISCRETE BLOCK named GUST.

Continuous Model

Equations (1) through (17) below define the continuous implementation of the Dryden turbulence model. In these equations the variables $\xi_u$, $\xi_v$, $\xi_w$, $\xi_p$, $\xi_q$, $\xi_r$ represent the body-axis u-, v-, w-, p-, q-, and r-components of turbulence, respectively. The variables $V_u$, $V_v$, $V_w$, $V_p$ represent the Gaussian noise forcing functions for the u-, v-, w-, and p-components, respectively.

Equations (1) and (2) are the differential equations defining the w-component of the turbulence.

$$\dot{x}_w = -2\tau_w^{-1} \dot{x}_w - \tau_w^{-2} x_w + \sigma_w \tau_w^{-3/2} T_v^{-1/2} V_v$$

The desired root-mean-square (rms) velocity magnitude of the w-component of the turbulence is represented by $\sigma_w$.

$$\xi_w = x_w + \tau_w \sqrt{3} \dot{x}_w \quad \text{and} \quad \dot{\xi}_w = \dot{x}_w + \tau_w \sqrt{3} \ddot{x}_w$$

where

$$\tau_w = \frac{L_w}{V}$$

The v-component can be obtained from equations (1) through (2) by substituting the subscript $v$ for the subscript $w$. The time constant for the v-component is given in equation (4).

$$\tau_v = \frac{L_v}{V}$$

The differential equation defining the u-component is
\[ \dot{\xi}_u = -\frac{1}{\tau_u} \xi_u + \sigma_u \frac{2}{\tau_u T_v} \nu_u \]  

(5)

where \( \tau_u = \frac{L_u}{V} \)  

(6)

Similarly, the \( p \)-component equations are

\[ \dot{\xi}_p = -\frac{1}{\tau_p} \xi_p + \sigma_p \sqrt{\frac{2}{\tau_p T_v}} \nu_p \]  

(7)

where \( \sigma_p = \frac{1.9}{\sqrt{L_w b_w}} \sigma_w \)  

(8)

\[ \tau_p = \frac{L_p}{V} \]  

(9)

\[ L_p = \frac{\sqrt{L_w b_w}}{2.6} \]  

(10)

The differential equations defining the \( q \)-component are

\[
\begin{align*}
\dot{x}_q &= -\frac{V \pi}{4 b_w} x_q + \frac{\pi}{4 b_w} \xi_w \\
\dot{\xi}_q &= \dot{x}_q \\
&= -\frac{V \pi}{4 b_w} x_q + \frac{\pi}{4 b_w} \xi_w
\end{align*}
\]  

(11)

or

\[ \dot{\xi}_q = -\frac{V \pi}{4 b_w} \xi_q + \frac{\pi}{4 b_w} \dot{\xi}_w \]  

(12)

A time constant for equation (12) can be defined by

\[ \tau_q = \frac{4 b_w}{\pi V} \]  

(13)

The rms value of the \( q \)-component for equation (12) is given approximately by

\[ \sigma_q \approx \sqrt{\frac{\pi}{2 L_w b_w}} \sigma_w \]  

(14)
Similarly, the r-component equation is

$$\dot{\xi}_r = -\frac{V\pi}{3b_w} \xi_r + \frac{\pi}{3b_w} \dot{\xi}_v$$  \hspace{1cm} (15)$$

A time constant for equation (15) can be defined by

$$\tau_r = \frac{3b_w}{\pi V}$$  \hspace{1cm} (16)$$

The rms value of the r-component for equation (15) is given approximately by

$$\sigma_r \approx \sqrt{\frac{2\pi}{3L_w b_w}} \sigma_w$$  \hspace{1cm} (17)$$

**Tustin Model**

The Tustin model can be implemented with recursive difference equations to compute components of turbulence. The definitions of \(\tau_u\) in equations (3), (4), (6), (9), (13), and (16) for the continuous model were used in the Tustin turbulence equations. Similarly the definitions of \(\sigma_v\) in equations (8), (14), and (17) apply. The Gaussian noise forcing functions are the same as used in the continuous model.

The recursive difference equation to compute the u-component of the turbulence is

$$\xi_u(k) = \left[ 1 - C_{BL} \tau_u \right] \xi_u(k-1) + \left[ \sigma_u \sqrt{\frac{2\tau_u}{T_v}} \right] \left[ \nu_u(k) + \nu_u(k-1) \right]$$  \hspace{1cm} (18)$$

where \(C_{BL}\) is given by

$$C_{BL} = \frac{1}{\tau_u} \cot \left( \frac{T_v}{2\tau_u} \right)$$  \hspace{1cm} (19)$$

The recursive difference equation to compute the w-component is

$$\xi_w(k) = -\frac{2(\omega_w^2 - C_{BL}^2)}{(\omega_w + C_{BL})^2} \xi_w(k-1) - \frac{(\omega_w - C_{BL})^2}{(\omega_w + C_{BL})^2} \xi_w(k-2)$$

$$+ \frac{\sigma_w \sqrt{3\omega_w}}{(\omega_w + C_{BL})^2} \left[ \left( C_{BL} + \frac{\omega_w}{\sqrt{3}} \right) v_w(k) + \frac{2\omega_w}{\sqrt{3}} v_w(k-1) \right. + \left. \left( \frac{\omega_w}{\sqrt{3}} - C_{BL} \right) v_w(k-2) \right]$$  \hspace{1cm} (20)$$

The \(\nu_u\)-component can be obtained from equation (20) by substituting the subscript \(v\) for the subscript \(w\).

The recursive difference equation to compute the \(\nu\)-component of the turbulence is
\[ \xi_p(k) = -\frac{1 - C_{BL_p} \tau_p}{1 + C_{BL_p} \tau_p} \xi_p(k - 1) + \left[ \frac{\sigma_p \sqrt{2 \tau_p}}{T_v} \right] \left[ v_p(k) + v_p(k - 1) \right] \]  

where

\[ C_{BL_p} = \frac{1}{\tau_p} \cot \left( \frac{T_v}{2 \tau_p} \right) \]  

The difference equation for \( \xi_q(k) \) is

\[ \xi_q(k) = \left( \frac{1 - 4b_w C_{BL_q} / \pi V}{1 + 4b_w C_{BL_q} / \pi V} \right) \xi_q(k - 1) + \left( \frac{C_{BL_q}}{V(1 + 4b_w C_{BL_q} / \pi V)} \right) \left[ \xi_w(k) - \xi_w(k - 1) \right] \]  

where

\[ C_{BL_q} = \frac{1}{\tau_q} \cot \left( \frac{T_v}{2 \tau_q} \right) = \frac{\pi V}{4b_w} \cot \left( \frac{\pi V T_v}{8b_w \tau_q} \right) \]  

The equations for the r-component are identical to those for the q-component except that the factor \((4b_w / \pi)\) in equations (23) and (24) is replaced by the factor \((3b_w / \pi)\). The resulting difference equation for \( \xi_r \) is

\[ \xi_r(k) = \left( \frac{1 - 3b_w C_{BL_r} / \pi V}{1 + 3b_w C_{BL_r} / \pi V} \right) \xi_r(k - 1) + \left( \frac{C_{BL_r}}{V(1 + 3b_w C_{BL_r} / \pi V)} \right) \left[ \xi_v(k) - \xi_v(k - 1) \right] \]  

where
The simple difference equation for the derivative of each turbulence component is

\[ \dot{\xi}(k) = \frac{\xi(k) - \xi(k-1)}{T_v} \]  

(27)

The difference equation form for the derivative of the u-component is

\[ \dot{\xi}_u(k) = -\frac{1 - \tau_u C_{BL}}{1 + \tau_u C_{BL}} \dot{\xi}_u(k-1) + \sigma_u \sqrt{\frac{2\tau_u}{T_v}} \frac{C_{BL}}{1 + \tau_u C_{BL}} \{ v_u(k) - v_u(k-1) \} \]  

(28)

The difference equation form for the derivative of the w-component is

\[ \dot{\xi}_w(k) = -2 \left( \frac{1 - \tau_w C_{BL}}{1 + \tau_w C_{BL}} \right) \dot{\xi}_w(k-1) - \left( \frac{1 - \tau_w C_{BL}}{1 + \tau_w C_{BL}} \right)^2 \dot{\xi}_w(k-2) + \frac{\sigma_w}{T_v} \sqrt{\frac{\tau_w C_{BL}}{(1 + \tau_w C_{BL})^2}} \left\{ \left( 1 + \sqrt{3} \tau_w C_{BL} \right) v_w(k) - 2 \sqrt{3} \tau_w C_{BL} v_w(k-1) - \left( 1 - \sqrt{3} \tau_w C_{BL} \right) v_w(k-2) \right\} \]  

(29)

**MIL STD Model**

The difference equations defining the MIL STD model were obtained from reference 1. Equations (30) through (35) were furnished by the Government and implemented to calculate the u-, v-, w-, p-, q-, and r-components of turbulence. The definitions of \( \tau \) and \( \sigma \) for the continuous model were also used in the MIL STD turbulence equations. Again, the Gaussian noise forcing functions are the same ones used in the continuous model.

\[ \dot{\xi}_u(k) = \left( 1 - \frac{T_v}{\tau_u} \right) \xi_u(k-1) + \sigma_u \sqrt{\frac{2T_v}{\tau_u}} v_u(k) \]  

(30)

\[ \dot{\xi}_v(k) = \left( 1 - \frac{2T_v}{\tau_v} \right) \xi_v(k-1) + \sigma_v \sqrt{\frac{4T_v}{\tau_v}} v_v(k) \]  

(31)

\[ \dot{\xi}_w(k) = \left( 1 - \frac{2T_v}{\tau_w} \right) \xi_w(k-1) + \sigma_w \sqrt{\frac{4T_v}{\tau_w}} v_w(k) \]  

(32)

\[ \dot{\xi}_p(k) = \left( 1 - \frac{T_v}{\tau_p} \right) \xi_p(k-1) + \sigma_p \sqrt{\frac{2T_v}{\tau_p}} v_p(k) \]  

(33)
\[ \xi_q(k) = \left(1 - \frac{T_V}{\tau_q}\right)\xi_q(k-1) + \frac{\pi}{4b_w} \left[\xi_w(k) - \xi_w(k-1)\right] \tag{34} \]

\[ \xi_r(k) = \left(1 - \frac{T_V}{\tau_r}\right)\xi_r(k-1) + \frac{\pi}{3b_w} \left[\xi_v(k) - \xi_v(k-1)\right] \tag{35} \]

**Variable Definition**

Table 1 correlates the algebraic symbols used in the equations with variable names used in coding the test program GUSTMDL. Variables are listed for each of the three implemented turbulence models. The equation number column was included as a reference to the equations listed in the previous section. Comments are included in the GUSTMDL code to help the reader identify the equations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Turbulence Model</th>
<th>Equation no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_p )</td>
<td>FILNP</td>
<td>FILNP</td>
</tr>
<tr>
<td>( V_p(k) )</td>
<td>FILNP</td>
<td>FILNP</td>
</tr>
<tr>
<td>( V_u )</td>
<td>FILNU</td>
<td>FILNU</td>
</tr>
<tr>
<td>( V_u(k) )</td>
<td>FILNU</td>
<td>FILNU</td>
</tr>
<tr>
<td>( V_v )</td>
<td>FILNV</td>
<td>FILNV</td>
</tr>
<tr>
<td>( V_v(k) )</td>
<td>FILNV</td>
<td>FILNV</td>
</tr>
<tr>
<td>( V_w )</td>
<td>FILNW</td>
<td>FILNW</td>
</tr>
<tr>
<td>( V_w(k) )</td>
<td>FILNW</td>
<td>FILNW</td>
</tr>
<tr>
<td>( \xi_q )</td>
<td>TURBP</td>
<td>MILPK</td>
</tr>
<tr>
<td>( \xi_q(k) )</td>
<td>FILP</td>
<td>MILPK</td>
</tr>
<tr>
<td>( \xi_q(k) )</td>
<td>TURBQ</td>
<td>MILQK</td>
</tr>
<tr>
<td>( \xi_r )</td>
<td>TURBR</td>
<td>MILPK</td>
</tr>
<tr>
<td>( \xi_r(k) )</td>
<td>FILR</td>
<td>MILRK</td>
</tr>
<tr>
<td>( \xi_u )</td>
<td>TURBU</td>
<td>MILUK</td>
</tr>
<tr>
<td>( \xi_u(k) )</td>
<td>FILU</td>
<td>MILUK</td>
</tr>
</tbody>
</table>
Table 1. Concluded

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Turbulence Model</th>
<th>Equation no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\dot{\xi}_v$</td>
<td>TURBV</td>
<td>2</td>
</tr>
<tr>
<td>$\dot{\xi}_v(k)$</td>
<td>FILV</td>
<td>20, 31</td>
</tr>
<tr>
<td>$\dot{\xi}_w$</td>
<td>TURBW</td>
<td>2</td>
</tr>
<tr>
<td>$\dot{\xi}_w(k)$</td>
<td>FILW</td>
<td>20, 32</td>
</tr>
<tr>
<td>$\sigma_p$</td>
<td>SIGP</td>
<td>8</td>
</tr>
<tr>
<td>$\sigma_q$</td>
<td>SIGQ</td>
<td>14</td>
</tr>
<tr>
<td>$\sigma_r$</td>
<td>SIGR</td>
<td>17</td>
</tr>
<tr>
<td>$\sigma_u$</td>
<td>SIGU (=TURBSIG)</td>
<td></td>
</tr>
<tr>
<td>$\sigma_v$</td>
<td>SIGV (=TURBSIG)</td>
<td></td>
</tr>
<tr>
<td>$\sigma_w$</td>
<td>SIGW (=TURBSIG)</td>
<td></td>
</tr>
<tr>
<td>$\tau_p$</td>
<td>TAUP</td>
<td>9</td>
</tr>
<tr>
<td>$\tau_q$</td>
<td>TAUQ</td>
<td>13</td>
</tr>
<tr>
<td>$\tau_r$</td>
<td>TAUR</td>
<td>16</td>
</tr>
<tr>
<td>$\tau_u$</td>
<td>TAU</td>
<td>6</td>
</tr>
<tr>
<td>$\tau_v$</td>
<td>TAU</td>
<td>4</td>
</tr>
<tr>
<td>$\tau_w$</td>
<td>TAU</td>
<td>3</td>
</tr>
<tr>
<td>$b_w$</td>
<td>BWING (=32.4)</td>
<td></td>
</tr>
<tr>
<td>$L_u$</td>
<td>TURBL (=1750.)</td>
<td></td>
</tr>
<tr>
<td>$L_v$</td>
<td>TURBL (=1750.)</td>
<td></td>
</tr>
<tr>
<td>$L_w$</td>
<td>TURBL (=1750.)</td>
<td></td>
</tr>
<tr>
<td>$L_p$</td>
<td>LP</td>
<td>10</td>
</tr>
<tr>
<td>$V$</td>
<td>VTOT</td>
<td></td>
</tr>
<tr>
<td>$T_v$</td>
<td>TSAMP (=0.0125)</td>
<td></td>
</tr>
</tbody>
</table>
The ACSL/FORTRAN source code for the GUSTMDL program (file GUSTMDL.CSL) is presented in this section. The program has MACRO, INITIAL, DYNAMIC, and TERMINAL BLOCKS. The DYNAMIC BLOCK is composed of the DERIVATIVE and DISCRETE BLOCKS. The three DISCRETE BLOCKS are the GUST, DISCRMS, and ASC. The continuous turbulence model is implemented in the DERIVATIVE BLOCK, and the Tustin and MIL STD models are implemented in the DISCRETE GUST BLOCK. Initialization of variables and variables that need to be calculated only one time are implemented in the INITIAL BLOCK. The DISCRETE BLOCK DISCRMS calls a macro to define root mean square and mean statistics for the variables in the three turbulence models. The DISCRETE BLOCK ASC calls FORTRAN subroutines ASCFM and WSTATS which are used to output variables for analysis in a format that is compatible with tools utilized by the DCB researchers.

```
PROGRAM GUSTMDL
!
!******************************************************************************
!
! MACRO SECTION
!
!******************************************************************************!
!
! RMS CALCULATES THE RMS RESPONSE OF A RANDOM VARIABLE.
!******************************************************************************

MACRO RMS(RMSX,X,TIV)
MACRO REDEFINE MSX,MSXD,MSXI
MACRO REDEFINE MSXL,EPS
CONSTANT MSXI = 0.
CONSTANT EPS = 1.E-22
MSXD = TIV*(X**2 - MSX)
MSX = INTVC(MSXD,MSXI)
MSXL = MAX(MSX, EPS)
RMSX = SQRT(MSXL)
MACRO EXIT
MACRO END

!******************************************************************************
!
! INVERTS TIME. USES 1/MINSTP IF TIME = ZERO.
!******************************************************************************

MACRO INVERT(TIV,T,MINSTP)
MACRO RELABEL L1,L2
PROCEDURAL(TIV=T)
  IF(T.EQ.0.) GOTO L1
  TIV=1./T
  GOTO L2
L1..CONTINUE
  TIV=1./MINSTP
L2..CONTINUE
END ! of procedural
MACRO EXIT
MACRO END
```

11
MACRO DRMS(RMSX, MNX , X, N )

MACRO TO COMPUTE SAMPLE STATISTICS OF RANDOM VARIABLE

MACRO REDEFINE SUM, SUMSQ, EPS

CONSTANT SUM = 0. , SUMSQ = 0.
CONSTANT EPS = 1.E-22
SUM = SUM + X
SUMSQ = SUMSQ + X*X
MNX = SUM/N
RMSX = SQRT(MAX((SUMSQ - MNX*SUM)/(MAX(N-1.,EPS)),0.))

MACRO EXIT
MACRO END

MACRO XCORR(XCOR, MXM , X, Y, N, NTAU )

MACRO REDEFINE SUMXY, SUMX, SUMY, MNX, MNY, NS, XDLY, YDLY, NDLY, I
REAL XDLY(100)
MACRO RELABEL LDLY
IF (N .LT. NTAU + i) THEN
   NDLY = N
ELSE
   NDLY = NTAU + 1
ENDIF
DO LDLY I = i, NDLY - 1
   XDLY(NDLY+I-I) = XDLY(NDLY-I)
LDLY.. CONTINUE
! END
XDLY(1) = X
CONSTANT SUMXY = 0., SUMX = 0., SUMY = 0., NS = 0
IF (N .GE. NTAU + 1) THEN
   NS = NS + 1
   SUMXY = SUMXY + XDLY(NDLY)*Y
   SUMX = SUMX + XDLY(NDLY)
   SUMY = SUMY + Y
   MNX = SUMX/NS
   MNY = SUMY/NS
   MXM = MNX * MNY
   XCOR = SUMXY/NS - MXM
ENDIF
MACRO EXIT
MACRO END

INITIAL
ALGORITHM IALG = 4
CINTERVAL CINT = 0.1
MAXTERVAL MXSTP = 0.0125
NSTEPS NSTP = 1
INTEGER PAGSIZ
CONSTANT PAGSIZ = 55

CONSTANT TSTP = 100.0
PI = ACOS(-1.)

INTEGER NUMOUT $ !DETERMINES FREQ OF DATA OUTPUT FOR ASC!
CONSTANT NUMOUT = 1

CONSTANT SAMS = 0.
CONSTANT EPSLON = 1.E-22

CONSTANT SDNOIS = 1., &
TSAMP = .0125, &
TURBL = 1750., &
TURBSIG = 5., &
TURBU0 = 0., &
TURBXVD0 = 0., &
TURBXV0 = 0., &
TURBXWD0 = 0., &
TURBXWO = 0.

! CONSTANT TURBNV = 0., &
TURBNW = 0., &
FILNU = 0., &
FILNV = 0., &
FILNW = 0.

! CONSTANT TURBR0 = 0., &
TURBQ0 = 0., &
TURBP0 = 0., &
BWING = 37.4

! CONSTANT TURBXQ0 = 0., &
TURBXR0 = 0.

