NASA/CR-1998-206937

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

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

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-96014

March 1998

Table of Contents

(Intentional blank page)

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	Intermediate variable defined by equation (44); used in equation (43)
a_i, a_j	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	Intermediate variable defined by equation (44)
b_W	Reference wing span
C_{BL}	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}	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	Intermediate variable defined by equation (44)
d	Intermediate variable defined by equation (44)
e	Intermediate variable defined by equation (44)
f	Intermediate variable defined by equation (44)
$G_{di}(\omega)$	Power spectral density of the i-th component of turbulence, discrete model
$\left.\begin{matrix} H_i(z) \\ H_i(s) \end{matrix}\right\}$	Transfer function used to compute the i-th component of turbulence in the z- or the s-domain

$\tilde{H}_i\left(e^{j\omega T_v}\right)$	Transfer function used to compute the frequency response of $H_i\left(z = e^{j\omega T_V}\right)$
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)
$\hat{m}_{\xi_i}^{(j)}$	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)
$R_{ii}(\tau)$	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
$v_j(z)$	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 $\xi_i(k)$
π	Pi (= arc cos(-1))
$\Phi_i(\omega)$	Single-sided power spectral density of i-th component of turbulence, continuous model

τ_i	Time constant of the i-th component of turbulence
σ_i	Desired root-mean-square value of the i-th component of turbulence
$\hat{\sigma}_{\xi_i}^{(j)}$	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 $\left(=\sqrt{E\left\{\left(\hat{\sigma}_i - E\{\hat{\sigma}_i\}\right)^2\right\}}\right)$
$\Sigma(\hat{m}_i)$	Standard deviation of the sample mean of i $\left(=\sqrt{E\left\{\left(\hat{m}_i - E\{\hat{m}_i\}\right)^2\right\}}\right)$
ω	Angular frequency
ω_w	Reciprocal of time constant for the w-component of turbulence ($= 1/\tau_w$)
ω_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
$\mathrm{F}^{-1}\{\cdot\}$	Denotes inverse Fourier transform of argument
$\sum_{k=1}^{N}[\cdot]$	Denotes summation of argument over k from 1 to N
$\lvert\cdot\rvert$	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 $\nu_u, \nu_v, \nu_w, \nu_p$ 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.

$$\ddot{x}_w = -2\tau_w^{-1}\dot{x}_w - \tau_w^{-2}x_w + \sigma_w \tau_w^{-3/2} T_V^{-1/2} \nu_w \tag{1}$$

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

$$\xi_w = x_w + \tau_w \sqrt{3}\dot{x}_w \text{ and } \dot{\xi}_w = \dot{x}_w + \tau_w \sqrt{3}\ddot{x}_w \tag{2}$$

where

$$\tau_w = \frac{L_w}{V} \tag{3}$$

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).

$$\tau_v = \frac{L_v}{V} \tag{4}$$

The differential equation defining the u-component is

$$\dot{\xi}_u = -\frac{1}{\tau_u}\xi_u + \sigma_u\sqrt{\frac{2}{\tau_u T_v}}v_u \tag{5}$$

where
$$\tau_u = \frac{L_u}{V} \tag{6}$$

Similarly, the p-component equations are

$$\dot{\xi}_p = -\frac{1}{\tau_p}\xi_p + \sigma_p\sqrt{\frac{2}{\tau_p T_v}}v_p \tag{7}$$

where
$$\sigma_p = \frac{1.9}{\sqrt{L_w b_w}}\sigma_w \tag{8}$$

$$\tau_p = \frac{L_p}{V} \tag{9}$$

$$L_p = \frac{\sqrt{L_w b_w}}{2.6} \tag{10}$$

The differential equations defining the q-component are

$$\left.\begin{aligned} \dot{x}_q &= -\frac{V\pi}{4b_w}x_q + \frac{\pi}{4b_w}\xi_w \\ \xi_q &= \dot{x}_q \\ &= -\frac{V\pi}{4b_w}x_q + \frac{\pi}{4b_w}\xi_w \end{aligned}\right\} \tag{11}$$

or

$$\dot{\xi}_q = -\frac{V\pi}{4b_w}\xi_q + \frac{\pi}{4b_w}\dot{\xi}_w \tag{12}$$

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

$$\tau_q = \frac{4b_w}{\pi V} \tag{13}$$

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

$$\sigma_q \approx \sqrt{\frac{\pi}{2L_w b_w}}\sigma_w \tag{14}$$

Similarly, the r-component equation is

$$\dot{\xi}_r = -\frac{V\pi}{3b_w}\xi_r + \frac{\pi}{3b_w}\dot{\xi}_v \tag{15}$$

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

$$\tau_r = \frac{3b_w}{\pi V} \tag{16}$$

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

$$\sigma_r \approx \sqrt{\frac{2\pi}{3L_w b_w}}\,\sigma_w \tag{17}$$

Tustin Model

The Tustin model can be implemented with recursive difference equations to compute components of turbulence. The definitions of *tau* 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*** 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