! irseed should be pos, odd, integer, 8 digits
!
INTEGER IRSEED
CONSTANT IRSEED = 28545269
GAUSI( IRSEED )
!

SIGU = TURBSIG ! same as sigv,sigw
SIGV = TURBSIG
SIGW = TURBSIG
SIGP = 1.9 / SQRT(TURBL * BWING) * SIGW ! eq.8

! APPROXIMATIONS OF EXPECTED VALUES OF STD. DEV.(P AND R COMPONENTS)
\[ \text{SIGQ} = \text{SQRT}(\pi/(2 \cdot \text{TURBL} \cdot \text{BWING})) \cdot \text{SIGW} \quad ! \text{eq.14} \]
\[ \text{SIGR} = \text{SQRT}(2 \cdot \pi/(3 \cdot \text{TURBL} \cdot \text{BWING})) \cdot \text{SIGW} \quad ! \text{eq.17} \]

\[
\text{CONSTANT VTOT} = 400. \quad \! \text{VTOT = VELOCITY, FT/SEC}
\]

\!
\! use turbl = 1750. for Lu, Lv, Lw
\! use tau for tauu, tauv, tauvw
\!
\text{TAU} = \text{TURBL}/\text{VTOT} \quad ! \text{eq.3,4,6}
\text{TAUQ} = 4.0 \cdot \text{BWING} / (\pi \cdot \text{VTOT}) \quad ! \text{eq.13}
\text{TAUR} = 3.0 \cdot \text{BWING} / (\pi \cdot \text{VTOT}) \quad ! \text{eq.16}

\!
\text{LP} = \text{SQRT}(\text{TURBL} \cdot \text{BWING}) / 2.6 \quad ! \text{eq.10}
\text{TAUP} = \text{LP}/\text{VTOT} \quad ! \text{eq.13}

\!
\text{TURBOMEGA} = \text{VTOT}/\text{TURBL} \quad ! \text{same as 1./TAU, 1./TAUV, 1./TAUW}
\text{WP} = \text{VTOT}/\text{LP} \quad ! \text{same as 1./TAUP}

\!
\! \text{CONSTANTS FOR CALCULATIONS OF CONTINUOUS W AND V} \quad ! \text{eq. 1,2}
\!
\text{TURBK2U} = \text{TURBSIG} \cdot \text{SQRT}(2 \cdot \text{TURBOMEGA} / \text{TSAMP})
\text{K2P} = \text{SIGP} \cdot \text{SQRT}(2 \cdot \text{WP} / \text{TSAMP})
\text{TURBK1VW} = 2 \cdot \text{TURBOMEGA}
\text{TURBK2VW} = \text{TURBOMEGA} \cdot \text{TURBOMEGA}
\text{TURBK3VW} = \text{TAU} \cdot \text{SQRT}(3.)
\text{TURBK4VW} = \text{TURBSIG} \cdot \text{SQRT}(\text{TURBOMEGA} \cdot 3 / \text{TSAMP})

\!
\! \text{CONSTANTS FOR CALCULATIONS of TURBULENCE VIA TUSTIN TRANSFORM}
\!
\text{TWOPIOVTNU} = \text{SQRT}(2 \cdot \pi / \text{TSAMP})

\!
\! \text{U-COMPONENT}
\!
\! \text{V-COMPONENT}

\!
\! \text{W-COMPONENT}

\!
\! \text{constants used in eq.18}
\!
\text{KFIL} = \text{TURBSIG} \cdot \text{SQRT}(1 / (\pi \cdot \text{TURBOMEGA}))
\text{CFIL} = \text{TURBOMEGA} / \tan(\text{TURBOMEGA} \cdot \text{TSAMP} / 2.) \quad ! \text{eq.19}
\text{FILK1} = (\text{TURBOMEGA} - \text{CFIL}) / (\text{TURBOMEGA} + \text{CFIL})
\text{FILK2} = \text{KFIL} / (1. + (\text{CFIL} / \text{TURBOMEGA}))
\text{FILK3} = \text{SQRT}(\pi \cdot (\text{TURBOMEGA} + \text{CFIL}) / \text{SDNOIS})

\!
\! \text{V-COMPONENT and W-COMPONENT}
\!
\! \text{constants used in eq.20}
\!
\text{KVW} = \text{SQRT}(3 \cdot \text{TURBOMEGA} / (2 \cdot \pi))
\text{WNPC} = \text{TURBOMEGA} + \text{CFIL}
\text{FILK4} = 2 \cdot \text{(TURBOMEGA} \cdot \text{TURBOMEGA} - \text{CFIL} \cdot \text{CFIL}) / (\text{WNPC} \cdot \text{WNPC})
FILK5 = (TURBOMEGA - CFIL)*(TURBOMEGA - CFIL)/(WNPC*WNPC)
FILK6 = CFIL + TURBOMEGA/SQRT(3.)
FILK7 = 2.*TURBOMEGA/SQRT(3.)
FILK8 = TURBOMEGA/SQRT(3.) - CFIL
FILK9 = KVW*TURBSIG / (WNPC*WNPC)

! used in eq.90
!
KFILDI = TURBSIG*SQRT(2.*TAU/TSAMP)
!
! potential alternate (filwd2, filvd2)
!
KWD1 = (1. - TAU*CFIL)/(1. + TAU*CFIL)
KWD2 = TURBSIG*SQRT(TAU/TSAMP)*CFIL/(1.+TAU*CFIL)**2
CTWSR3 = SQRT(3.)*TAU*CFIL

! P-COMPONENT

! used in eq.21
!
PFIL = SIGP* SQRT(2./ (TSAMP * WP))
PCFIL = WP / TAN( WP *TSAMP/2.) ! eq.22
PK1 = (WP - PCFIL) / (WP + PCFIL)
PK2 = PFIL / (1. + (PCFIL/WP))

! Q-COMPONENT and R-COMPONENT

! used in eq.23
!
CFILQ = (1. / TAUQ) / TAN(TSAMP / (2. * TAUQ)) ! eq. 24
QR1 = (1. - CFILQ * TAUQ) / (1. + CFILQ * TAUQ)
QR2 = (CFILQ / VTOT) / (1. + CFILQ * TAUQ)
!
! used in eq.25
!
CFILR = (1. / TAUQ) / TAN(TSAMP / (2. * TAUQ)) ! eq. 26
RK1 = (1. - CFILR * TAUQ) / (1. + CFILR * TAUQ)
RK2 = (CFILR / VTOT) / (1. + CFILR * TAUQ)

! constants for MIL STD calcs

8-28-97
!
! used in eqs. 30 - 35
!
tovtau = tsamp/tau ! (for u,w,v)
tovtau2= 2.* tovtau
tovtau4= 4.* tovtau
tovtaup = tsamp/taup ! (for p)
tovtaup2= 2.* tovtaup
tovtauq = tsamp/tauq ! (for q)
tovtaur = tsamp/taur   !(for q)
milk1 = (1. - tovtau)   !(for u,w,v)
milk2 = (1. - tovtau2)  
milk3 = sqrt(tovtau2)  
milk4 = sqrt(tovtau4)  
milk5 = (1. - tovtaup)  !(for p)
milk6 = sqrt(tovtaup2)   
milk7 = (1. - tovtaug)  !(for q)
milk8 = (1. - tovtaur)  !(for r)
piov4b=pi/(4.*bwing)  
piov3b=pi/(3.*bwing)

END OF INITIAL!

CONTINUOUS TURBULENCE MODEL    WTB, JCY 2-24-94 through 11-30-97

U - COMPONENT

TURBUD = - TURSOMEGA * TURBU + TURBK2U * FILNU
TURBU = INTVC (TURBUD, TURBU0)

V - COMPONENT

TURBXVDD = - TURBKIVW * TURBXVD - TURBK2VW * TURBXV &
           + TURBK4VW * FILNV
TURBXVD = INTVC (TURBXVDD, TURBXVD0)
TURBXVDI = TURBXVD
TURBXV = INTVC (TURBXVDI, TURBXV0)
TURBV = TURBXV + TURBK3VW * TURBXVDI
TURBV0 = TURBXV0 + TURBK3VW * TURBXVDD

W - COMPONENT

TURBXWDD = - TURBK1VW * TURBXWD - TURBK2VW * TURBXW &
           + TURBK4VW * FILNW
TURBXWD = INTVC (TURBXWDD, TURBXWDO)
TURBXWDI = TURBXWD
TURBXW = INTVC (TURBXWDI, TURBXW0)
TURBW = TURBXW + TURBK3VW * TURBXWDI
TURBWD = TURBXWDI + TURBK3VW * TURBXWDD

!***************************************************************************
! ALPHA, BETA COMPONENTS ADDED 4-28-97, 5-6-97
!***************************************************************************
!
TURBALP = TURBW/VTOT  ! ALPHA COMPONENT
TURBBET = TURBV/VTOT  ! BETA COMPONENT
TURBALPD = TURBWD/VTOT  ! ALPHA COMPONENT
TURBBETD = TURBVD/VTOT  ! BETA COMPONENT

!***************************************************************************
! P, Q, R COMPONENTS ADDED 5-15-95 JCY
!***************************************************************************
!
TURBPD = - WP * TURBP + K2P * FILNP
TURBP = INTVC (TURBPD, TURBP0)

!***************************************************************************
! Q - COMPONENT
!***************************************************************************
!
used in eq.7
!
TURBQD = TURBQ / TAUQ + TURBWD / (TAUQ * VTOT)
TURBQ = INTVC (TURBQD, TURBQ0)

!***************************************************************************
! R - COMPONENT
!***************************************************************************
!
used in eq.12
!
TURBRD = TURBR / TAUUR + TURBVD / (TAUUR * VTOT)
TURBR = INTVC (TURBRD, TURBR0)

!***************************************************************************
! OF DERIVATIVE!
!***************************************************************************
!
END !OF DERIVATIVE!

***************************************************************************
! DISCRETE BLOCK
***************************************************************************
!
DISCRETE GUST
***************************************************************************
!
INTERVAL TS_GAUS = 0.0125

************************************************************************

IF(SDNOIS.LE.EPSLON) GOTO LETA1
   FILNU = GAUSS(0.,SDNOIS)
   FILNV = GAUSS(0.,SDNOIS)
   FILNW = GAUSS(0.,SDNOIS)
   FILNP = GAUSS(0.,SDNOIS)
   GOTO LETA2

LETA1..CONTINUE
   FILNU = 0.
   FILNV = 0.
   FILNW = 0.
   FILNP = 0.

LETA2..CONTINUE

************************************************************************

U-COMPONENT

UFILK = FILNU
   IF (T .EQ. 0.) THEN
      UFILKM1 = 0.
      GUFILKM1 = 0.
      FILUDKM1 = 0.
      FILUD2KM1 = 0.
   ENDIF

GUFILK = -FILKI*GUFILKMI + TWOPIOVTNU*FILK2*(UFILK + UFILKMI)

FILUD = (GUFILK - GUFILKMI)/TSAMP

FILUD2 = - KWD1 * filud2km1
   & (kfiled1*CFIL/(1. + tau*CFIL))
   & *(ufilk - ufilkm1)  5-9-97

FILUDKM1 = FILUD
   FILUD2KM1 = FILUD2
   UFILKM1 = UFILK
   GUFILKM1 = GUFILK
   FILU = GUFILK
V-COMPONENT

VFILK = FILNV
IF (T .EQ. 0.) THEN
VFILKM1 = 0.
VFILKM2 = 0.
GVFILKM1 = 0.
GVFILKM2 = 0.
FILVDKM1 = 0.
FILVDKM2 = 0.
ENDIF
FILVKM1 = GVFILKM1

eq. 20

GVFILK = -FILK4*GVFILKM1 - FILK5*GVFILKM2 &
+ TWOPIOVTNU*FILK9*(FILK6*VFILK + FILK7*VFILKM1 &
+ FILK8*VFILKM2)

! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF V-COMPONENT

eq. 27

FILVD = (GVFILK - GVFILKM1)/TSAMP

eq. 29

FILVD2 = -2.*KWD1*FILVDKM1 &
- KWD1**2*FILVDKM2 &
+ kwd2*((1.+CTWSR3)*VFILK &
- (2.*CTWSR3)*VFILKM1 -(1.-CTWSR3)*VFILKM2)

19-12-97

GVFILKM2 = GVFILKM1
GVFILKM1 = GVFILK
VFILKM2 = VFILKM1
VFILKM1 = VFILK
FILVDKM2 = FILVDKM1
FILVDKM1 = FILVD
FILV = GVFILK

W-COMPONENT

WFILK = FILNW
IF (T .EQ. 0.) THEN
WFILKM1 = 0.
WFILKM2 = 0.
GWFILKM1 = 0.
GWFILKM2 = 0.
FILWDKMK1 = 0.
FILWDKMK2 = 0.
FILWD2KM2 = 0.
ENDIF
FILWKMI = GWFILKMI
!
! eq.20
!
GWFILK = - FILK4*GWFILKMI - FILK5*GWFILKM2 & + TWOPIOVTNU*FILK9*(FILK6*WFILK + FILK7*WFILKMI & + FILK8*WFILKM2)

! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF W-COMPONENT
!********************************************************************
!
FILWD = (GWFILK - GWFILKMI)/TSAMP
!
! eq.29
!
FILWD2 = -2.*KWDI*FILWD2KMI & - KWDI**2*FILWD2KM2 & + kwd2*((I.+CTWSR3)*WFILK & (2.*CTWSR3)*WFILKMI -(I.-CTWSR3)*WFILKM2) !5-12-97
!
GWFILKM2 = GWFILKMI
GWFILKMI = GWFILK
WFILKM2 = WFILKMI
WFILKMI = WFILK
FILWD2KM2 = FILWD2KMI
FILWD2KMI = FILWD2
FILWDKMI = FILWD
FILW = GWFILK

! P, Q, R COMPONENT ADDED 5-15-95 JCY
!********************************************************************
!
PFILK = FILNP
IF (T .EQ. 0.) THEN
PFILKMI = 0.
GPFILKMI = 0.
ENDIF
GPFILK = -PK1*GPFILKMI + PK2*(PFILK + PFILKMI) ! eq. 21
PFILKMI = PFILK ! 5-2-97
GPFILKMI = GPFILK ! 5-2-97
FILP = GPFILK

!********************************************************************
!
Q-COMPONENT
!********************************************************************
!
IF (T .EQ. 0.) THEN
FILQKMI = 0.
ENDIF
FILQ = -QK1*FILQKMI + QK2*(FILW - FILWKMI) ! eq. 23
FILQKMI = FILQ

!*************************************************************************
! R-COMPONENT
!*************************************************************************
IF (T .EQ. 0.) THEN
    FILRKM1 = 0.
ENDIF
FILR = -RK1*FILRKM1 + RK2*(FILV - FILVKMI) ! eq. 25
FILRKM1 = FILR

!*************************************************************************
! ALPHA , BETA COMPONENTS ADDED 5-7-97
!*************************************************************************

FILA = FILW/VTOT ! ALPHA COMPONENT
FILB = FILV/VTOT ! BETA COMPONENT
FILAD = FILWD/VTOT
FILBD = FILVD/VTOT

!*************************************************************************
! ALPHA , BETA COMPONENTS ALTERNATES ADDED 6-19-97
!*************************************************************************

FILAD2 = FILWD2/VTOT
FILBD2 = FILVD2/VTOT

!*************************************************************************
! MIL STD IMPLEMENTATION OF CALCS OF u,v,w,p,q,r-COMPONENT
!*************************************************************************
IF (T .EQ. 0.) THEN
    MILUKMI = 0.
    MILVKMI = 0.
    MILWKMI = 0.
    MILPKMI = 0.
    MILQKMI = 0.
    MILRKMI = 0.
ENDIF
MILUK = milkI*MILUKMI + sigu*milk3*ufilk ! eq 30
MILVK = milk2*MILVKMI + sigv*milk4*vfilk ! eq 31
MILWK = milk2*MILWKMI + sigw*milk4*wfilk ! eq 32
MILPK = milk5*MILPKMI + sigp*milk6*pfilk ! eq.33
MILQK = milk7*MILQKMI + piov4b*(milwk-milwkml) ! eq.34
MILRK = milk8*MILRKMI + piov3b*(milvk-milvkml) ! eq.35
!
Calc derivatives
!
MILUDK = (MILUK - MILUKMI)/TSAMP
MILVDK = (MILVK - MILVKMI)/TSAMP
MILWDK = (MILWK - MILWKMI)/TSAMP
!
! save past values
!
MILUKMI = MILUK
MILVKMI = MILVK
MILWKMI = MILWK
MILPKMI = MILPK
MILQKMI = MILQK
MILRKMI = MILRK

END ! GUST DISCRETE

DISCRETE DISCRMS

INTERVAL TRMS = 0.0125
SAMPS = SAMPS + 1.

DRMS(DRMSTURBU, DMNTURBU, TURBU, SAMPS)
DRMS(DRMSTURBV, DMNTURBV, TURBV, SAMPS)
DRMS(DRMSTURBW, DMNTURBW, TURBW, SAMPS)

DRMS(DRMSFILNU, DMNFILNU, FILNU, SAMPS)
DRMS(DRMSFILNV, DMNFILNV, FILNV, SAMPS)
DRMS(DRMSFILNW, DMNFILNW, FILNW, SAMPS)

DRMS(DRMSFILNP, DMNFILNP, FILNP, SAMPS)

DRMS(DRMSFILU, DMNFILU, FILU, SAMPS)
DRMS(DRMSFILV, DMNFILV, FILV, SAMPS)
DRMS(DRMSFILW, DMNFILW, FILW, SAMPS)

DRMS(DRMSTURBUD, DMNTURBUD, TURBUD, SAMPS)
DRMS(DRMSTURBVD, DMNTURBVD, TURBVD, SAMPS)
DRMS(DRMSTURBW, DMNTURBW, TURBW, SAMPS)

DRMS(DRMSTURBP, DMNTURBP, TURBP, SAMPS)
DRMS(DRMSTURBQ, DMNTURBQ, TURBQ, SAMPS)
DRMS(DRMSTURBR, DMNTURBR, TURBR, SAMPS)

DRMS(DRMSFILUD, DMNFILUD, FILUD, SAMPS)
DRMS(DRMSFILVD, DMNFILVD, FILVD, SAMPS)
DRMS(DRMSFILWD, DMNFILWD, FILWD, SAMPS)

DRMS(DRMSFILP, DMNFILP, FILP, SAMPS)
DRMS(DRMSFILQ, DMNFILQ, FILQ, SAMPS)
DRMS(DRMSFILR, DMNFILR, FILR, SAMPS)

FOLLOWING ADDED 5-8-97
DRMS(DRMSTURALP, DMNTURBALP, TURBALP, SAMPS)
DRMS(DRMSTURBBET, DMNTURBBET, TURBBET, SAMPS)
DRMS(DRMSTURALPD, DMNTURBALPD, TURBALPD, SAMPS)
DRMS(DRMSTURBBETD, DMNTURBBETD, TURBBETD, SAMPS)
DRMS(DRMSFILA, DMNFILA, FILA, SAMPS)
DRMS(DRMSFILB, DMNFILB, FILB, SAMPS)
DRMS(DRMSFILAD, DMNFILAD, FILAD, SAMPS)
DRMS(DRMSFILBD, DMNFILBD, FILBD, SAMPS)
FILAD2, FILBD2 ADDED 6-29-97
DRMS(DRMSFILAD2, DMNFILAD2, FILAD2, SAMPS)
DRMS(DRMSFILBD2, DMNFILBD2, FILBD2, SAMPS)
DRMS(DRMSTURBQ2, DMNTURBQ2, TURBQ2, SAMPS)
DRMS(DRMSTURBR2, DMNTURBR2, TURBR2, SAMPS)
DRMS(DRMSFILUD2, DMNFILUD2, FILUD2, SAMPS)
DRMS(DRMSFILVD2, DMNFILVD2, FILVD2, SAMPS)
DRMS(DRMSFILWD2, DMNFILWD2, FILWD2, SAMPS)
XCORR(XCORTW, MXMTW, TURBW, TURBW, SAMPS, 1)
XCORR(XCORFW, MXMFW, FILW, FILW, SAMPS, 1)
XCORR(XCORTV, MXMTV, TURBV, TURBV, SAMPS, 1)
XCORR(XCORFV, MXMFV, FILV, FILV, SAMPS, 1)
XCORR(XCORTQW, MXMTQW, TURBQ, TURBW, SAMPS, 0)
XCORR(XCORFQW, MXMFQW, FILQ, FILW, SAMPS, 0)
XCORR(XCORTQV, MXMTQV, TURBQ, TURBV, SAMPS, 0)
XCORR(XCORFRV, MXMFRV, FILR, FILV, SAMPS, 0)
MIL STD calcs, u,v,w,p,q,r ADDED 8-28-97
DRMS(DRMSMILUK, DMNMILUK, MILUK, SAMPS)
DRMS(DRMSMILVK, DMNMILVK, MILVK, SAMPS)
DRMS(DRMSMILWK, DMNMILWK, MILWK, SAMPS)
DRMS(DRMSMILPK, DMNMILPK, MILPK, SAMPS)
DRMS(DRMSMILQK, DMNMILQK, MILQK, SAMPS)
DRMS(DRMSMILRK, DMNMILRK, MILRK, SAMPS)
DRMS(DRMSMILUDK, DMNMILUDK, MILUDK, SAMPS)
DRMS(DRMSMILVDK, DMNMILVDK, MILVDK, SAMPS)
DRMS(DRMSMILWDK, DMNMILWDK, MILWDK, SAMPS)
END ! of DISCRETE

**************************************************************************
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - DISCRETE TO SAVE DATA TO AN ARRAY IN DRYDEN GETDATA FORMAT - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

23
DISCRETE ASC

INTERVAL TSASC = 0.0125

CALL ASCFM(T, &
    TURBU, TURBV, TURBW, FILU, &
    FILV, FILW, TURBU, TURBV, &
    TURBW, FILUD, FILVD, FILWD, &
    FILNU, FILNV, FILNW, FILNP, &
    TURBP, TURBQ, TURBR, &
    FILP, FILQ, FILR, TURBALP, &
    TURBBET, TURBALPD, TURBBETD, FILA, &
    FILB, FILAD, FILBD, FILB2, &
    TURBR2, FILUD2, FILVD2, FILWD2, &
    FILAD2, FILBD2, MILUK, MILVK, &
    MILWK, MILPK, MILQK, MILRK, &
    MILUDK, MILVDK, MILWDK, &
    NUMOUT, TSTP)

TERMT(T.GE.TSTP)

END !------------------- ASC DISCRETE

END !OF DYNAMIC!

TERMINAL

CALL WS STATS (T, &
    DRMSTURBU, DRMSTURBV, DRMSTURBW, &
    DRMSTURBP, DRMSTURBQ, DRMSTURBR, &
    DRMSTURBQ2, DRMSTURBR2, &
    DRMSFILU, DRMSFILV, DRMSFILW, &
    DRMSFILP, DRMSFILQ, DRMSFILR, &
    DRMSFILNU, DRMSFILNV, DRMSFILNW, DRMSFILNP, &
    DRMSTURBUD, DRMSTURBVD, DRMSTURBWD, &
    DRMSFILUD, DRMSFILVD, DRMSFILWD, &
    DRMSFILUD2, DRMSFILVD2, DRMSFILWD2, &
    DRMSTURALP, DRMSTURBBET, &
    DRMSTURALPD, DRMSTURBBETD, &
    DRMSFILA, DRMSFILB, DRMSFILAD, DRMSFILBD, &
    DRMSFILAD2, DRMSFILBD2, &
    DMNTURBU, DMNTURBV, DMNTURBW, &
    DMNTURBP, DMNTURBQ, DMNTURBR, &
    DMNTURBQ2, DMNTURBR2, &
    DMNFILU, DMNFILV, DMNFILW, &
    DMNFILP, DMNFILQ, DMNFILR, &
    DMNFILNU, DMNFILNV, DMNFILNW, DMNFILNP, &
    DMNTURBUD, DMNTURBVD, DMNTURBWD, &
    DMNFILUD, DMNFILVD, DMNFILWD, &
    DMNFILUD2, DMNFILVD2, DMNFILWD2, &
    DMNTURALP, DMNTURBBET, &
    DMNTURALPD, DMNTURBBETD, &
    DMNFILA, DMNFILB, DMNFILAD, DMNFILBD, &
SUBROUTINE WSTATS (T,

C ******** 35 +2 = 37 rms  ***************************************
C
* DRMSTURBU, DRMSTURBV, DRMSTURBW,
* DRMSTURBP, DRMSTURBQ, DRMSTURBR,
* DRMSTURBQ2, DRMSTURBR2,
* DRMSFILU, DRMSFILV, DRMSFILW,
* DRMSFILP, DRMSFILQ, DRMSFILR,
* DRMSFILNU, DRMSFILNV, DRMSFILNW, DRMSFILNP,
* DRMSTURBU, DRMSTURBV, DRMSTURBW,
* DRMSTURBP, DRMSTURBQ, DRMSTURBR,
* DRMSTURBQ2, DRMSTURBR2,
* DMNFILU, DMNFILV, DMNFILW,
* DMNFILP, DMNFILQ, DMNFILR,
* DMNFILNU, DMNFILNV, DMNFILNW, DMNFILNP,
* DMNTURBU, DMNTURBV, DMNTURBW,
* DMNTURBP, DMNTURBQ, DMNTURBR,
* DMNTURBQ2, DMNTURBR2,
* DMNFILA, DMNFILB, DMNFILAD, DMNFILBD,
* DMNFILAD2, DMNFILBD2,

C ********* 35 + 2 = 37 means + 12 mils= 49 values

********************************************************************
C
* DMNFILA, DMNFILB, DMNFILAD, DMNFILBD,
* DMNFILAD2, DMNFILBD2,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
* DRMSMILWK, DMNMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DMNMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILUK, DMNMILUK, DRMSMILVK, DMNMILVK,
C* WRITES AN OUTPUT FILE IN THE GETDATA FORMAT ASC. TO IMPORT TO EXCEL OR KG

C******************************************************************************************************************
     DIMENSION XLAB2(93)
     CHARACTER*16 XLAB1, XLAB2
C
     DATA XLAB1 '/names'/
     DATA NSIMCH2 / 93 /
C******************************************************************************************************************
C INCLUDE 'dpguststats.inc'
C
DATA XLAB2/ 'names',
C
C ******** 35 rms ************************************************************
C
   * 'DRMSTURBU', 'DRMSTURBV', 'DRMSTURBW',
   * 'DRMSTURBP', 'DRMSTURBQ', 'DRMSTURBR',
   * 'DMNTURBU', 'DMNTURBV', 'DMNTURBW',
   * 'DMNTURBP', 'DMNTURBQ', 'DMNTURBR',
   * 'DRMSFILU', 'DRMSFILV', 'DRMSFILW',
   * 'DRMSTURBUD', 'DRMSTURBUD', 'DRMSTURBUD',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
   * 'DMNFILU', 'DMNFILV', 'DMNFILW',
C
26
C OPEN A DATA CHANNEL FOR WRITING THE GETDATA ASC FORMAT FILE.
C
OPEN (33, FILE='KGACSL.ASC', STATUS='UNKNOWN')
C
WRITE OUT DATA IN TO FILE FOR USE IN GETDATA.
C
WRITE(33, ' (format asc 2 .i' ',/,' 'nChans',t14,i3)') NSIMCH2
WRITE(33, '(6a13)') XLAB1, (XLAB2(I), I=2, NSIMCH2)
WRITE(33, ' (6G13.7) ') T,
* DRMSTURBU, DRMSTURBV, DRMSTURBW,
* DRMSTURBP, DRMSTURBQ, DRMSTURBR,
* DRMSTURBQ2, DRMSTURBR2,
* DRNSFILU, DRNSFILV, DRMSFILW,
* DRNSFILP, DRNSFILQ, DRMSFILR,
* DRNSFILNU, DRNSFILNQ, DRMSFILN, DRMSFILNP,
* DRMSTURBUD, DRMSTURBVD, DRMSTURBWD,
* DRMSTURLD, DRMSTURLQD, DRMSTURLD,
* DRMSTURLUD2, DRMSTURLVD2, DRMSTURLD2,
* DRMSTURLP, DRMSTURBBET,
* DRMSTURALP, DRMSTURBBETD,
* DRMSTURALPDD, DRMSTURBBETDD,
* DRMSFIL, DRMSFILB, DRMSFILAD, DRMSFILBD,
c ******* 37 means + 2 rms + 12 mils ****************************
* DMNTURBU, DMNTURBV, DMNTURBW,
* DMNTURBP, DMNTURBQ, DMNTURBR,
* DMNTURBQ2, DMNTURBR2,
* DMNFILU, DMNFILV, DMNFILW,
* DMNFILP, DMNFILQ, DMNFILR,
* DMNFILNU, DMNFILNQ, DMNFILN, DMNFILNP,
* DMNTURBUD, DMNTURBVD, DMNTURBWD,
* DMNTURLD, DMNTURLQD, DMNTURLD,
* DMNTURLUD2, DMNTURLVD2, DMNTURLD2,
* DMNTURLP, DMNTURBBET,
* DMNTURLPDD, DMNTURBBETDD,
* DMNFILA, DMNFILB, DMNFILAD, DMNFILBD,
* DRMSFILAD2, DRMSFILBD2, DMNFILAD2, DMNFILBD2,
* DRMSMILUK, DRMSMILUK, DRMSMILV, DRMSMILVK,
* DRMSMILWK, DRMSMILWK, DRMSMILPK, DMNMILPK,
* DRMSMILQK, DRMSMILQK, DRMSMILRK, DMNMILRK,
* DRMSMILDK, DRMSMILDK, DRMSMILVDK, DMNMILVDK,
* DRMSMILWDK, DRMSMILWDK
C
CLOSE(33)
C
RETURN
END

SUBROUTINE ASCFM(T,
* TURBU, TURBV, TURBW, FILU,
* FILV, FILW, TURBWD, TURBVD,
* TURBWD, FILUD, FILVD, FILWD,
* FILNU, FILNV, FILNW, FILNP,
* TURBP, TURBQ, TURBR,
* FILP, FILQ, FILR, TURBALP,
TURBBET, TURBALPD, TURBBETD, FILA,
* FILB, FILAD, FILBD, TURBQ2,
* TURBR2, FILUD2, FILVD2, FILWD2,
* FILAD2, FILBD2, MILUK, MILV VK,
* MILW K, MILPK, MILQK, MILR K,
* MILUDK, MILVDK, MILWDK,
* NUMOUT, TSTP

SUBROUTINE ASCFM

* WRITES AN OUTPUT FILE IN THE GETDATA FORMAT ASC.

IMPLICIT REAL*8 (a-h,o-z)
DIMENSION XLABEL2(47)
CHARACTER*I6 XLABEL1, XLABEL2

REAL MILUK, MILVK, MILWK, MILPK, MILQK, MILRK,
* MILUDK, MILVDK, MILWDK

DATA XLABEL1 /'names'/
DATA NSIMCH2 / 47/
DATA XLABEL2/'names',
* 'TURBU', 'TURBV', 'TURBW', 'FILU',
* 'FILV', 'FILW', 'TURBUD', 'TURBVD',
* 'TURBUD', 'FILUD', 'FILVD', 'FILWD',
* 'FILN', 'FILNV', 'FILNW', 'FILNP',
* 'TURB', 'TURBQ', 'TURBR',
* 'FILP', 'FILQ', 'FILR', 'TURBALP',
* 'TURBBET', 'TURBALPD', 'TURBBETD', 'FILA',
* 'FILB', 'FILAD', 'FILBD', TURBQ2',
* 'TURBR2', 'FILUD2', 'FILVD2', 'FILWD2',
* 'FILAD2', 'FILBD2', 'MILUK', 'MILVK',
* 'MILWK', 'MILPK', 'MILQK', 'MILR K',
* 'MILUDK', 'MILVDK', 'MILVDK' /

OPEN A DATA CHANNEL FOR WRITING THE GETDATA ASC FORMAT FILE.

IF((T .EQ. 0.0) ) THEN
OPEN (3,FILE='ACSL.ASC',STATUS='UNKNOWN')

WRITE OUT DATA IN TO FILE FOR USE IN GETDATA.

WRITE(3,('format asc 2 .1',/,'nChans',t14,i3)')NSIMCH2-1
WRITE(3,('6a13')') XLABEL1,(XLABEL2(I),I=2,NSIMCH2)
WRITE(3,('data001'''))
INTERVA = NUMOUT
ENDIF
IF(T .GE. 0.0 .AND. T .LT. TSTP) THEN
  IF (INTERVA .EQ. NUMOUT) THEN
    WRITE(3,'(6G13.7)') T,
    * TURBU,  TURBV,  TURBW,  FILU,
    * FILV,   FILW,   TURBUD,  TURBVD,
    * TURBWD, FILUD,  FILVD,   FILWD,
    * FILNU,  FILNV,  FILNW,   FILNP,
    * TURBP,  TURBQ,  TURBR,
    * FILP,   FILQ,   FILR,    TURBALP,
    * TURBBET, TURBALPD, TURBBETD, FILA,
    * FILB,   FILAD,  FILBD,   TURBQ2,
    * TURBR2, FILUD2,  FILVD2,  FILWD2,
    * FILAD2, FILBD2,  MILUK,   MILVK,
    * MILWK,  MILPK,  MILQK,   MILRK,
    * MILUDK, MILVDK,  MILWDK
  ELSE
    INTERVA = INTERVA + 1
  ENDIF
ENDIF
IF(T .GT. TSTP) CLOSE(3)
RETURN
END
**Time History Plots**

Numerous 100-second runs were made with the GUSTMDL test program to produce turbulence sequences for analysis of the continuous, Tustin, and MIL STD turbulence models. Simulated aircraft velocities of 100 ft/sec and 1000 ft/sec were used. Sample time history plots of the first 10 seconds of these sequences for each model are included in figures 1 through 3 for \( V = 100 \text{ ft/sec} \) and in figures 4 through 6 for \( V = 1000 \text{ ft/sec} \).

![Sample time histories](image)

*Figure 1. Sample time histories; \( V =100 \text{ ft/sec} \) - continuous model.*
Figure 2. Sample time histories; V = 100 ft/sec - Tustin discrete model.
Figure 3. Sample time histories; $V = 100$ ft/sec - MIL STD discrete model.
Figure 4. Sample time histories; $V = 1000$ ft/sec - continuous model.
Figure 5. Sample time histories; \( V = 1000 \) ft/sec - Tustin discrete model.
Performance of the three implemented turbulence models was evaluated by computing the power spectral densities (PSD) of results of simulated turbulence from the GUSTMDL program. These PSD's of the simulated turbulence components were then compared to the theoretical Dryden spectral model components. Equations used to compute the PSD for each model are listed in the section below, and the MATLAB m-files used to implement these equations and produce the PSD plots follow in the next section.

Equations

The two-sided theoretical power spectral densities for the continuous model were provided by the Government from the Dryden model (ref. 1) and are listed below as equations (36) through (41).
\[ S_u(\omega) = \frac{\sigma^2 u \tau u}{\pi} \frac{1}{1 + (\tau_u \omega)^2} \]  

(36)

\[ S_v(\omega) = \frac{1}{2} \bar{\Phi}_v(\omega) = \frac{\sigma^2 v \tau v}{2\pi} \frac{1 + 3(\tau_v \omega)^2}{\left(1 + (\tau_v \omega)^2\right)^2} \]  

(37)

\[ S_w(\omega) = \frac{1}{2} \bar{\Phi}_w(\omega) = \frac{\sigma^2 w \tau w}{2\pi} \frac{1 + 3(\tau_w \omega)^2}{\left(1 + (\tau_w \omega)^2\right)^2} \]  

(38)

\[ S_p(\omega) = \frac{\sigma^2 p \tau p}{\pi} \frac{1}{1 + (\tau_p \omega)^2} \]  

(39)

\[ S_q(\omega) = \frac{(\omega/V)^2}{1 + \left[\left(4b_w/\pi\right)(\omega/V)\right]^2} S_w(\omega) \]  

(40)

\[ S_r(\omega) = \frac{(\omega/V)^2}{1 + \left[\left(3b_w/\pi\right)(\omega/V)\right]^2} S_v(\omega) \]  

(41)

The theoretical power spectral densities for the Tustin model were provided by the Government and are listed below as equations (42) through (51).

\[ G_{du}(\omega) = \frac{T_v}{2\pi} \left| H_u(\mathcal{e}^{j\omega T_v}) \right|^2 = \frac{\sigma^2 u \omega_u}{\pi} \left[ 1 + \cos(\omega T_v) \right] \frac{1 + \cos(\omega T_v)}{\left(\omega^2 + C_{BL}^2\right) + \left(\omega^2 - C_{BL}^2\right) \cos(\omega T_v)} \]  

(42)

\[ G_{di}(\omega) = \frac{T_v}{2\pi} \left| H_i(\mathcal{e}^{j\omega T_v}) \right|^2 \quad \text{for } i = v, w \]

\[ = \frac{K_i^2 \sigma_i^2 \left[ a^2 + b^2 + c^2 + 2(ab + bc) \cos(\omega T_v) + 2ac \cos(2\omega T_v) \right]}{d^2 + e^2 + f^2 + 2(de + ef) \cos(\omega T_v) + 2df \cos(2\omega T_v)} \]  

(43)
\[ \omega_i = V/L_i \]

\[ K_i = \frac{3\omega_i}{2\pi} \]

\[ a = C_{BL} + \omega_i \sqrt{3} \]

\[ b = 2\omega_i \sqrt{3} \]

\[ c = \omega_i \sqrt{3} - C_{BL} \]

\[ d = (\omega_i + C_{BL})^2 \]

\[ e = 2(\omega_i^2 - C_{BL}^2) \]

\[ f = (\omega_i - C_{BL})^2 \]

\[ G_{dp}(\omega) = \frac{T_v}{2\pi} |H_p (e^{j\omega T_v})|^2 = \frac{\xi_p^2 \omega_p}{\pi} \frac{1 + \cos(\omega T_v)}{\left(\omega_p^2 + C_{BL p}^2\right) + \left(\omega_p^2 - C_{BL p}^2\right)} \cos(\omega T_v) \] (45)

\[ G_{dq}(\omega) = |H_q (e^{j\omega T_v})|^2 G_{dv}(\omega) \] (46)

where

\[ |H_q (e^{j\omega T_v})|^2 = \frac{2C_{BL q}^2}{V^2} \frac{1 - \cos(\omega T_v)}{M_{q+}^2 + M_{q-}^2 + 2M_{q+}M_{q-}\cos(\omega T_v)} \] (47)

and

\[ M_{q+} = 1 + \tau_q C_{BL q} \]

\[ M_{q-} = 1 - \tau_q C_{BL q} \] (48)

\[ G_{dr}(\omega) = |H_r (e^{j\omega T_v})|^2 G_{dv}(\omega) \] (49)

where

\[ |H_r (e^{j\omega T_v})|^2 = \frac{2C_{BL r}^2}{V^2} \frac{1 - \cos(\omega T_v)}{M_{r+}^2 + M_{r-}^2 + 2M_{r+}M_{r-}\cos(\omega T_v)} \] (50)

and

\[ M_{r+} = 1 + \tau_r C_{BL r} \]

\[ M_{r-} = 1 - \tau_r C_{BL r} \] (51)