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

where C_{BL} is given by

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

The recursive difference equation to compute the w-component is

$$\begin{aligned}\xi_w(k) = &-\frac{2\left(\omega_w^2 - C_{BL}^2\right)}{\left(\omega_w + C_{BL}\right)^2}\xi_w(k-1) - \frac{\left(\omega_w - C_{BL}\right)^2}{\left(\omega_w + C_{BL}\right)^2}\xi_w(k-2)\\ &+ \frac{\sigma_w\sqrt{\frac{3\omega_w}{T_v}}}{\left(\omega_w + C_{BL}\right)^2}\left[\left(C_{BL} + \frac{\omega_w}{\sqrt{3}}\right)v_w(k) + \frac{2\omega_w}{\sqrt{3}}v_w(k-1) + \left(\frac{\omega_w}{\sqrt{3}} - C_{BL}\right)v_w(k-2)\right]\end{aligned} \tag{20}$$

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

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

$$\xi_p(k) = -\left[\frac{1 - C_{BL_p}\tau_p}{1 + C_{BL_p}\tau_p}\right]\xi_p(k-1) + \left[\frac{\sigma_p\sqrt{\frac{2\tau_p}{T_v}}}{1 + C_{BL_p}\tau_p}\right]\left[v_p(k) + v_p(k-1)\right] \tag{21}$$

where

$$C_{BL_p} = \frac{1}{\tau_p}\cot\left(\frac{T_v}{2\tau_p}\right) \tag{22}$$

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

$$\begin{aligned}\xi_q(k) = &-\left(\frac{1 - 4b_w C_{BL_q}/\pi V}{1 + 4b_w C_{BL_q}/\pi V}\right)\xi_q(k-1) \\ &+ \left(\frac{C_{BL_q}}{V\left(1 + 4b_w C_{BL_q}/\pi V\right)}\right)\left[\xi_w(k) - \xi_w(k-1)\right]\end{aligned} \tag{23}$$

where

$$\begin{aligned}C_{BL_q} &= \frac{1}{\tau_q}\cot\left(\frac{T_v}{2\tau_q}\right) \\ &= \frac{\pi V}{4b_w}\cot\left(\frac{\pi V T_v}{8b_w\tau_q}\right)\end{aligned} \tag{24}$$

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

$$\begin{aligned}\xi_r(k) = &-\left(\frac{1 - 3b_w C_{BL_r}/\pi V}{1 + 3b_w C_{BL_r}/\pi V}\right)\xi_r(k-1) \\ &+ \left(\frac{C_{BL_r}}{V\left(1 + 3b_w C_{BL_r}/\pi V\right)}\right)\left[\xi_v(k) - \xi_v(k-1)\right]\end{aligned} \tag{25}$$

where

$$\begin{aligned} C_{BL_r} &= \frac{1}{\tau_r}\cot\left(\frac{T_V}{2\tau_r}\right) \\ &= \frac{\pi V}{3b_w}\cot\left(\frac{\pi V T_V}{6b_w\tau_q}\right) \end{aligned} \tag{26}$$

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

$$\dot{\xi}(k) = \frac{\xi(k) - \xi(k-1)}{T_V} \tag{27}$$

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

$$\dot{\xi}_u(k) = -\frac{1-\tau_u C_{BL}}{1+\tau_u C_{BL}}\dot{\xi}_u(k-1) + \sigma_u\sqrt{\frac{2\tau_u}{T_V}}\frac{C_{BL}}{1+\tau_u C_{BL}}\{v_u(k) - v_u(k-1)\} \tag{28}$$

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

$$\begin{aligned} \dot{\xi}_w(k) = -2\left(\frac{1-\tau_w C_{BL}}{1+\tau_w C_{BL}}\right)\dot{\xi}_w(k-1) - \left(\frac{1-\tau_w C_{BL}}{1+\tau_w C_{BL}}\right)^2\dot{\xi}_w(k-2) + \frac{\sigma_w\sqrt{\frac{\tau_w}{T_V}}C_{BL}}{(1+\tau_w C_{BL})^2} \\ \times\left\{\left(1+\sqrt{3}\tau_w C_{BL}\right)v_w(k) - 2\sqrt{3}\tau_w C_{BL}v_w(k-1) - \left(1-\sqrt{3}\tau_w C_{BL}\right)v_w(k-2)\right\} \end{aligned} \tag{29}$$

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.

$$\xi_u(k) = \left(1 - \frac{T_V}{\tau_u}\right)\xi_u(k-1) + \sigma_u\sqrt{\frac{2T_V}{\tau_u}}\,v_u(k) \tag{30}$$

$$\xi_v(k) = \left(1 - \frac{2T_V}{\tau_v}\right)\xi_v(k-1) + \sigma_v\sqrt{\frac{4T_V}{\tau_v}}\,v_v(k) \tag{31}$$

$$\xi_w(k) = \left(1 - \frac{2T_V}{\tau_w}\right)\xi_w(k-1) + \sigma_w\sqrt{\frac{4T_V}{\tau_w}}\,v_w(k) \tag{32}$$

$$\xi_p(k) = \left(1 - \frac{T_V}{\tau_p}\right)\xi_p(k-1) + \sigma_p\sqrt{\frac{2T_V}{\tau_p}}\,v_p(k) \tag{33}$$