The theoretical power spectral densities for the MIL STD model were provided by the Government and are listed below as equations (52) through (57).
where
\[ G_{di}(\omega) = \frac{T_v}{2\pi} |H_i(e^{j\omega T_v})|^2 \]
\[ = \frac{T_v^2}{\pi \tau_i} \sigma_i^2 \]
\[ = \frac{1}{1 + (1 - a_i)^2 - 2(1 - a_i)\cos(\omega T_v)} \]

for \( i = u, v, w, p \) \hfill (52)

\[ H_i(e^{j\omega T_v}) = \frac{\sigma_i \sqrt{2a_i}}{1 - (1 - a_i)e^{-j\omega T_v}} \]
\[ = \frac{\sigma_i \sqrt{2a_i}}{[1 - (1 - a_i)\cos(\omega T_v)] - j(1 - a_i)\sin(\omega T_v)} \]

for \( i = u, v, w, p \) \hfill (53)

\[ a_u = \frac{T_v}{\tau_u} \]
\[ a_v = \frac{2T_v}{\tau_v} \]
\[ a_w = \frac{2T_v}{\tau_w} \]
\[ a_p = \frac{T_v}{\tau_p} \]

\[ G_{di}(\omega) = \frac{T_v}{2\pi} |H_i(e^{j\omega T_v})|^2 \]
\[ = \frac{T_v}{2\pi i k_i^2 k_j^2 [2 - 2\cos(\omega T_v)]}{(1 + X^2 + Y^2 - 2X(1 + Y)\cos(\omega T_v) + 2Y\cos(2\omega T_v))} \]

for \( i, j = q, w \) or \( r, v \) \hfill (55)

\[ H_i(z) = \frac{\xi_i(z)}{\nu_j(z)} \]
\[ = \frac{k_i k_j (1 - z^{-1})}{1 - (2 - a_i - a_j)z^{-1} + (1 - a_i)(1 - a_j)z^{-2}} \]
\[ = \frac{k_i k_j (1 - z^{-1})}{1 - Xz^{-1} + Yz^{-2}} \]

where \hfill (56)
\begin{align*}
    a_q &= \frac{T_v}{\tau_q} \\
    a_r &= \frac{T_v}{\tau_r} \\
    k_q &= \frac{\pi}{4b_w} \\
    k_r &= \frac{\pi}{3b_w} \\
    k_v &= \sigma_v \sqrt{\frac{4T_v}{\tau_v}} \\
    k_w &= \sigma_w \sqrt{\frac{4T_v}{\tau_w}} \\
    X &= 2 - a_i - a_j \\
    Y &= (1 - a_i)(1 - a_j)
\end{align*}

for \( i, j = q, w \text{ or } r, v \) \hspace{1cm} (57)

**Code**

The PSD equations listed in the previous section were coded into MATLAB m-file `fig12dat.m`. Time history sequences from the GUSTMDL simulation were input using the gdload utility avoiding conversion of input files prior to use of the MATLAB m-files. Execution of `fig12dat.m` required assigning a value for the velocity \( V \) to correspond with the velocity used in the input file obtained from the GUSTMDL program. Using the gdwrite utility, the m-file produces output files in the asc2 format which is compatible with one of the plotting programs used by the DCB researchers. Source code for the m-file `fig12dat.m` is presented below.

```matlab
% Computes data for PSD plots
file=fig12dat.m

% ***** ***** ***** ***** ***** ***** *****
% First calc theoretical psd's
% ***** ***** ***** ***** ***** ***** *****

% Snu = sampling function transform
% Hxsq = magnitude in dB of transfer function for x-component
% Sx = spectrum of q-, r-components
% Tnu = sampling interval = 1/80
% Lu = Lw = Lv = Lvw = Dryden scale length
% sigma = sigu = sigw = sigv = std dev of turbulence

Tnu = 0.0125;
V = 100.;
bwing = 37.42;
Lu = 1750;
Lwv = 1750;
Lp = sqrt(Lwv*bwing)/2.6;
sigma = 5.;
```
sigp = 1.9/sqrt(Lw*bwing)*sigma;
Npts = 200;

f = logspace(-2,2,Npts);
w = 2.*pi*f;

tauu = Lu/V;
tauwv = Lw/V;
taup = Lp/V;
tauq = 4.*bwing/(pi*V);
taur = 3.*bwing/(pi*V);

Kwv = sigma^2*tauwv/(2*pi);
Ku = sigma^2*tauu/pi;
Kp = sigp^2*taup/pi;

% Pre-allocate vectors

Snu = zeros(size(w));
Swv = zeros(size(w));
Su = zeros(size(w));
Sp = zeros(size(w));
Hqsq = zeros(size(w));
Hrsq = zeros(size(w));
Sq = zeros(size(w));
Sr = zeros(size(w));
Gdu = zeros(size(w));
Gdwv = zeros(size(w));
Gdp = zeros(size(w));
Gdq = zeros(size(w));
Gdr = zeros(size(w));

% Calculate psd's

for i = 1:Npts,
    Snu(i) = 10.*log10((sin(w(i)*Tnu/2.)*(w(i)*Tnu/2.)^2)/(w(i)*Tnu/2.)^2);
    N = 1. + 3.*(tauwv*w(i))^2;
    D = (1 + (tauwv*w(i))^2)^2;
    Swv(i) = 10.*log10(Kwv*N/D);
    Su(i) = 10.*log10(Ku/(1. + (tauu*w(i))^2));
end
Sp(i) = 10.*log10(Kp/(1. + (taup*w(i))^2));

Hqsq(i) = 10.*log10((w(i)/V)^2/(1 + (tauq*w(i))^2));

Hrsq(i) = 10.*log10((w(i)/V)^2/(1 + (taur*w(i))^2));

Sq(i) = Hqsq(i) + Swv(i);

Sr(i) = Hrsq(i) + Swv(i);

end

% calc discrete theoretical psd's
% u-component constants

wud = i./tauu;
cfil = wud/tan(wud*Tnu/2.);
sdnois = i.;
kud = sigma*sqrt(tauu/pi);
cud = sdnois*sqrt(pi*(wud + cfil));
ufacl = kud^2*wud^2;
ucl = wud^2+cfil^2;
u2 = wud^2-cfil^2;

% Mil constants - u
au = Tnu/tauu;
onemau = 1. - au;
onemausq = onemau^2;
Thusq = Tnu^2;
sigusq=sigma^2;
milunum= Thusq/(pi*tauu)*sigusq;

% v and w-component constants

wvdw = 1./tauwv;

kwvd = sqrt(3./(2.*tauwv*pi));

wosr3 = wvdw/sqrt(3.);
a = cfil + wosr3;
b = 2.*wosr3;
c = wosr3- cfil;
wnc = wvdw+cfil;
d = wnc^2;
e = 2.)*(wvdw^2-cfil^2);
f = (wvdw-cfil)^2;
wvfacl = kwvd^2*sigma^2;
abcsun = a^2+b^2+c^2;
defsum = d^2+e^2+f^2;

Mil constants - w, v
awv = 2.*Tnu/tauwv;
onemawv = 1. - awv;
onemawvsq = onemawv^2;
milwvnum= Tnusq/(pi*tauwv)*sigusq;

milwvnum= Tnusq/(pi*tauwv)*sigusq;

% p-component constants
wpd = 1./taup;
pcfil = wpd/tan(wpd*Tnu/2.);
mapt = (1. + pcfil*taup);
mmpt = (1. - pcfil*taup);
kpd = sigp*sqrt(taup/pi);

milpnum = Tnusq/(pi*taup)*sigpsq;

% q-component
wqd = 1./tauq;
cfilq = wqd/tan(Tnu/(2.*tauq));
maqt = (1. + cfilq*tauq);
mmtq = (1. - cfilq*tauq);
kqd = cfilq/V;

milqwkl = i. + Xqw^2 + Yqw^2;
milqwkl2 = 2.*Xqw*(1. + Yqw);
milqwkl3 = 2.*Yqw;
kq = pi/(4.*bwing);
kq = replies/sqrt(2.*awv);
Thuov2pi = Thu/(2.*pi);
kqsq=kq^2;
kwsq=kw^2;
milqwnkl = Tnuov2pi*kqsq*kwsq;
% milqwnkl = Tnuov2pi^3*kqsq*kwsq;
%
% * * * * * * * * * * * * * * * * * * * * *
% r-component constants
% * * * * * * * * * * * * * * * * * * * * *

wrd = 1./taur;
cfilr = wrd/tan(Tnu/(2.*taur));
mart = (1. + cfilr*taur);
mmrt = (1. - cfilr*taur);
krd = cfilr/V;
%
% Mil constants - r
ar = Tnu/taur;
onemar = 1. - ar;
% onemawv, awv defined above
Xrv = 2. - ar - awv;
Yrv = onemar*onemawv;
milrvk4 = 1. + Xrv^2 + Yrv^2;
milrvk5 = 2.*Xrv*(1. + Yrv);
milrvk6 = 2.* Yrv;
kr = pi/(3.*bwing);
% kw defined above
kv = kw;
% Tnuov2pi defined above
krsq=kr^2;
kvsq=kv^2;
milrvnkl = Tnuov2pi*krsq*kvsq;
%
% Calculate psd's
%
% u-component
%
for i = 1:Npts,

    coswt = cos(w(i)*Tnu);
    cos2wt = cos(2*w(i)*Tnu);
    UN = ufacl*(1. + coswt);
    UD = (udl + (ud2*coswt));

    Gdu(i) = 10.*log10(UN/UD);
%
% Mil specs - u
%
    milud = 1. + onemausq - 2.*onemau*coswt;
    Gmilu(i) = 10.*log10(milunum/milud);
%
% v and w-component
%

wvfac2 = 2.*(a*b + b*c)*coswt;
wvfac3 = 2.*a*c*cos2wt;
wvfac4 = 2.*(d*e + e*f)*coswt;
wvfac5 = 2.*d*f*cos2wt;

WVN = wvfac1*(abcsum + wvfac2+ wvfac3);
WVD = defsum + wvfac4 + wvfac5;

Gdwv(i) = 10.*log10(WVN/WVD);

Mil specs - w,v

milwvd = 1. + onemawv^2 - 2.*onemawv*coswt;
Gmilwv(i) = 10.*log10(milwvnum/milwvd);

p-component

PN = kpd^2*2.*(1. + coswt);
PD = maqt^2 + mmqt^2 + 2.*maqt*mmqt*(coswt);

Gdp(i) = 10.*log10(PN/PD);

Mil specs - p

milpd = 1. + onemap^2 - 2.*onemap*coswt;
Gmilp(i) = 10.*log10(milpnum/milpd);

q-component

QN = kqd^2*2.*(1. - coswt);
QD = maqt^2 + mmqt^2 + 2.*maqt*mmqt*(coswt);

Gdq(i) = 10.*log10(QN/QD) + Gdwv(i);

Mil specs - q

milqwnk2 = (2. - 2.*coswt);
milqwnum = milqwnk1*milqwnk2;
milqwd = milqwk1 - milqwk2*coswt + milqwk3*cos2wt;
Gmilq(i) = 10.*log10(milqwnum/milqwd);

r-component

Mil specs - r
RN = krd^2*2.*(1. - coswt);
RD = mart^2 + mmrt^2 + 2.*mart*mmrt*(coswt);

Gdr(i) = 10.*log10(RN/RD) + Gdv(i);
%
%
% Mil specs - r
%
%
% milqwnk2 defined above
milrvnk2 = milqwnk2;
milrvnum = milrvnk1*milrvnk2;
milrvd = milrvk4 - milrvk5*coswt + milrvk6*cos2wt;
Gmilr(i) = 10.*log10(milrvnum/milrvd);
%
end
%
 *
% Now calc MEASURED psd's of run gustrxxx.asc2 data
 *
%
% First load data from run gustrxxx.asc2
%
gdload gustr32.asc2
%
% Now do continuous psd's
%
nfft = 1024;
nov = 512;
fs = 80;
win = hanning(1024);
SF = norm(win)^2/sum(win)^2;
%
[Puc, fm] = psd(turbu, nfft, fs, win, nov);
Puc = 10.*log10(Puc*SF);
%
[Pvc, fm] = psd(turbv, nfft, fs, win, nov);
Pvc = 10.*log10(Pvc*SF);
%
[Pwc, fm] = psd(turbw, nfft, fs, win, nov);
Pwc = 10.*log10(Pwc*SF);
%
[Ppc, fm] = psd(turbp, nfft, fs, win, nov);
Ppc = 10.*log10(Ppc*SF);
%
[Pqc, fm] = psd(turbq, nfft, fs, win, nov);
Pqc = 10.*log10(Pqc*SF);
%
[Prc, fm] = psd(turbr, nfft, fs, win, nov);
Prc = 10.*log10(Prc*SF);
%
% Now do discrete psd's
%
[Pud, fm] = psd(filu, nfft, fs, win, nov);
Pud = 10.*log10(Pud*SF);  
% 
[Pvd, fm] = psd(filv,nfft,fs,win,nov); 
Pvd = 10.*log10(Pvd*SF);  
% 
[Pwd, fm] = psd(filw,nfft,fs,win,nov); 
Pwd = 10.*log10(Pwd*SF);  
% 
[Ppd, fm] = psd(filp,nfft,fs,win,nov); 
Ppd = 10.*log10(Ppd*SF);  
% 
[Pqd, fm] = psd(filq,nfft,fs,win,nov); 
Pqd = 10.*log10(Pqd*SF);  
% 
[Prd, fm] = psd(filr,nfft,fs,win,nov); 
Prd = 10.*log10(Prd*SF);  
%  
% Now do discrete mil spec psd's  
% 
[Pmilud, fm] = psd(miluk,nfft,fs,wln,nov);  
Pmilud = 10.*log10(Pmilud*SF);  
% 
[Pmilvd, fm] = psd(milvk,nfft,fs,wln,nov);  
Pmilvd = 10.*log10(Pmilvd*SF);  
% 
[Pmilwd, fm] = psd(milwk,nfft,fs,wln,nov);  
Pmilwd = 10.*log10(Pmilwd*SF);  
% 
[Pmilpd, fm] = psd(milpk,nfft,fs,wln,nov);  
Pmilpd = 10.*log10(Pmilpd*SF);  
% 
[Pmilqd, fm] = psd(milqk,nfft,fs,wln,nov);  
Pmilqd = 10.*log10(Pmilqd*SF);  
%  
[Pmilrd, fm] = psd(milrk,nfft,fs,wln,nov);  
Pmilrd = 10.*log10(Pmilrd*SF);  
%  
% * * * * * * * * * * * * * * * * * * * * *  
% Write results  
% * * * * * * * * * * * * * * * * * * * * *  
% 
outvect = ['ft Snu Su Swv Hqsq Hrsq Sp Sq Sr Gdu Gdwv Gdp Gdq Gdr Gmilw Gmilp Gmilq Gmilr'];  
gdwrite('figl2datj.asc2 asc2', outvect)  
gdwrite('figl2datmj.asc2 asc2', 'fm Puc,Pvc,Pwc Ppc,Pqc,Prc,Pud,Pvd,Pwd,Ppd,Pqd,Prd,Pmilud,Pmilvd, Pmilwd,Pmilpd,Pmilqd,Pmilrd')  

Plots

PSD's of sample sequences of each turbulence component were produced using figl2dat.m for each of the three models. These data were input into a commercial plotting program, and the resulting plots are shown below in figures 7 through 12. These plots compare the sample PSD and theoretical PSD for each of the turbulence sequences that were shown in figures 1 through 6.
Figure 7. Power spectral densities, $V = 100$ ft/sec, continuous model.
Figure 8. Power spectral densities, \( V = 100 \) \( \text{ft/sec} \), Tustin model.
Figure 9. Power spectral densities, $V = 100$ ft/sec, MIL STD model.
Figure 10. Power spectral densities, $V = 1000$ ft/sec, continuous model.
Figure 11. Power spectral densities, V = 1000 ft/sec, Tustin model.
Figure 12. Power spectral densities, $V = 1000$ ft/sec, MIL STD model.
Measured Statistics

To evaluate the statistical accuracy of the simulated turbulence components extended runs were made with the GUSTMDL program. Each run was 1000 seconds long to reduce start-up effects and reduce the variance of sample statistics. A set of ten computer runs each using a different seed for the random noise generator was made using a velocity of 100 ft/sec and the parameter values identical to those for figures 1 through 6. A second set of runs using a velocity of 1000 ft/sec was generated. Sample statistics (rms and mean values) were calculated for each component of turbulence for each of the implemented models by invoking the DRMS macro of the GUSTMDL program.

Equations

Sample statistics (mean $\bar{m}_{\xi_i}^{(j)}$ and standard deviation $\hat{\sigma}_{\xi_i}^{(j)}$) were computed for each sequence for each computer run according to

$$\bar{m}_{\xi_i}^{(j)} = \frac{1}{N} \sum_{k=1}^{N} \xi_{\xi_i}(k); \quad i = u, v, w, p, q, r; \quad j = 1, 2, \ldots, 10$$

$$\hat{\sigma}_{\xi_i}^{(j)} = \left( \frac{1}{N-1} \sum_{k=1}^{N} \left[ \xi_{\xi_i}^{(j)}(k) - \bar{m}_{\xi_i}^{(j)} \right]^2 \right)^{1/2}$$

Also computed were the mean $\hat{M}_g$ and standard deviation $\hat{\sigma}_g$ of the sample means and sample standard deviations taken over the set of ten sequences for each component as follows:

$$\hat{M}_g = \frac{1}{10} \sum_{j=1}^{10} g^{(j)}$$

$$\hat{\sigma}_g = \left( \frac{1}{10-1} \sum_{j=1}^{10} \left[ g^{(j)} - m_g \right]^2 \right)^{1/2}$$

where $g = \bar{m}_{\xi_i}$ or $\hat{\sigma}_{\xi_i}; \quad i = u, v, w, p, q, r.$

Results

Results computed using equations (58) and (59) can be found in Tables 2 through 7. Sample means and standard deviations of each turbulence component for each of the ten 1000-second runs are shown for the continuous, Tustin, and MIL STD models. Also shown in the tables are the mean and standard deviation of the sample means and sample standard deviations taken over the set of ten sequences for each component using equations (60) and (61).
Table 2. Statistics for Ten 1000-Second Simulation Runs;
Continuous Case; \( V = 100 \text{ ft/sec} \)

<table>
<thead>
<tr>
<th>Run</th>
<th>Standard deviation</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
<td>( v )</td>
<td>( w )</td>
<td>( p )</td>
<td>( q )</td>
<td>( r )</td>
</tr>
<tr>
<td>34</td>
<td>4.95</td>
<td>5.27</td>
<td>4.90</td>
<td>0.0371</td>
<td>0.0212</td>
<td>0.0242</td>
</tr>
<tr>
<td>36</td>
<td>4.80</td>
<td>5.51</td>
<td>5.83</td>
<td>0.0375</td>
<td>0.0214</td>
<td>0.0245</td>
</tr>
<tr>
<td>37</td>
<td>5.27</td>
<td>5.39</td>
<td>4.82</td>
<td>0.0373</td>
<td>0.0209</td>
<td>0.0239</td>
</tr>
<tr>
<td>38</td>
<td>4.84</td>
<td>4.20</td>
<td>5.16</td>
<td>0.0366</td>
<td>0.0209</td>
<td>0.0236</td>
</tr>
<tr>
<td>39</td>
<td>5.27</td>
<td>4.77</td>
<td>4.51</td>
<td>0.0379</td>
<td>0.0214</td>
<td>0.0243</td>
</tr>
<tr>
<td>40</td>
<td>5.24</td>
<td>5.32</td>
<td>4.76</td>
<td>0.0394</td>
<td>0.0207</td>
<td>0.0236</td>
</tr>
<tr>
<td>41</td>
<td>5.11</td>
<td>4.88</td>
<td>5.00</td>
<td>0.0366</td>
<td>0.0210</td>
<td>0.0245</td>
</tr>
<tr>
<td>42</td>
<td>5.46</td>
<td>5.11</td>
<td>4.78</td>
<td>0.0372</td>
<td>0.0209</td>
<td>0.0240</td>
</tr>
<tr>
<td>43</td>
<td>5.36</td>
<td>5.20</td>
<td>5.07</td>
<td>0.0368</td>
<td>0.0208</td>
<td>0.0236</td>
</tr>
<tr>
<td>44</td>
<td>4.40</td>
<td>5.23</td>
<td>5.24</td>
<td>0.0360</td>
<td>0.0215</td>
<td>0.0243</td>
</tr>
<tr>
<td>Mean</td>
<td>5.07</td>
<td>5.09</td>
<td>5.01</td>
<td>0.0372</td>
<td>0.0211</td>
<td>0.0241</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.321</td>
<td>0.382</td>
<td>0.358</td>
<td>0.000912</td>
<td>0.000273</td>
<td>0.000369</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>Mean</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
<td>( v )</td>
<td>( w )</td>
<td>( p )</td>
<td>( q )</td>
<td>( r )</td>
</tr>
<tr>
<td>34</td>
<td>0.296</td>
<td>-0.112</td>
<td>-0.463</td>
<td>0.000106</td>
<td>-6.42e-05</td>
<td>6.54e-05</td>
</tr>
<tr>
<td>36</td>
<td>-1.23</td>
<td>-0.951</td>
<td>-0.388</td>
<td>-0.000551</td>
<td>3.45e-06</td>
<td>2.08e-05</td>
</tr>
<tr>
<td>37</td>
<td>-0.261</td>
<td>-0.740</td>
<td>0.214</td>
<td>-0.00264</td>
<td>-2.90e-05</td>
<td>-0.000109</td>
</tr>
<tr>
<td>38</td>
<td>0.259</td>
<td>1.11</td>
<td>-0.418</td>
<td>-0.000202</td>
<td>-5.56e-05</td>
<td>5.06e-05</td>
</tr>
<tr>
<td>39</td>
<td>1.77</td>
<td>-0.499</td>
<td>-0.185</td>
<td>-0.00267</td>
<td>8.01e-06</td>
<td>-5.22e-06</td>
</tr>
<tr>
<td>40</td>
<td>1.11</td>
<td>0.849</td>
<td>0.606</td>
<td>-6.02e-05</td>
<td>2.72e-05</td>
<td>1.67e-05</td>
</tr>
<tr>
<td>41</td>
<td>-0.419</td>
<td>0.229</td>
<td>-0.351</td>
<td>0.000188</td>
<td>-7.82e-05</td>
<td>-3.97e-05</td>
</tr>
<tr>
<td>42</td>
<td>-1.09</td>
<td>0.887</td>
<td>-0.284</td>
<td>-0.000544</td>
<td>-4.67e-05</td>
<td>-2.52e-05</td>
</tr>
<tr>
<td>43</td>
<td>0.425</td>
<td>-1.26</td>
<td>0.443</td>
<td>-0.000526</td>
<td>1.81e-05</td>
<td>-6.10e-05</td>
</tr>
<tr>
<td>44</td>
<td>-0.660</td>
<td>-0.258</td>
<td>-0.923</td>
<td>-0.00361</td>
<td>-4.15e-05</td>
<td>5.71e-05</td>
</tr>
<tr>
<td>Mean</td>
<td>0.0204</td>
<td>-0.0743</td>
<td>-0.175</td>
<td>-0.00105</td>
<td>-2.58e-05</td>
<td>-2.96e-06</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.950</td>
<td>0.822</td>
<td>0.463</td>
<td>0.00138</td>
<td>3.73e-05</td>
<td>5.63e-05</td>
</tr>
</tbody>
</table>
### Table 3. Statistics for Ten 1000-Second Simulation Runs; Tustin Case; \( V = 100 \text{ ft/sec} \)

<table>
<thead>
<tr>
<th>Run</th>
<th>Standard deviation</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
<td>( v )</td>
<td>( w )</td>
</tr>
<tr>
<td>34</td>
<td>4.95</td>
<td>5.23</td>
<td>4.89</td>
</tr>
<tr>
<td>36</td>
<td>4.80</td>
<td>5.44</td>
<td>5.75</td>
</tr>
<tr>
<td>37</td>
<td>5.26</td>
<td>5.35</td>
<td>4.79</td>
</tr>
<tr>
<td>38</td>
<td>4.84</td>
<td>4.18</td>
<td>5.11</td>
</tr>
<tr>
<td>39</td>
<td>5.26</td>
<td>4.74</td>
<td>4.51</td>
</tr>
<tr>
<td>40</td>
<td>5.23</td>
<td>5.27</td>
<td>4.73</td>
</tr>
<tr>
<td>41</td>
<td>5.11</td>
<td>4.84</td>
<td>4.98</td>
</tr>
<tr>
<td>42</td>
<td>5.45</td>
<td>5.08</td>
<td>4.77</td>
</tr>
<tr>
<td>43</td>
<td>5.36</td>
<td>5.15</td>
<td>5.07</td>
</tr>
<tr>
<td>44</td>
<td>4.40</td>
<td>5.21</td>
<td>5.22</td>
</tr>
<tr>
<td>Mean</td>
<td>5.07</td>
<td>5.05</td>
<td>4.98</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.320</td>
<td>0.373</td>
<td>0.341</td>
</tr>
</tbody>
</table>
Table 4. Statistics for Ten 1000-Second Simulation Runs; MIL STD Case; $V = 100$ ft/sec

<table>
<thead>
<tr>
<th>Run</th>
<th>Standard deviation</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$u$</td>
<td>$v$</td>
</tr>
<tr>
<td>34</td>
<td>4.95</td>
<td>5.29</td>
</tr>
<tr>
<td>36</td>
<td>4.80</td>
<td>5.47</td>
</tr>
<tr>
<td>37</td>
<td>5.27</td>
<td>5.33</td>
</tr>
<tr>
<td>38</td>
<td>4.84</td>
<td>4.22</td>
</tr>
<tr>
<td>39</td>
<td>5.26</td>
<td>4.80</td>
</tr>
<tr>
<td>40</td>
<td>5.24</td>
<td>5.28</td>
</tr>
<tr>
<td>41</td>
<td>5.11</td>
<td>4.91</td>
</tr>
<tr>
<td>42</td>
<td>5.46</td>
<td>5.09</td>
</tr>
<tr>
<td>43</td>
<td>5.36</td>
<td>5.11</td>
</tr>
<tr>
<td>44</td>
<td>4.40</td>
<td>5.18</td>
</tr>
<tr>
<td>Mean</td>
<td>5.07</td>
<td>5.07</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.320</td>
<td>0.357</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>$u$</th>
<th>$v$</th>
<th>$w$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>0.295</td>
<td>-0.121</td>
<td>-0.451</td>
<td>0.000106</td>
<td>-6.13e-05</td>
<td>5.09e-05</td>
</tr>
<tr>
<td>36</td>
<td>-1.24</td>
<td>-0.956</td>
<td>-0.368</td>
<td>-0.000551</td>
<td>5.42e-06</td>
<td>3.08e-06</td>
</tr>
<tr>
<td>37</td>
<td>-0.258</td>
<td>-0.752</td>
<td>0.203</td>
<td>-0.00264</td>
<td>-2.34e-05</td>
<td>-0.000121</td>
</tr>
<tr>
<td>38</td>
<td>0.260</td>
<td>1.10</td>
<td>-0.423</td>
<td>-0.000197</td>
<td>-5.72e-05</td>
<td>4.72e-05</td>
</tr>
<tr>
<td>39</td>
<td>1.77</td>
<td>-0.491</td>
<td>-0.191</td>
<td>-0.00267</td>
<td>9.76e-06</td>
<td>-1.80e-05</td>
</tr>
<tr>
<td>40</td>
<td>1.11</td>
<td>0.835</td>
<td>0.617</td>
<td>-6.16e-05</td>
<td>1.64e-05</td>
<td>1.69e-05</td>
</tr>
<tr>
<td>41</td>
<td>-0.413</td>
<td>0.232</td>
<td>-0.341</td>
<td>0.000191</td>
<td>-6.41e-05</td>
<td>-5.00e-05</td>
</tr>
<tr>
<td>42</td>
<td>-1.08</td>
<td>0.866</td>
<td>-0.295</td>
<td>-0.000548</td>
<td>-4.24e-05</td>
<td>-2.16e-05</td>
</tr>
<tr>
<td>43</td>
<td>0.422</td>
<td>-1.27</td>
<td>0.443</td>
<td>-0.000532</td>
<td>2.34e-05</td>
<td>-6.00e-05</td>
</tr>
<tr>
<td>44</td>
<td>-0.661</td>
<td>-0.262</td>
<td>-0.930</td>
<td>-0.00362</td>
<td>-2.94e-05</td>
<td>6.86e-05</td>
</tr>
<tr>
<td>Mean</td>
<td>0.0206</td>
<td>-0.0819</td>
<td>-0.174</td>
<td>-0.00105</td>
<td>-2.23e-05</td>
<td>-8.37e-06</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.948</td>
<td>0.821</td>
<td>0.464</td>
<td>0.00138</td>
<td>3.39e-05</td>
<td>5.83e-05</td>
</tr>
</tbody>
</table>
Table 5. Statistics for Ten 1000-Second Simulation Runs;  
Continuous Case;  \( V = 1000 \text{ ft/sec} \)

<table>
<thead>
<tr>
<th>Run</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
</tr>
<tr>
<td>35</td>
<td>5.07</td>
</tr>
<tr>
<td>45</td>
<td>5.17</td>
</tr>
<tr>
<td>46</td>
<td>5.16</td>
</tr>
<tr>
<td>47</td>
<td>5.10</td>
</tr>
<tr>
<td>48</td>
<td>5.28</td>
</tr>
<tr>
<td>49</td>
<td>5.31</td>
</tr>
<tr>
<td>50</td>
<td>5.26</td>
</tr>
<tr>
<td>51</td>
<td>5.22</td>
</tr>
<tr>
<td>52</td>
<td>4.92</td>
</tr>
<tr>
<td>53</td>
<td>4.91</td>
</tr>
<tr>
<td>Mean</td>
<td>5.14</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.141</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
</tr>
<tr>
<td>35</td>
<td>0.0393</td>
</tr>
<tr>
<td>45</td>
<td>-0.417</td>
</tr>
<tr>
<td>46</td>
<td>-0.0929</td>
</tr>
<tr>
<td>47</td>
<td>0.0400</td>
</tr>
<tr>
<td>48</td>
<td>0.551</td>
</tr>
<tr>
<td>49</td>
<td>0.391</td>
</tr>
<tr>
<td>50</td>
<td>-0.125</td>
</tr>
<tr>
<td>51</td>
<td>-0.367</td>
</tr>
<tr>
<td>52</td>
<td>0.143</td>
</tr>
<tr>
<td>53</td>
<td>-0.210</td>
</tr>
<tr>
<td>Mean</td>
<td>-0.00478</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.308</td>
</tr>
</tbody>
</table>
Table 6. Statistics for Ten 1000-Second Simulation Runs; Tustin Case; \( V = 1000 \text{ ft/sec} \)

<table>
<thead>
<tr>
<th>Run</th>
<th>Standard deviation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
<td>( v )</td>
</tr>
<tr>
<td>35</td>
<td>5.06</td>
<td>5.02</td>
</tr>
<tr>
<td>45</td>
<td>5.16</td>
<td>5.18</td>
</tr>
<tr>
<td>46</td>
<td>5.15</td>
<td>4.93</td>
</tr>
<tr>
<td>47</td>
<td>5.08</td>
<td>4.77</td>
</tr>
<tr>
<td>48</td>
<td>5.26</td>
<td>5.05</td>
</tr>
<tr>
<td>49</td>
<td>5.30</td>
<td>4.90</td>
</tr>
<tr>
<td>50</td>
<td>5.25</td>
<td>5.02</td>
</tr>
<tr>
<td>51</td>
<td>5.21</td>
<td>4.95</td>
</tr>
<tr>
<td>52</td>
<td>4.91</td>
<td>4.84</td>
</tr>
<tr>
<td>53</td>
<td>4.90</td>
<td>5.08</td>
</tr>
<tr>
<td>Mean</td>
<td>5.13</td>
<td>4.97</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.141</td>
<td>0.122</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>Mean</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( u )</td>
<td>( v )</td>
</tr>
<tr>
<td>35</td>
<td>0.0389</td>
<td>-0.0219</td>
</tr>
<tr>
<td>45</td>
<td>-0.418</td>
<td>-0.303</td>
</tr>
<tr>
<td>46</td>
<td>-0.0919</td>
<td>-0.266</td>
</tr>
<tr>
<td>47</td>
<td>0.0404</td>
<td>0.362</td>
</tr>
<tr>
<td>48</td>
<td>0.551</td>
<td>-0.157</td>
</tr>
<tr>
<td>49</td>
<td>0.390</td>
<td>0.271</td>
</tr>
<tr>
<td>50</td>
<td>-0.123</td>
<td>0.0647</td>
</tr>
<tr>
<td>51</td>
<td>-0.365</td>
<td>0.272</td>
</tr>
<tr>
<td>52</td>
<td>0.142</td>
<td>-0.416</td>
</tr>
<tr>
<td>53</td>
<td>-0.211</td>
<td>-0.0689</td>
</tr>
<tr>
<td>Mean</td>
<td>-0.00473</td>
<td>-0.0262</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.308</td>
<td>0.267</td>
</tr>
</tbody>
</table>
Table 7. Statistics for Ten 1000-Second Simulation Runs;
MIL STD Case; $V = 1000$ ft/sec

<table>
<thead>
<tr>
<th>Run</th>
<th>$u$</th>
<th>$v$</th>
<th>$w$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>5.07</td>
<td>5.06</td>
<td>5.10</td>
<td>0.0386</td>
<td>0.0258</td>
<td>0.0308</td>
</tr>
<tr>
<td>45</td>
<td>5.17</td>
<td>5.19</td>
<td>5.23</td>
<td>0.0385</td>
<td>0.0258</td>
<td>0.0308</td>
</tr>
<tr>
<td>46</td>
<td>5.17</td>
<td>4.97</td>
<td>5.01</td>
<td>0.0385</td>
<td>0.0257</td>
<td>0.0307</td>
</tr>
<tr>
<td>47</td>
<td>5.10</td>
<td>4.81</td>
<td>5.05</td>
<td>0.0383</td>
<td>0.0258</td>
<td>0.0307</td>
</tr>
<tr>
<td>48</td>
<td>5.28</td>
<td>5.06</td>
<td>5.15</td>
<td>0.0386</td>
<td>0.0260</td>
<td>0.0310</td>
</tr>
<tr>
<td>49</td>
<td>5.31</td>
<td>4.93</td>
<td>4.95</td>
<td>0.0389</td>
<td>0.0260</td>
<td>0.0307</td>
</tr>
<tr>
<td>50</td>
<td>5.27</td>
<td>5.06</td>
<td>5.05</td>
<td>0.0383</td>
<td>0.0258</td>
<td>0.0310</td>
</tr>
<tr>
<td>51</td>
<td>5.23</td>
<td>4.98</td>
<td>5.03</td>
<td>0.0385</td>
<td>0.0258</td>
<td>0.0309</td>
</tr>
<tr>
<td>52</td>
<td>4.92</td>
<td>4.88</td>
<td>5.03</td>
<td>0.0383</td>
<td>0.0257</td>
<td>0.0308</td>
</tr>
<tr>
<td>53</td>
<td>4.92</td>
<td>5.10</td>
<td>5.21</td>
<td>0.0381</td>
<td>0.0259</td>
<td>0.0308</td>
</tr>
<tr>
<td>Mean</td>
<td>5.15</td>
<td>5.00</td>
<td>5.08</td>
<td>0.0385</td>
<td>0.0258</td>
<td>0.0308</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.141</td>
<td>0.112</td>
<td>0.0906</td>
<td>0.000227</td>
<td>0.000102</td>
<td>0.000132</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run</th>
<th>$u$</th>
<th>$v$</th>
<th>$w$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>0.0387</td>
<td>-0.0229</td>
<td>-0.159</td>
<td>3.71e-05</td>
<td>-6.71e-07</td>
<td>-3.69e-06</td>
</tr>
<tr>
<td>45</td>
<td>-0.418</td>
<td>-0.301</td>
<td>-0.115</td>
<td>-0.000175</td>
<td>-4.36e-06</td>
<td>5.25e-07</td>
</tr>
<tr>
<td>46</td>
<td>-0.0919</td>
<td>-0.264</td>
<td>0.0600</td>
<td>-0.000823</td>
<td>-9.53e-06</td>
<td>-4.69e-06</td>
</tr>
<tr>
<td>47</td>
<td>0.0403</td>
<td>0.360</td>
<td>-0.148</td>
<td>-4.77e-05</td>
<td>-2.11e-07</td>
<td>-1.92e-06</td>
</tr>
<tr>
<td>48</td>
<td>0.551</td>
<td>-0.157</td>
<td>-0.0590</td>
<td>-0.000865</td>
<td>6.75e-07</td>
<td>-9.44e-06</td>
</tr>
<tr>
<td>49</td>
<td>0.390</td>
<td>0.270</td>
<td>0.201</td>
<td>-4.53e-05</td>
<td>-7.01e-06</td>
<td>-8.05e-06</td>
</tr>
<tr>
<td>50</td>
<td>-0.123</td>
<td>0.0644</td>
<td>-0.126</td>
<td>5.54e-05</td>
<td>4.44e-06</td>
<td>-9.36e-06</td>
</tr>
<tr>
<td>51</td>
<td>-0.365</td>
<td>0.270</td>
<td>-0.104</td>
<td>-0.000167</td>
<td>-3.17e-07</td>
<td>-3.44e-06</td>
</tr>
<tr>
<td>52</td>
<td>0.142</td>
<td>-0.414</td>
<td>0.144</td>
<td>-0.000173</td>
<td>8.52e-06</td>
<td>-6.00e-06</td>
</tr>
<tr>
<td>53</td>
<td>-0.211</td>
<td>-0.0691</td>
<td>-0.303</td>
<td>-0.00115</td>
<td>2.65e-06</td>
<td>7.13e-06</td>
</tr>
<tr>
<td>Mean</td>
<td>-0.00476</td>
<td>-0.0263</td>
<td>-0.0609</td>
<td>-0.000335</td>
<td>-5.82e-07</td>
<td>-3.89e-06</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.308</td>
<td>0.266</td>
<td>0.153</td>
<td>0.000437</td>
<td>5.33e-06</td>
<td>5.04e-06</td>
</tr>
</tbody>
</table>
Theoretical Statistics

Theoretical, or expected, values for the sample statistics were desired for comparison with the measured values in Tables 2 through 7. This section presents the equations used for these calculations, the code implementing these equations, and results obtained in the form of tabulated data and plots.

Equations

The following equations defining the theoretical statistics were provided by the Government for implementation and computation. Equations (62) through (64) calculate the theoretical standard deviation of sample means of the various turbulence components.

\[
\Sigma(m_i) = \sigma_i \sqrt{\frac{\tau_i}{T}} \quad i = u, p
\]

\[
\Sigma(m_i) = \sigma_i \sqrt{\frac{\tau_i}{T}} \quad i = v, w
\]

\[
\Sigma(m_i) = \sigma_j \sqrt{\frac{\tau_j}{T}} \left\{ \frac{T \tau_j (R_j - R_j^+)}{\tau_i} + T \tau_i (R_i - R_i^+) + T \tau_i^2 \left( \frac{R_{i+ - R_j^+}}{T} \right) \right\}^{1/2} \quad i, j = q, w \text{ or } r, v
\]

Equations (65) through (68) define the theoretical standard deviation of sample standard deviations for each component.

\[
\Sigma(\hat{\sigma}_i) = \sigma_i \left\{ \frac{\tau_i}{2T} \left[ \frac{9 \tau_i^2}{4T^2} + \frac{4 \tau_i^3}{T^3} - \frac{2 \tau_i^4}{4T^4} \right] \right\}^{1/2} \quad i = u, p
\]

\[
\Sigma(\hat{\sigma}_i) = \sigma_i \left\{ \frac{13 \tau_i}{2T} \left[ \frac{83 \tau_i^2}{8T^2} \right] \right\}^{1/2} \quad i = v, w
\]

\[
\Sigma(\hat{\sigma}_i) = \frac{\sigma_j^2}{\sigma_i \sqrt{\tau_i^2 T}} \sqrt{2} \left\{ S_{00} - 2 \left[ T \tau_j (R_{i+ - R_{j^+}}) + T \tau_i (R_{i- + R_{i^+}}) \right] \right\} \right\}^{1/2}
\]

where
\[ S_{00} = \frac{T\tau_i}{2} \left( R_j^2 - R_{j+}^2 \right) + \frac{T\tau_i}{2} \left( R_i^2 + R_{i+}^2 \right) + \frac{2T}{(1/\tau_i + 1/\tau_j)} \left( R_{j-} R_{j-} - R_{j+} R_{j+} \right) \]
\[ + \frac{\tau_i^2}{4} \left( R_j^2 - 2R_j R_{j-\tau} T - R_{j+}^2 + 2R_j R_{j+\tau} T \right) - \frac{\tau_i^2}{4} \left( R_i^2 + R_{i+}^2 \right) \]
\[ + \frac{2R_i (R_{j+\tau} T - R_{j+})}{(1/\tau_i + 1/\tau_j)^2} \left( R_{j-\tau} R_{j-\tau} + R_{j-} R_{j-} \right) \]
\[ + \frac{4(R_{j+\tau} R_{i+} + R_{j-\tau} R_{i-})}{(1/\tau_i + 1/\tau_j)^3} \]
\[ + \frac{\tau_i^3}{4} \left[ R_{j+\tau} (R_j T - 2R_{j+}) + R_{j-\tau} (R_j T + 2R_{j-}) \right] - \frac{3\tau_i^4}{8} \left( R_{j-\tau}^2 + R_{j+\tau}^2 \right) \]

(68)

In the above equations \( T \), defined by \( T = NT_v \), is the length of the turbulence sequence. \( N \) is the number of samples in the sequence.) The variables \( \sigma, \tau, V, T_v \) were defined previously in the section entitled Continuous Model. The variables \( R_{i-}, R_{j-}, R_{i+}, R_{j+}, R_{j-\tau}, R_{j+\tau} \) are defined in the subsequent section discussing autocorrelation.

**Code**

Equations (62) through (68) were implemented in the MATLAB m-file `newA41.m`. Comments were added to the file to identify equation numbers listed in the previous section. The variables \( \text{Smhatuu}, \text{Smhatpp}, \text{Smhatqq}, \text{Smhatrr}, \text{Smhatwv} \) defined in file `newA41.m` are the standard deviations of the sample means. The variables \( \text{Ssighatuu}, \text{Ssighatpp}, \text{Ssighatqq}, \text{Ssighatrr}, \text{Ssighatwv} \) defined in file `newA41.m` are the standard deviations of sample standard deviations.

```matlab
% Computes eq. A29, 62, 65 for u or p components
% A37-A43, 64, 67, 62 for q or r components
% A31,63, 66 for w or v components
%
% Tnu = sampling interval = 1/80
% Lu = Lw = Lv = Lvw = Dryden scale length
% sigma = sigu = sigw = std dev of turbulence

b = 37.42;
Lu = 1750;
Lwv = 1750;
Lp = sqrt(Lwv*b)/2.6;
sigma = 5.;
sigp = 1.9/sqrt(Lwv*b)*sigma;
% add sigq uses sigw; sigr uses sigv; for 67
%degs sigq = sqrt(0.05699*sigma^2);
%degs sigr = sqrt(0.07667*sigma^2);
sigq = sqrt(1.736e-05*sigma^2);
sigr = sqrt(2.336e-05*sigma^2);
Tnu = 0.0125;
Npts = 80000;
```

61
% T=1000 sec, Npts=80000
T=Npts*Tnu;
%
T2 = 2.*T;
Tsq = T^2;
Tcube = T^3;
Tfour = T^4;

Ntau = 501;
tau = linspace(-10,10,Ntau)';

tauu = Lu/V;
tauw = Lwv/V;
taup = Lp/V;
tauq = 4.*b/(pi*V);
taur = 3.*b/(pi*V);

%   **************************************************************************
% Pre-allocate vectors  (autocorrelation vectors)
%   **************************************************************************
Rii = zeros(size(tau));
Rqq = zeros(size(tau));
Rrr = zeros(size(tau));
Ruu = zeros(size(tau));
Rpp = zeros(size(tau));
Rwv = zeros(size(tau));

%   **************************************************************************
% Logic to select u,w,p,q,r etc. for sigi, tauj, etc.
%   use variable selup, or selgr, etc for equations
%   **************************************************************************

if selup == 'u'
sigi = sigma;
taui = tauu;
elseif selup == 'p'
sigi = sigp;
taui = taup;
end

%   *******************************************************
% eq. 69 eq. Rii  autocorrelation function for u,p
%   *******************************************************
% tau - 'for loop' calculations
% * * * * * * * * * * * * * * * * * * * *
% for n = 1:Ntau
% A29pwr = abs(tau(n))/taui;
% Rii(n) = sigi^2*exp(-A29pwr);
% if selup == 'u'
%   Ruu(n) = Rii(n);
% elseif selup == 'p'
%   Rpp(n) = Rii(n);
% end
% end tau-loop for eq. 69
%
% Compute expected values of sample statistics u or p components
% * * * * * * * * * * * * * * * * * * * *
% constants for 62 and 65
% eq. 62
% for 62 and 65
tisq = taui^2;
% constants for 62
UPk1 = (taui/T);
UPk2 = (tisq/Tsq);
% eq. 62 for sigma(mean-hat)
Smhat46 = sigi*sqrt(2.*(UPk1 - UPk2));
% eq. 65
% constants for 65 and use tisq defined above
ticube = taui^3;
tifour = taui^4;
% UPk4 = taui/T2;
UPk5 = 9.*tisq/(4.*Tsq);
UPk6 = 4.*ticube/(Tcube);
UPk7 = 2.*tifour/(Tfour);
% eq. 65 for sigma(sigma-hat)
Ssighat53 = sigi*sqrt(UPk4 - UPk5 + UPk6 - UPk7);
% Set all u or p calculations

if selup == 'u'
% Ruu(n) = Rii(n);
S_hatuu = Smhat46;
Ssigatuu = Ssigat53;
S_hatpp = 'undef';
Ssigatpp = 'undef';
elseif selup == 'p'
% Rpp(n) = Rii(n);
S_hatpp = Smhat46;
Ssigatpp = Ssigat53;
S_hatuu = 'undef';
Ssigatuu = 'undef';
end

% end u, p Logic selection and calculations

% Start q, r Logic selection

% Selected sigma and tau i,j = q,w or r,v

if selqr == 'q'
sigi = sigq;
sigj = sigma;
tau_i = tauq;
tau_j = tauwv;
elseif selqr == 'r'
sigi = sigr;
sigj = sigma;
tau_i = taur;
tau_j = tauwv;
end

% eq. 71 - 77 constants for Rii autocorrelation function
% for q,r sigma(mean-hat)

% for eq. 64 also- use tisq, tjsq
tisq = tau_i^2;
tjsq = tauj^2;
% tauij = tauj*taui;
% oneovtau = 1./taui;
oneovtauj = 1./tauj;
RAk3 = oneovtau + oneovtauj;
RAk12 = oneovtau / RAK3;

RAk4 = 1. / (2.*tauij);
RAk3sq = RAK3^2;
RA4ov3sq = RAK4/RAK3sq;

RAk5 = oneovtau - oneovtauj;
RAk13 = oneovtau / RAK5;
RAk5sq = RAK5^2;
RA4ov5sq = RAK4/RAK5sq;

oneovtauisq = 1./tisq;
oneovtaujsq = 1./tjsq;
RAK6 = oneovtauisq - oneovtaujsq;
RAk11 = oneovtauisq / RAK6;

RAk7 = RAK4 * oneovtau;
RAk5by3sq = RAK5 * RAK3sq;
RA5sqby3 = RAK5sq * RAK3;

RAk8 = 1. / (2.*tauj);
RAk4ov3 = RAK4 / RAK3;
RAk4ov5 = RAK4 / RAK5;
RAk7ov6 = RAK7 / RAK6;

RAk9 = 1. / (2.*tauj);
RAk9ov5 = RAK9 / RAK5;
RAk9ov3 = RAK9 / RAK3;
RAk10 = 1. / (4.*tauij);
%
RAk3cube used in Sk7den eq. 68
RAk3cube = RAK3^3;
% not - used RA3by5 = RAK3 * RAK5;
%
% Eq 72
% * * * * * * * * %
Rjneg = 1. - RAK12 + RA4ov3sq - RAK13 - RA4ov5sq ...
  + RAK11 - RAK7 / RAK5by3sq + RAK7 / RAK5sqby3 ;
%
% Eq 73
% * * * * * * * * %
Rjnegt = RAK8 - RAK4ov3 - RAK4ov5 + RAK7ov6;
% * * * * * * * * %
Rineg = RAk9ov5 - RAk9ov3 + RAk10/RAk5sq + RAk10/RAk3sq;

Rjpos = 1. - RAk13 - RA4ov5sq - RAk12 + RA4ov3sq ... 
+ RAk11 + RAk7/RA5sqby3 - RAk7/RAk5by3sq;

Rjpos = Rjpos^2;
Rjneg = Rjneg^2;
Rjpos = Rjpos^2;
Rjn = Rjn^2;
Rjptsq = Rjpos^2;
Rjnts = Rjneg^2;

% eq. 71 eq. Rii autocorrelation function for q,r
% start tau - 'for loop' for 71
% constant for 71
Riikl = sigj^2/(V^2*tisq);

for n = 1:Ntau

    A37pwr1 = tau(n)/tauj;
    A37pwr2 = tau(n)/taui;

    if tau(n) < 0
        Rii(n) = Riikl*(Rjneg*exp(A37pwr1) ... 
            + Rjneg*tau(n)*exp(A37pwr1) + Rineg*exp(A37pwr2));
    else
        Rii(n) = Riikl*(Rjpos*exp(-A37pwr1) ... 
            + Rjpos*tau(n)*exp(-A37pwr1) + Ripos*exp(-A37pwr2));
    end
if selqr == 'q'
    Rqq(n) = Rii(n);
elseif selqr == 'r'
    Rrr(n) = Rii(n);
end
%
end % end tau-loop for eq. 71
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
%
% Compute expected values of sample statistics for q or r components
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% eq. 64
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
%
% constants needed for i,j = q,w or r,v
% for 64 and tisq, tjsq defined above
Tti = T*taui;
Ttj = T*tauj;
tjcube = tauj^3;
% for 68 also use tjcube, Tti, Ttj defined above
tfour = tauj^4;
%
%ticube = taui^3;
%tifour = taui^4;
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% 64 use tisq, tjsq calcs defined above
% constants for 64
%
QRk1 = sigj/(V*Tti);
QRk2 = Ttj*(Rjneg + Rjpos);
QRk3 = Tti*(Rineg + Ripos);
QRk4 = T*tjsq*(Rjneg - Rjnegt);
QRk5 = tjsq*(Rjpos + Rjneg);
QRk6 = tisq*(Rpos + Rineg);
QRk7 = 2.*tjcube*(Rjneg - Rjpos);
QRk2toQRk7 = (QRk2 + QRk3 + QRk4 - QRk5 - QRk6 + QRk7);
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% eq. 64 for q,r sigma(mean-hat)
%
Smhat50 = QRk1*sqrt(QRk2toQRk7);%
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
%
% eq. 68 constants for S00
%
Sk1 = Ttj/2.*(Rjnsq + Rjpsq);
Sk2 = Tti/2.*(Rinsq + Ripsq);
Sk3 = T2/RAk3*(Rjneg*Rjneg + Rjpos*Ripos);
Sk4 = Tjsq/4.*(-Rjnsq - 2.*Rjneg*Rjnegt + Rjpsq - 2.*Rjpos*Rjpost);  
Sk5 = tisq/4.*(Rinsq + Ripsq);
Sk6num = 2.*Ripos*(Rjpost*T - Rjpos) - 2.*Rineg*(Rjneg*T + Rjneg);  
Sk6den = RAk3sq;  
Sk7num = 4.*(Rjpost*Ripos + Rjnegt*Rineg);  
Sk7den = RAk3cube;  
Sk8 = (Rjpost*(Rjpost*T - 2.*Rjpos));  
Sk9 = (Rjnegt*(Rjnegt*T + 2.*Rjneg));  
Sk10 = tjcube/4.*(Sk8 + Sk9);  
Sk11 = 3.*tjfour/8.*(Rjntsq + Rjptsq);  

% eq. 68 calculation - S00 constant  
S00 = Sk1 + Sk2 + Sk3 + Sk4 - Sk5 + Sk6num/Sk6den ...  
+ Sk7num/Sk7den + Sk10 - Sk11 ;  

% eq. 67  
% constants for 67  
QRk8:sigj^2/(sigi*(V*taui)^2*T*sqrt(2.));  
% eq. 67 for sigma(sigma-hat)  
Ssighat62 = QRk8*sqrt(S00 - 2.*(QRk2toQRk7)^2);  

% Set all q or r calculations  
% if selqr == 'q'  
% Rqq(n) = Rii(n);  
% Smhatqq = Smhat50;  
% Ssighatqq = Ssighat62;  
% Sihatrr = 'undef';  
% Sihatrr = 'undef';  
elseif selqr == 'r'  
% Rrr(n) = Rii(n);  
% Sihatrr = Smhat50;  
% Sihatrr = Ssighat62;  
% Sihatqq = 'undef';  
% Sihatqq = 'undef';  
end  

% w or v calculation  
% w or v calculation
% eq.70 Rwv  autocorrelation function for v,w
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% tau - 'for loop' calculations
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
for n = l:Ntau
    abstau = abs(tau(n));
    A31pwr = abstau/tauwv;
    WVk1 = 3.*sigma^2/tauwv;
    WVk2 = tauwv/3.;
    WVk3 = (1./6.)*abstau;
    Rwv(n) = WVk1*(WVk2*exp(-A31pwr) - WVk3*exp(-A31pwr));
end % end tau-loop for eq. 70
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% eq. 63 for v, w sigma(mean-hat)
% Smhat48 = sigma*sqrt(tauwv/T);
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% eq. 66 for v, w sigma(sigma-hat)
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% WVk4 = 13.*tauwv/(4.*T);
% WVk5 = 83*tauwv^2/(8.*Tsq);
% Ssighat55 = (sigma/2.)*sqrt(WVk4 - WVkS);
% Smhatwv = Smhat48;
% Ssighatwv = Ssighat55;
% % Display expected values of sample statistics
% Smhatuu
% Smhatpp
% Smhatqq
% Smhatrr
% Ssighatuu
% Ssighatpp
% Ssighatqq
% Ssighatrr
% Smhatwv
% Ssighatwv

**Autocorrelation**

The theoretical autocorrelation functions were needed to compute some of the theoretical statistics discussed previously. The functions were also used for comparison with correlation functions computed from the sample sequences produced with the GUSTMDL program.
Equations

The autocorrelation function $R_{ii}(\tau)$, $i = u, p$, for the $u$- and $p$-components can be found from the Fourier transform of the spectral density as follows:

$$R_{ii}(\tau) = F^{-1}\{S_{ii}(\omega)\} = F^{-1}\left\{\frac{\sigma_i^2\tau_i}{\pi} \frac{1}{1 + (\tau_i\omega)^2}\right\} \quad i = u, p$$

$$= \sigma_i^2 e^{-|\tau|/\tau_i}$$

(69)

The autocorrelation function for the $w$-component is

$$R_{ww}(\tau) = \frac{3\sigma_w^2}{\tau_w} \left[ \frac{\tau_w}{3} e^{-|\tau|/\tau_w} - \frac{1}{6} |\tau| e^{-|\tau|/\tau_w} \right]$$

(70)

Since the spectra of the $v$-component and $w$-component are the same, the autocorrelation function $R_{vv}(\tau)$ is also given by equation (70). For the $q$- and $r$-components

$$R_{ij}(\tau) = \begin{cases} \frac{\sigma_j^2}{V^2\tau_i^2} \left[ R_{j-\tau} \right. & \left. - \frac{1}{2} e^{\tau/\tau_i} \frac{1}{(\tau_i + 1/\tau_j)} \left( \frac{1}{\tau_j} - 1/\tau_i \right)^2 \right] \text{ for } \tau < 0 \\
\frac{\sigma_j^2}{V^2\tau_i^2} \left[ R_{j+\tau} \right. & \left. - \frac{1}{2} e^{-\tau/\tau_i} \frac{1}{(\tau_i + 1/\tau_j)} \left( \frac{1}{\tau_j} - 1/\tau_i \right)^2 \right] \text{ for } \tau \geq 0 
\end{cases}$$

(71)

where

$$R_{j-\tau} = \frac{1}{2\tau_j} - \frac{1}{(\tau_i + 1/\tau_j)} - \frac{1}{(\tau_i - 1/\tau_j)} + \frac{1}{2\tau_j \tau_i}$$

(72)

$$R_{j+\tau} = \frac{1}{2\tau_j} - \frac{1}{(\tau_i + 1/\tau_j)} - \frac{1}{(\tau_i - 1/\tau_j)} + \frac{1}{2\tau_j \tau_i}$$

(73)

$$R_i = \frac{1}{(\tau_i - 1/\tau_j)} - \frac{1}{(\tau_i + 1/\tau_j)} + \frac{1}{(\tau_i + 1/\tau_j)^2} + \frac{1}{(\tau_i - 1/\tau_j)^2}$$

(74)
These equations were coded in the MATLAB m-file newA41.m presented in the previous section. Comments were added to the code to refer to equation numbers listed in this section. The variables $R_{uu}$, $R_{pp}$, $R_{qq}$, $R_{rr}$, and $R_{wv}$ are the autocorrelation functions from equations (69) through (71) for the turbulence components. The variables $R_{jneg}$, $R_{jnegt}$, $R_{ineg}$, $R_{jpos}$, $R_{jpost}$, and $R_{ipos}$ in file newA41.m are the constants defined in equations (72) through (77).

**Code**

Code in MATLAB file A41plts.m was generated to plot the theoretical autocorrelation functions discussed above that are calculated in file newA41.m. The MATLAB files newA41.m and A41plts.m should be executed sequentially to be able to plot theoretical autocorrelation functions. The file A41plts.m also computes and plots the sample autocorrelation functions of the turbulence sequences produced by the GUSTMDL program. These autocorrelation functions were computed using the MATLAB function `xcorr` from the Signal Processing Toolbox. The code for A41plts.m follows.

```matlab
% autocorrelation plots
%
% Theoretical autocorrelation plots calculated in newA41.m
%
% Select u or p component
% if selup == 'u'
% plot(tau,Ruu)
title('Autocorrelation Ruu V100')
xlabel('tau, sec')
ylabel('Autocorrelation')
% pause
%print Ruu
%
```
elseif selup == 'p'

plot(tau,Rpp)
title('Autocorrelation Rpp VI00')
xlabel('tau, sec')
ylabel('Autocorrelation')
% pause
% print Rpp
%
end
%
%
% Select q or r component
%
if selqr == 'q'
%
plot(tau,Rqq)
title('Theoretical autocorrelation Rqq VI00')
xlabel('tau, sec')
ylabel('Autocorrelation')
% pause
% print Rqq
%
elseif selqr == 'r'
%
plot(tau,Rrr)
title('Theoretical autocorrelation Rrr VI00')
xlabel('tau, sec')
ylabel('Autocorrelation')
% pause
% print Rrr
%
end
%
% Select w or v component
%
plot(tau,Rwv)
title('Theoretical autocorrelation Rwv VI00')
xlabel('tau, sec')
ylabel('Autocorrelation')
% pause
% print Rwv
%
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
Experimental autocorrelation plots from GUSTMDL
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
if V==100.

gdload('gustr54sigs.cmp3')
elseif V==1000.
%     gdload('gustr55sigs.cmp3')
end

% * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% compute experimental correlation (corrariance) function
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Ntx=1000;
dt=.0125;
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% Select w or v component to determine tx (plot time)
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% if selwv == 'v'

[Rvvc] = xcorr(turbv, turbv, 'biased');
[RvvT] = xcorr(turbv, turbv, 'biased');
[RvvM] = xcorr(turbv, turbv, 'biased');

tx = ((1:length(Rvvc))' - fix(length(Rvvc)/2) - 1)*dt;
% Select middle +-10 seconds
nx = fix(length(tx)/2) + 1;
tplt = tx(nx-800:nx+800);
Rvvcplt = Rvvc(nx-800:nx+800);
RvvTplt = RvvT(nx-800:nx+800);
RvvMplt = RvvM(nx-800:nx+800);

% Plot w or v component
%
plot(tplt,Rvvcplt,'k-',tplt,RvvTplt,'y-',tplt,RvvMplt,'m-',...
tau,Rwv,'k--')
xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Rvvc','RvvT','RvvM','Rwv')
% if V==100.
% title('Experimental vs. Theoretical autocorrelation Rvv V100')
axis([-2. 2. -5. 25.])
% elseif V==1000.
% title('Experimental vs. Theoretical autocorrelation Rvv V1000')
elseif selvw=='w'

[Rwwc] = xcorr(turbw, turbw, 'biased');
[RwwT] = xcorr(turbw, turbw, 'biased');
[RwwM] = xcorr(turbw, turbw, 'biased');

tx = ([1:length(Rwwc)]' - fix(length(Rwwc)/2) - 1)*dt;

% Select middle +10 seconds

nx = fix(length(tx)/2) + 1;
tplt = tx(nx-800:nx+800);
Rwwcplt = Rwwc(nx-800:nx+800);
RwwTplt = RwwT(nx-800:nx+800);
RwwMplt = RwwM(nx-800:nx+800);

% Plot w or v component

plot(tplt,Rwwcplt,'k-',tplt,RwwTplt,'y-',tplt,RwwMplt,'m-',...
    tau,Rwv,'k--')

xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Rwwx','Rwv')

if V==100.

    title('Experimental vs. Theoretical autocorrelation Rww V100')
    axis([-2. 2. -5. 25.])

elseif V==1000.

    title('Experimental vs. Theoretical autocorrelation Rww V1000')
    axis([-2. 2. -10. 25.])
end

pause
end

% Select u or p component

if selup == 'u'

[Ruuc] = xcorr(turbu, turbu, 'biased');
[RuuT] = xcorr(turbu, turbu, 'biased');
[RuuM] = xcorr(turbu, turbu, 'biased');
Ruucplt = Ruuc(nx-800:nx+800);
Ruulplt = RuuT(nx-800:nx+800);
RuuMplt = RuuM(nx-800:nx+800);

plot(tplt,Ruucplt,'k-',tplt,RuuTplt,'y-',tplt,RuuMplt,'m-','...
tau,Ruu,'k--')

xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Ruuc', 'RuuT', 'RuuM', 'Ruu')

if V==100.
    title('Experimental vs. Theoretical autocorrelation Ruu V100')
    axis([-2. 2. -5. 35.])
elseif V==1000.
    title('Experimental vs. Theoretical autocorrelation Ruu V1000')
    axis([-2. 2. -10. 30.])
end
pause
elseif selup == 'p'
    [Rppc] = xcorr(turbp, turbp, 'biased');
    [RppT] = xcorr(turbp, turbp, 'biased');
    [RppM] = xcorr(turbp, turbp, 'biased');

Rppcplt = Rppc(nx-800:nx+800);
RppTplt = RppT(nx-800:nx+800);
RppMplt = RppM(nx-800:nx+800);

plot(tplt,Rppcplt,'k-',tplt,RppTplt,'y-',tplt,RppMplt,'m-','...
tau,Rpp,'k--')

xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Rppx', 'Rpp')

if V==100.
    title('Experimental vs. Theoretical autocorrelation Rpp V100')
    axis([-2. 2. -2.e-4 14.e-4])
elseif V==1000.

75
title('Experimental vs. Theoretical autocorrelation Rpp V1000')
axis([-2. 2. -2.e-4 14.e-4])
end

% pause
end

% Select q or r component
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
% if selqr == 'q'
% [Rqqc] = xcorr(turbq, turbq, 'biased');
[RqqT] = xcorr(turbq, turbq, 'biased');
[RqqM] = xcorr(turbq, turbq, 'biased');
Rqqcplt = Rqqc(nx-800:nx+800);
RqqTplt = RqqT(nx-800:nx+800);
RqqMplt = RqqM(nx-800:nx+800);

% plot(tplt,Rqqcplt,'k-',tplt,RqqTplt,'y-',tplt,RqqMplt,'m-',...%
tau,Rqq,'k--')

xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Rqqc','RqqT','RqqM','Rqq')

% if V==100.
% title('Experimental vs. Theoretical autocorrelation Rqq V100')
% axis([-2. 2. -1.e-4 5.e-4])
% elseif V==1000.
% title('Experimental vs. Theoretical autocorrelation Rqq V1000')
% axis([-2. 2. -1.e-4 5.e-4])
% end

% pause
% elseif selqr == 'r'
%
[Rrrc] = xcorr(turbr, turbr, 'biased');
[RrrT] = xcorr(turbr, turbr, 'biased');
[RrrM] = xcorr(turbr, turbr, 'biased');
Rrrcplt = Rrrc(nx-800:nx+800);
RrrTplt = RrrT(nx-800:nx+800);
RrrMplt = RrrM(nx-800:nx+800);

plot(tplt, Rrrcplt, 'k-', tplt, RrrTplt, 'y-', tplt, RrrMplt, 'm-', ... 
tau, Rrr, 'k--')

xlabel('tau, sec')
ylabel('Autocorrelation')
grid
legend('Rrrc', 'RrrT', 'RrrM', 'Rrr')

if V==100.
    title('Experimental vs. Theoretical autocorrelation Rrr V100')
    axis([-2. 2. -1.e-4 6.e-4])
elseif V==1000.
    title('Experimental vs. Theoretical autocorrelation Rrr V1000')
    axis([-2. 2. -1.e-4 6.e-4])
end

pause

Plots

Example plots of autocorrelation functions produced by A41plots.m for V = 100 ft/sec and V = 1000 ft/sec are found in this section. The sample autocorrelation functions are for 819-sec turbulence sequences produced using GUSTMDL.
Figure 13. Autocorrelation functions for $V = 100$ ft/sec.
Figure 14. Autocorrelation functions for $V = 1000$ ft/sec.
Aircraft Simulation

The turbulence models and implementations discussed in previous sections of this report were developed for use in aircraft simulations, primarily nonlinear six-degree-of-freedom simulations. The models were installed in a nonlinear six-degree-of-freedom simulation of the High Alpha Research Vehicle (refs. 3 and 4) for evaluation and comparison among the three models. The HARV simulation is written in ACSL with FORTRAN subroutines to implement input/output, sensor models, and control laws. The equations of motion are integrated in the ACSL DERIVATIVE BLOCK (ref. 5).

Code

Much of the code installed in the HARV simulation to implement the turbulence models was taken without change from the GUSTMDL program. In GUSTMDL the coefficients in the differential and difference equations were computed one time in the INITIAL BLOCK, since aircraft airspeed VTOT was kept constant. In the HARV simulation some of these coefficients must be computed in the DERIVATIVE or DISCRETE BLOCKS to accommodate changes in airplane airspeed during the simulated flight.

In this section the entire HARV simulation code will not be presented for to do so would significantly increase the length of the report while adding very little information relative to the turbulence models and their implementation. Instead, portions of code which show all of the modifications made to the simulation to install the turbulence models will be included. In the code presentation below, to avoid confusion sections of simulation code are separated from portions of report text by a line of #’s.

In the MACRO SECTION MACRO DRMS was inserted after the existing MACRO ALFDYN. The code for MACRO DRMS follows and was used to compute statistics needed for evaluation of the performance of the three turbulence models.

MACRO DRMS(RMSX, MNX, X, N)
MACRO TO COMPUTE SAMPLE STATISTICS OF RANDOM VARIABLE
MACRO REDEFINE SUM, SUMSQ, EPS
CONSTANT SUM = 0., SUMSQ = 0.
CONSTANT EPS = 1.E-22
SUM = SUM + X
SUMSQ = SUMSQ + X*X
MNX = SUM/N
RMSX = SQRT(MAX((SUMSQ - MNX*SUM)/(MAX(N-1.,EPS)),0.))
MACRO EXIT
MACRO END

In the INITIAL BLOCK constants were initialized. The code showing initial values needed by the implementation of the three turbulence models follows. Comments indicating equation numbers refer to the previously defined equations in this report.

RA2DG = 180./ACOS(-1.)
DG2RA = 1./RA2DG
CONSTANT UWG : 0.
CONSTANT VWG : 0.
CONSTANT WWG : 0.

CONSTANT UWND : 0.
CONSTANT VWND : 0.
CONSTANT WWND : 0.

CONSTANT VWNDIC = 0., &
HDWIC = 0., &
VWNDRT = 0., &
HDWRT = 0.

Still in the INITIAL BLOCK definitions of turbulence constants and initial values of turbulence variables were placed after the END OF/EULER ANGLE/QUATERNION/DIRECTION COSINE/ TRIM UPDATE comment in the aircraft simulation.

!-------- END OF/EULER ANGLE/QUATERNION/DIRECTION COSINE/ TRIM UPDATE.!
!******************************************************************************
!
******************************************************************************
!-------- START IMPLEMENTATION OF TURBULENCE MODEL WTB,JCY 9-10-97!
!******************************************************************************

CONSTANT SDNOIS = 1., &
TSAMP = .0125, &
TURBL = 1750., &
TURBSIG = 5.

CONSTANT FILNU = 0., &
FILNV = 0., &
FILNW = 0., &
FILNP = 0., &
BWING = 37.4

CONSTANT FILU = 0., &
FILV = 0., &
FILW = 0., &
FILP = 0., &
FILQ = 0., &
FILR = 0.

CONSTANT MILUK = 0., &
MILVK = 0., &
MILWK = 0., &
MILPK = 0., &
MILQK = 0., &
MILRK = 0.

Character variable TURBFLG was defined to allow a choice of which turbulence model to be implemented during execution of the program. Values of this variable can be 'CON', 'TUS', or 'MIL'.

81
CHARACTER TURBFLG*3
CONSTANT TURBFLG = 'TUS'
CONSTANT EPSLON = 1.e-22
CONSTANT SAMPS = 0.
!
LOGICAL PQRFLG
!
CONSTANT PQRFLG = .TRUE.

Logical variable PQRFLG was defined to allow a choice to use p-, q-, or r- gusts to turbulence model outputs. Values of .True. or .False. should be used for this variable. The following variables were defined before the end statement of the INITIAL BLOCK as in the GUSTMDL program. These variables are constant and do not depend on velocity.

pi = acos(-1.)
!
irseed should be pos, odd, integer, possibly 8 digits
!
multiply defined irseed    CONSTANT IRSEED = 28545269
GAUSI( IRSEED )
!
CONSTANT TURBU0 = 0., &
TURBVD0 = 0., &
TURBV0 = 0., &
TURBW0 = 0., &
TURBXW0 = 0.
!
CONSTANT TURBR0 = 0., &
TURBQ0 = 0., &
TURBP0 = 0.
!
SIGU = TURBSIG           ! same as sigv, sigw
SIGV = TURBSIG
SIGW = TURBSIG
SIGP = 1.9 / SQRT(TURBL * BWING) * SIGW    ! eq.8
!
APPROXIMATIONS OF EXPECTED VALUES OF STD. DEV. (P AND R COMPONENTS)
!
SIGQ = SQRT(pi/(2.*TURBL * BWING)) * SIGW  ! eq.14
SIGR = SQRT(2.*pi/(3.*TURBL * BWING)) * SIGW  ! eq.17
!
**********************************************************************
END $ ! OF INITIAL 

DYNAMIC

**********************************************************************
DERIVATIVE

82
Variables implementing the continuous models were placed in the DERIVATIVE BLOCK of the ACSL simulation. They were inserted with the steady-state wind model after the END statement for the linearization procedural and before the table look-ups for the HARV aerodynamic model. This placement of the model allows the latest turbulence velocities to be included in the calculation of aerodynamic forces and moments.

```
! NOLLN..CONTINUE
!
ENDS !ENGINE LINEARIZATION PROCEDURAL!
!
--- -- COMPUTE BODY FRAME COMPONENTS OF WIND!
VELWND = VWNDIC + VWNDRT*( H - 15000.)/1000.
HEADWND = HDWIC + HDWRT*( H - 15000.)/1000.
XDWND = VELWND * COS(HEADWND*DG2RA)
YDWND = -VELWND * SIN(HEADWND*DG2RA)
ZDWND = 0.
XDWDND = VWNDRT*(HD/1000.)*COS(HEADWND*DG2RA)
- VELWND*Sin(HEADWND*DG2RA)*HDWRT*(HD/1000.)*DG2RA
YDWDND = -VWNDRT*(HD/1000.)*SIN(HEADWND*DG2RA)
- VELWND*COS(HEADWND*DG2RA)*HDWRT*(HD/1000.)*DG2RA
ZDWDND = 0.
UWND = CXX*XDWND + CXY*YDWND + CXZ*ZDWND
VWND = CYX*XDWND + CYY*YDWND + CYZ*ZDWND
WWND = CZX*XDWND + CZY*YDWND + CZZ*ZDWND
!
! * * * * * * * * * * *
!
! Turbulence or Gust components
!
UAM = U - UWND
VAM = V - VWND
WAM = W - WWND
UA2M = UAM*UAM
VA2M = VAM*VAM
WA2M = WAM*WAM
!
```

The variable VTMINUS defines a “pseudo steady-state” value of true airspeed for use in calculating turbulence model coefficients. VTMINUS is the total air speed minus, or without, turbulence.

```
VTMINUS = SQRT( UA2M + VA2M + WA2M )
VTOT = VTMINUS
!
! use turb1= 1750. for Lu, Lv, Lw
! use tau for tauu, tauv, tauvw
!
TAU = TURBL/VTOT  ! eq.3, 4, 6
TAUQ = 4.0 * BWING / (PI * VTOT)  ! eq.13
TAUR = 3.0 * BWING / (PI * VTOT)  ! eq.16
!```
\[ LP = \sqrt{TURBL \times BWING} / 2.6 \]  \quad \text{! eq.10}

\[ TAUP = LP / VTOT \]  \quad \text{! eq.13}

\[ TURBOMEGA = VTOT / TURBL \quad \text{! same as 1./TAU, 1./TAUV, 1./TAUW} \]

\[ WP = VTOT / LP \quad \text{! same as 1./TAUP} \]

\[ TURBK2U = TURBSIG \times \sqrt{2 \times TURBOMEGA / TSAMP} \]

\[ K2P = SIGP \times \sqrt{2 \times WP / TSAMP} \]

\[ TURBK1VW = 2 \times TURBOMEGA \]

\[ TURBK2VW = TURBOMEGA \times TURBOMEGA \]

\[ TURBK3VW = TAU \times \sqrt{3} \]

\[ TURBK4VW = TURBSIG \times \sqrt{TURBOMEGA^3 / TSAMP} \]

\[ \text{TURBULENCE MODEL WTB, JCY 2-24-94} \]

\[ \text{U - COMPONENT} \]

\[ \text{used in eq.5} \]

\[ TURBUD = - TURBOMEGA \times TURBU + TURBK2U \times FILNU \]

\[ TURBU = \text{INTVC (TURBUD, TURBU0)} \]

\[ \text{V - COMPONENT} \]

\[ \text{used in eq.2} \]

\[ TURBXVDD = - TURBK1VW \times TURBXVD - TURBK2VW \times TURBXV + TURBK4VW \times FILNV \]

\[ TURBXVD = \text{INTVC (TURBXVDD, TURBXVD0)} \]

\[ TURBXVDI = TURBXVD \]

\[ TURBXV = \text{INTVC (TURBXVDI, TURBXV0)} \]

\[ TURBV = TURBXV + TURBK3VW \times TURBXVDI \]

\[ TURBV = \text{INTVC (TURBV, TURBV0)} \]

\[ \text{W - COMPONENT} \]

\[ \text{used in eq.2} \]

\[ TURBXWDD = - TURBK1VW \times TURBXWD - TURBK2VW \times TURBXW + TURBK4VW \times FILNW \]

\[ TURBXWD = \text{INTVC (TURBXWDD, TURBXWD0)} \]

\[ TURBXWDI = TURBXWD \]

\[ TURBXW = \text{INTVC (TURBXWDI, TURBXW0)} \]

\[ TURBW = TURBXW + TURBK3VW \times TURBXWDI \]

\[ TURBW = \text{INTVC (TURBW, TURBW0)} \]

\[ \text{P, Q, R COMPONENTS ADDED 5-15-95 JCY} \]
**P - COMPONENT**

```
TURBP = INTVC (TURBPD, TURBP0)
```

**Q - COMPONENT**

```
TURBQ = INTVC (TURBQD, TURBQ0)
```

**R - COMPONENT**

```
TURBR = INTVC (TURBRD, TURBR0)
```

---

The following code selects the continuous, Tustin, or MIL STD models for use in the aircraft simulation.

```
PROCEDURAL(wwg, TURBU, TURBV, TURBW, TURBP, TURBQ, TURBR)
```

**P - COMPONENT**

```
TURBP = - WP * TURBP + K2P * FILNP
TURBP = INTVC (TURBPD, TURBP0)
```

**Q - COMPONENT**

```
TURBQD = - TURBQ / TAUQ + TURBWD / (TAUQ * VTOT)
TURBQ = INTVC (TURBQD, TURBQ0)
```

**R - COMPONENT**

```
TURBRD = - TURBR / TAUR + TURBVD / (TAUR * VTOT)
TURBR = INTVC (TURBRD, TURBR0)
```

---

The following code selects the continuous, Tustin, or MIL STD models for use in the aircraft simulation.
RWG = MILRK
IF (.not. PQRFLG) THEN
  PWG = 0.
  QWG = 0.
  RWG = 0.
END IF
UDWG = FILUD2
VDWG = FILVD2
WDWG = FILWD2
ELSE IF (TURBFLG .EQ. 'CON' .AND. T .GT. 0.) THEN
  UWG = TURBU
  VWG = TURBV
  WWG = TURBW
  PWG = TURBP
  QWG = TURBQ
  RWG = TURBR
  IF (.not. PQRFLG) THEN
    PWG = 0.
    QWG = 0.
    RWG = 0.
  END IF
  UDWG = TURBUD
  VDWG = TURBVD
  WDWG = TURBWD
ELSE
  UWG = 0.
  VWG = 0.
  WWG = 0.
  PWG = 0.
  QWG = 0.
  RWG = 0.
  UDWG = 0.
  VDWG = 0.
  WDWG = 0.
END IF
END $!OF PROCEDURAL TO CALCULATE turbulence!

!......................................................................
....... Compute time rate of change of body frame components of wind.
!....... Vb(ody) = [L] Ve(arth)
!....... d/dt Vb = (d/dt[L])Ve + [L](d/dt Ve)
!....... d/dt[L] = - [omega][L] Etkin, eq (5.2,11)
!....... d/dt Vb = - [omega][L]Ve + [L](d/dt Ve)
!......................................................................
UDWND = r*vwg - q*wwg + CXX*XDDWND + CXY*YDDWND + CXZ*ZZDWND
VDWND = -r*uwg + p*wwg + CYX*XDDWND + CYY*YDDWND + CYZ*ZZDWND
WDWND = q*uwg - p*vwg + CZX*XDDWND + CZY*YDDWND + CZZ*ZZDWND
UA = U - UWND - UWG
VA = V - VWND - VWG
WA = W - WWND - WWG
UDA = UD - UDWND - UDWG
VDA = VD - VDWND - VDWG
WDA = WD - WDWND - WDWG
The variables PA, QA, and RA define instantaneous attitude rates relative to the turbulent atmosphere for use in the aerodynamic model table look-ups.

Calc PA, QA, RA

\[
\begin{align*}
PA &= P + PWG \\
QA &= Q + QWG \\
RA &= R + RWG
\end{align*}
\]

PROCEDURAL (TMPK=H)

\[
\begin{align*}
\text{CALL ATMAT62 (TMPK, H, 3)}
\end{align*}
\]

TMPKOLD = TMPK

\[
\begin{align*}
\text{TMPK} &= \text{TMPK} + \text{DELT}\!
\end{align*}
\]

END S1OF PROCEDURAL TO CALCULATE TEMPERATURE IN KELVIN!

Compute wind axes variables!

\[
\begin{align*}
\text{CSALF} &= \cos(\text{ALF}) \\
\text{CSBET} &= \cos(\text{BET}) \\
\text{SNALF} &= \sin(\text{ALF}) \\
\text{SNBET} &= \sin(\text{BET})
\end{align*}
\]

\[
\begin{align*}
\text{PW} &= (P \times \text{CSALF} \times \text{CSBET} + R \times \text{SNALF} \times \text{CSBET} + (Q - \text{ALFD}) \times \text{SNBET}) \\
\text{QW} &= (-P \times \text{CSALF} \times \text{SNBET} - R \times \text{SNALF} \times \text{SNBET} + (Q - \text{ALFD}) \times \text{CSBET}) \\
\text{RW} &= (-P \times \text{SNALF} + R \times \text{CSALF} + \text{BETD}) \\
\text{PWDG} &= \text{PW} \times \text{RA2DG} \\
\text{QWDG} &= \text{QW} \times \text{RA2DG} \\
\text{RWDG} &= \text{RW} \times \text{RA2DG} \\
\text{AYW} &= \text{VT} \times \text{RW} \\
\text{AZW} &= -\text{VT} \times \text{QW}
\end{align*}
\]

The variables PA, QA, and RA were used as parameters in the calls to subroutines SFAERRF and AEROINC, which define the HARV aerodynamic model, to replace the inertial variables P, Q, and R. These variables (PA, QA, and RA) were also used in the PROCEDURAL surrounding the calls to subroutines SFAERRF and AEROINC. Existing variables ALDGRF, BEDGRF, MACHRF were used in the calls to the aero models, but these variables now include the effects of turbulence.

PROCEDURAL ( &
  FVRA &
  ,CXRFSF ,CYRFSF ,CZRF SF ,ClRFSF ,CMRF SF ,CNRF SF &
  ,CDRF SF ,CLRFSF ,TVJP AVG ,TVJJAVG ,TVJRAVG &
  ,cno ,cnbeta ,cnlef ,cntef ,cna il ,cnrud ,cnraero &
  ,cnp ,clo ,cibeta ,cilef ,citef ,clail ,clail ,clrud &
The Tustin model and the MIL STD model were implemented in the DISCRETE BLOCK GUST which computes the turbulence outputs from these models at intervals of 0.0125 seconds (80 Hz).
IF(SDNOIS.LE.EPSLON) GOTO LETAI
FILNU = GAUSS(0.,SDNOIS)
FILNV = GAUSS(0.,SDNOIS)
FILNW = GAUSS(0.,SDNOIS)
FILNP = GAUSS(0.,SDNOIS)
GOTO LETA2
LETA1..CONTINUE
FILNU = 0.
FILNV = 0.
FILNW = 0.
FILNP = 0.
LETA2..CONTINUE

CONSTANTS FOR CALCULATIONS of TURBULENCE VIA TUSTIN TRANSFORM

TWOPIOVTNU = SQRT(2.*PI/TSAMP)

U-COMPONENT

constants used in eq.18

KFIL = TURBSIG* SQRT(1./ (PI * TURBOMEGA))
CFIL = TURBOMEGA/ TAN(TURBOMEGA*TSAMP/2.) ! eq.19
FILK1 = (TURBOMEGA - CFIL)/(TURBOMEGA + CFIL)
FILK2 = KFIL /{(1. + (CFIL/TURBOMEGA))
FILK3 = SQRT(PI * (TURBOMEGA + CFIL))/SDNOIS

V-COMPONENT and W-COMPONENT

constants used in eq.20

KVW = SQRT(3.*TURBOMEGA/(2.*PI))
WNPC = TURBOMEGA + CFIL
FILK4 = 2.*(TURBOMEGA*TURBOMEGA - CFIL*CFIL)/(WNPC*WNPC)
FILK5 = (TURBOMEGA - CFIL)*(TURBOMEGA - CFIL)/(WNPC*WNPC)
FILK6 = CFIL + TURBOMEGA/SQRT(3.)
FILK7 = 2.*TURBOMEGA/SQRT(3.)
FILK8 = TURBOMEGA/SQRT(3.) - CFIL
FILK9 = KVW*TURBSIG / (WNPC*WNPC)

used in eq.90

KFILDI = TURBSIG*SQRT(1.*TAU/TSAMP)

potential alternate (filwd2, filvd2)
KWD1 = (1. - TAU*CFIL)/(1. + TAU*CFIL)
KWD2 = TURBSIG*SQRT(TAU/TSAMP)*CFIL/(1.+TAU*CFIL)**2
CTWSR3 = SQRT(3.)*TAU*CFIL
!
******************************************************************************!
! P-COMPONENT
******************************************************************************!
!
used in eq.21
!
PFIL = SIGP* SQRT(2./ (TSAMP * WP))
PCFIL = WP / TAN( WP * TSAMP/2.) ! eq.22
PK1 = (WP - PCFIL) / (WP + PCFIL)
PK2 = PFIL / (1. + (PCFIL/WP))
******************************************************************************!
! Q-COMPONENT and R-COMPONENT
******************************************************************************!
!
used in eq.23
!
CFILQ = (1. / TAUQ) / TAN(TSAMP / (2. * TAUQ)) ! eq. 24
QK1 = (1. - CFILQ * TAUQ) / (1. + CFILQ * TAUQ)
QK2 = (CFILQ / VTOT) / (1. + CFILQ * TAUQ)
!
used in eq.25
!
CFILR = (1. / TAUR) / TAN(TSAMP / (2. * TAUR)) ! eq. 26
RK1 = (1. - CFILR * TAUR) / (1. + CFILR * TAUR)
RK2 = (CFILR / VTOT) / (1. + CFILR * TAUR)
******************************************************************************!
!
constants for MIL STD calcs 8-28-97
!
used in eqs. 30 - 35
!
tovtau = tsamp/tau !(for u,w,v)
tovtau2= 2.* tovtau

tovtau4= 4.* tovtau
tovtaup = tsamp/taup !(for p)
tovtaup2= 2.* tovtaup
tovtauq = tsamp/tauq !(for q)
tovtaur = tsamp/taur !(for q)
milk1 = (1. - tovtau) !(for u,w,v)
milk2 = (1. - tovtau2)
milk3 = sqrt(tovtau2)
milk4 = sqrt(tovtau4)
milk5 = (1. - tovtaup) !(for p)
milk6 = sqrt(tovtaup2)
milk7 = (1. - tovtauq) !(for q)
milk8 = (1. - tovtaur) !(for r)
piov4b=pi/(4.*bwing)
piov3b=pi/(3.*bwing)
**TURBULENCE VIA BILINEAR TRANSFORM, OR TUSTIN TRANSFORM**

**U-COMPONENT**

```
UFILK = FILNU
IF (T .EQ. 0.) THEN
  UFILKM1 = 0.
  GUFILKM1 = 0.
  FILUDKM1 = 0.
  FILUD2KM1 = 0.
ENDIF

GUFILK = -FILKI*GUFILKMI + TWOPIOVTNU*FILK2*(UFILK + UFILKMI)
```

**DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF U-COMPONENT**

```
FILUD = (GUFILK - GUFILKMI)/TSAMP

FILUD2 = - KWD1 * filud2km1 &
          + (kfield1*CFIL/(1. + tau*CFIL)) &
          *(ufilk - ufilm1)
          5-9-97
```

FILUDKM1 = FILUD
FILUD2KM1 = FILUD2
UFILKM1 = UFILK
GVFILKM1 = GUFILK
FILU = GUFILK

**V-COMPONENT**

```
VFILK = FILNV
IF (T .EQ. 0.) THEN
  VFILKM1 = 0.
  VFILKM2 = 0.
  GVFILKM1 = 0.
  GVFILKM2 = 0.
  FILVDM1 = 0.
  FILVD2KM1 = 0.
  FILVD2KM2 = 0.
ENDIF
```

91
**DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF V-COMPONENT**

**eq.27**

\[
\text{FILVD} = \frac{\text{GVFILK} - \text{GVFILKMI}}{\text{TSAMP}}
\]

**eq.29**

\[
\text{FILVD} = -2.0 \times KWDI \times \text{FILVDKMI}
\]

\[
+ KWDI^2 \times \text{FILVD2KM2}
\]

\[
+ kwd2 \times (1.0 + \text{CTWSR3}) \times \text{VFILK}
\]

\[
- (2.0 \times \text{CTWSR3}) \times \text{VFILKMI} - (1.0 - \text{CTWSR3}) \times \text{VFILKM2}
\]

**5-12-97**

**GVFILKM2 = GVFILKMI**

**GVFILKMI = GVFILK**

**VFILKM2 = VFILKMI**

**VFILKMI = VFILK**

**FILVD2KM2 = FILVD2KMI**

**FILVD2KMI = FILVD2**

**FILVDKMI = FILVD**

**FILV = GVFILK**

**W-COMPONENT**

**eq.27**

\[
\text{FILWD} = \frac{\text{GWFILK} - \text{GWFILKMI}}{\text{TSAMP}}
\]

**eq.20**

\[
\text{WFILK} = \text{FILNW}
\]

**IF (T .EQ. 0.0) THEN**

\[
\text{WFILKM1} = 0.0
\]

\[
\text{WFILKM2} = 0.0
\]

\[
\text{GWFILKM1} = 0.0
\]

\[
\text{GWFILKM2} = 0.0
\]

\[
\text{FILWD2KM1} = 0.0
\]

\[
\text{FILWD2KM2} = 0.0
\]

**ENDIF**

\[
\text{FILWKM1} = \text{GWFILKM1}
\]

**eq.20**

\[
\text{GWFILK} = - \text{FILK4} \times \text{GWFILKMI} - \text{FILK5} \times \text{GWFILKM2}
\]

\[
+ \text{TWOPIOVTNU} \times \text{FILK9} \times (\text{FILK6} \times \text{WFILK} + \text{FILK7} \times \text{WFILKM1} &
\]

\[
+ \text{FILK8} \times \text{WFILKM2})
\]

**DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF W-COMPONENT**

**eq.27**

\[
\text{FILWD} = \frac{\text{GWFILK} - \text{GWFILKMI}}{\text{TSAMP}}
\]
FILWD2 = -2.*KWDI*FILWD2KMI - KWDI**2*FILWD2KM2 + kwd2*((I.+CTWSR3)*WFILK (2.*CTWSR3)*WFILKMI -(I.-CTWSR3)*WFILKM2) !5-12-97

GWFILKM2 = GWFILKMI
GWFILKMI = GWFILK
WFILKM2 = WFILKMI
WFILKMI = WFILK
FILWD2KM2 = FILWD2KMI
FILWD2KMI = FILWD2
FILWDKMI = FILWD
FILW = GWFILK

P, Q, R COMPONENT ADDED 5-15-95 JCY

P-COMPONENT

PFILK = FILNP
IF (T .EQ. 0.) THEN
PFILKMI = 0.
ENDIF
GPFLIK = -PKI*GPFLKMI + PK2*(PFILK + PFILKMI) ! eq. 21
PFILKMI = PFILK ! 5-2-97
GPFLIKMI = GPFLIK ! 5-2-97

Q-COMPONENT

IF (T .EQ. 0.) THEN
FILOQM1 = 0.
ENDIF
FILO = -QKI*FILOQM1 + QK2*(FILW - FILWKMI) ! eq. 23
FILOQM1 = FILW

R-COMPONENT

IF (T .EQ. 0.) THEN
FILRKMI = 0.
ENDIF
FILR = -RKI*FILRKMI + RK2*(FILV - FILVKMI) ! eq. 25
FILRKMI = FILW

ALPHA, BETA COMPONENTS ADDED 5-7-97

FILA = FILW/VTOT ! ALPHA COMPONENT
FILB = FILV/VTOT ! BETA COMPONENT

FILAD = FILWD/VTOT
FILBD = FILVD/VTOT

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! ALPHA , BETA COMPONENTS ALTERNATES ADDED 6-19-97
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FILAD2 = FILWD2/VTOT
FILBD2 = FILVD2/VTOT

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! MIL STD IMPLEMENTATION OF CALCS OF u,v,w,p,q,r-COMPONENT
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
IF (T .EQ. 0.) THEN
MILUKMI = 0.
MILVKMI = 0.
MILWKMI = 0.
MILPKMI = 0.
MILQKMI = 0.
MILRKMI = 0.
ENDIF
MILUK = milk1*MILUKM1 + sigu*milk3*ufilk ! eq.30
MILVK = milk2*MILVKM1 + sigv*milk4*vfilk ! eq.31
MILWK = milk2*MILWKM1 + sigw*milk4*wfilk ! eq.32
MILPK = milk5*MILPKM1 + sigp*milk6*pfilk ! eq.33
MILQK = milk7*MILQKMI + piov4b*(milwk-milwkml) ! eq.34
MILRK = milk8*MILRKMI + piov3b*(milvk-milvkml) ! eq.35

! Calc derivatives 9-23-97
MILUDK = (MILUK - MILUKM1)/TSAMP
MILVDK = (MILVK - MILVKM1)/TSAMP
MILWDK = (MILWK - MILWKM1)/TSAMP

! save past values
MILUKM1 = MILUK
MILVKM1 = MILVK
MILWKM1 = MILWK
MILPKM1 = MILPK
MILQKMI = MILQK
MILRKMI = MILRK

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
END ! GUST DISCRETE

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! DISCRETE DISCRMS
DISCRETE TO COMPUTE RMS OF TURBULENCE

INTERVAL TRMS = 0.0125
SAMPS = SAMPS + 1.

DRMS(DRMSUWG, DMNUWG, UWG, SAMPS)
DRMS(DRMSVWG, DMNVWG, VWG, SAMPS)
DRMS(DRMSWWG, DMNWWG, WWG, SAMPS)

DRMS(DRMSUWG, DMNUWG, UDWG, SAMPS)
DRMS(DRMSVWG, DMNVWG, VDWG, SAMPS)
DRMS(DRMSWWG, DMNWWG, WDWG, SAMPS)

MIL STD calcs, u,v,w,p,q,r extra if needed

DRMS(DRMSMILUK, DMNMILUK, MILUK, SAMPS)
DRMS(DRMSMILVK, DMNMILVK, MILVK, SAMPS)
DRMS(DRMSMILWK, DMNMILWK, MILWK, SAMPS)
DRMS(DRMSMILPK, DMNMILPK, MILPK, SAMPS)
DRMS(DRMSMILQK, DMNMILQK, MILQK, SAMPS)
DRMS(DRMSMILRK, DMNMILRK, MILRK, SAMPS)

END ! of DISCRMS DISCRETE

Time History Plots

Numerous runs were made with the modified HARV aircraft simulation to evaluate the performance of the implemented turbulence models. The 20-second runs were made with five trim cases. Some runs set the PQRFLG variable to .True. to use effects of p, q, and r gust effects, while other runs had a value of .False. for the PQRFLG variable. All runs used the same random seed for noise generation. Time history plots of comparisons of the continuous, Tustin, and MIL STD models for each trim case are presented below in figures 15 through 59. Attitude and air data variables are also plotted. The five trim cases plotted were for \( \alpha \) values of 5°, 25°, 35°, 55°, and 60°. Figures 15 through 23 are the time histories for the \( \alpha = 5^\circ \) trim case. Figures 24 through 32 are the time histories for the \( \alpha = 25^\circ \) trim case. Figures 33 through 41 are the time histories for the \( \alpha = 35^\circ \) trim case. Figures 42 through 50 are the time histories for the \( \alpha = 55^\circ \) trim case. Figures 51 through 59 are the time histories for the \( \alpha = 60^\circ \) trim case.
Figure 15. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 5^\circ$, seed no. 1.
Figure 16. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 5^\circ$, seed no. 1.
Figure 17. Comparison of continuous and Tustin air data and accelerations for $\alpha = 5^\circ$, seed no. 1.
Figure 18. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 5^\circ$, seed no. 1.
Figure 19. Comparison of continuous and Tustin air data and accelerations for $\alpha = 5^\circ$, seed no. 1.
Figure 20. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 5^\circ$, seed no. 1.
Figure 21. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 5^\circ$, seed no. 1.
Figure 22. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 5^\circ$, seed no. 1.
Figure 23. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 5^\circ$, seed no. 1.
Figure 24. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 25^\circ$, seed no. 1.
Figure 25. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 25^\circ$, seed no. 1.
Figure 26. Comparison of continuous and Tustin air data and accelerations for $\alpha = 25^\circ$, seed no. 1.
Figure 27. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 25^\circ$, seed no. 1.
Figure 28. Comparison of continuous and Tustin air data and accelerations for $\alpha = 25^\circ$, seed no. 1.
Figure 29. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 25^\circ$, seed no. 1.
Figure 30. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 25^\circ$, seed no. 1.
Figure 31. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 25^\circ$, seed no. 1.
Figure 32. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 25^\circ$, seed no. 1.
Figure 33. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 35^\circ$, seed no. 1.
Figure 34. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 35^\circ$, seed no. 1.
Figure 35. Comparison of continuous and Tustin air data and accelerations for $\alpha = 35^\circ$, seed no. 1.
Figure 36. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 35^\circ$, seed no. 1.
Figure 37. Comparison of continuous and Tustin air data and accelerations for \( \alpha = 35^\circ \), seed no. 1.
Figure 38. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 35^\circ$, seed no. 1.
Figure 39. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for \( \alpha = 35^\circ \), seed no. 1.
Figure 40. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 35^\circ$, seed no. 1.
Figure 41. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 35^\circ$, seed no. 1.
Figure 42. Comparison of continuous, Tustin, and MIL STD model turbulence for \( \alpha = 55^\circ \), seed no. 1.
Figure 43. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 55^\circ$, seed no. 1
Figure 44. Comparison of continuous and Tustin air data and accelerations for $\alpha = 55^\circ$, seed no. 1.
Figure 45. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 55^\circ$, seed no. 1.
Figure 46. Comparison of continuous and Tustin air data and accelerations for $\alpha = 55^\circ$, seed no. 1.
Figure 47. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 55^\circ$, seed no. 1.
Figure 48. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 55^\circ$, seed no. 1.
Figure 49. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 55^\circ$, seed no. 1.
Figure 50. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 55^\circ$, seed no. 1.
Figure 51. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 60^\circ$, seed no. 1.
Figure 52. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 60^\circ$, seed no. 1.
Figure 53. Comparison of continuous and Tustin air data and accelerations for $\alpha = 60^\circ$, seed no. 1.
Figure 54. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 60^\circ$, seed no. 1.
Figure 55. Comparison of continuous and Tustin air data and accelerations for $\alpha = 60^\circ$, seed no. 1.
Figure 56. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 60^\circ$, seed no. 1.
Figure 57. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 60^\circ$, seed no. 1.
Figure 58. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 60^\circ$, seed no. 1.
Figure 59. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 60^\circ$, seed no. 1.
References


Implementation and Testing of Turbulence Models for the F18-HARV Simulation

Jessie C. Yeager

Lockheed Martin Engineering and Sciences
Mail Stop 371
Hampton, VA 23681-0001

National Aeronautics and Space Administration
NASA Langley Research Center
Hampton, VA 23681-2199

Language Technical Monitor: W. Thomas Bundick

Unclassified—Unlimited
Subject Category 08 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

This report presents three methods of implementing the Dryden power spectral density model for atmospheric turbulence. Included are the equations which define the three methods and computer source code written in Advanced Continuous Simulation Language to implement the equations. Time-history plots and sample statistics of simulated turbulence results from executing the code in a test program are also presented. Power spectral densities were computed for sample sequences of turbulence and are plotted for comparison with the Dryden spectra. The three model implementations were installed in a nonlinear six-degree-of-freedom simulation of the High Alpha Research Vehicle airplane. Aircraft simulation responses to turbulence generated with the three implementations are presented as plots.