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

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

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 1. Correlation between Algebraic Symbol and ACSL/FORTRAN Variable

Symbol	Turbulence Model			Equation no.
	Continuous	Tustin	MIL STD	
v_p	FILNP			
$v_p(k)$		FILNP	FILNP	
v_u	FILNU			
$v_u(k)$		FILNU	FILNU	
v_v	FILNV			
$v_v(k)$		FILNV	FILNV	
v_w	FILNW			
$v_w(k)$		FILNW	FILNW	
ξ_p	TURBP			7
$\xi_p(k)$		FILP	MILPK	21, 33
ξ_q	TURBQ			11
$\xi_q(k)$		FILQ	MILQK	23, 34
ξ_r	TURBR			15
$\xi_r(k)$		FILR	MILRK	25, 35
ξ_u	TURBU			5
$\xi_u(k)$		FILU	MILUK	18, 30

Table 1. Concluded

Symbol	Turbulence Model			Equation no.
	Continuous	Tustin	MIL STD	
ξ_v	TURBV			2
$\xi_v(k)$		FILV	MILVK	20, 31
ξ_w	TURBW			2
$\xi_w(k)$		FILW	MILWK	20, 32
σ_p	SIGP	SIGP	SIGP	8
σ_q	SIGQ	SIGQ	SIGQ	14
σ_r	SIGR	SIGR	SIGR	17
σ_u	SIGU (=TURBSIG)	SIGU (=TURBSIG)	SIGU (=TURBSIG)	
σ_v	SIGV (=TURBSIG)	SIGV (=TURBSIG)	SIGV (=TURBSIG)	
σ_w	SIGW(=TURBSIG)	SIGW(=TURBSIG)	SIGW(=TURBSIG)	
τ_p	TAUP	TAUP	TAUP	9
τ_q	TAUQ	TAUQ	TAUQ	13
τ_r	TAUR	TAUR	TAUR	16
τ_u	TAU	TAU	TAU	6
τ_v	TAU	TAU	TAU	4
τ_w	TAU	TAU	TAU	3
b_W	BWING (=32.4)	BWING (=32.4)	BWING (=32.4)	
L_u	TURBL (=1750.)	TURBL (=1750.)	TURBL (=1750.)	
L_v	TURBL (=1750.)	TURBL (=1750.)	TURBL (=1750.)	
L_w	TURBL (=1750.)	TURBL (=1750.)	TURBL (=1750.)	
L_p	LP	LP	LP	10
V	VTOT	VTOT	VTOT	
T_v	TSAMP (=0.0125)	TSAMP (=0.0125)	TSAMP (=0.0125)	

Code

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 RESONSE 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
!
```

```
!*********************************************************************!
MACRO DRMS(RMSX, MNX , X, N )
!*********************************************************************!
!
!    MACRO TO COMPUTE SAMPLE STATISTICS OF RANDOM VARIABLE
!             WTB, JCY 2-24-94
!
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 TO COMPUTE SAMPLE CROSS-CORRELATION OF RANDOM VARIABLE
!             WTB, JCY 6-2-97
!
MACRO REDEFINE SUMXY, SUMX, SUMY, MNX, MNY, NS, XDLY, YDLY, NDLY, I
  REAL XDLY(100)
MACRO RELABEL LDLY
  IF (N .LT. NTAU + 1)  THEN
     NDLY = N
  ELSE
     NDLY = NTAU + 1
  ENDIF
  DO LDLY I = 1, NDLY - 1
     XDLY(NDLY+1-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 SAMPS = 0.
      CONSTANT EPSLON = 1.E-22

      CONSTANT SDNOIS    = 1.,    &
               TSAMP     = .0125, &
               TURBL     = 1750., &
               TURBSIG   = 5.,    &
               TURBU0    = 0.,    &
               TURBXVD0  = 0.,    &
               TURBXV0   = 0.,    &
               TURBXWD0  = 0.,    &
               TURBXW0   = 0.
!
      CONSTANT TURBNU    = 0.,    &
               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)
```

```
!
      SIGQ = SQRT(pi/(2.*TURBL * BWING)) * SIGW     ! eq.14
      SIGR = SQRT(2.*pi/(3.*TURBL * BWING)) * SIGW  ! eq.17
!
      CONSTANT VTOT = 400.                      !   VTOT = VELOCITY, FT/SEC
!
! use turbl= 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 * BWING) / 2.6                      ! eq.10
      TAUP = LP/VTOT                                       ! eq.13

      TURBOMEGA = VTOT/TURBL          ! same as 1./TAU,1./TAUV,1./TAUW
      WP = VTOT/LP                    ! same as 1./TAUP
!
!  CONSTANTS FOR CALCULATIONS OF CONTINUOUS W AND V        ! eq. 1,2
!
      TURBK2U  = TURBSIG*SQRT(2.*TURBOMEGA/TSAMP)
          K2P  = SIGP * SQRT(2. * WP/TSAMP)
      TURBK1VW = 2.*TURBOMEGA
      TURBK2VW = TURBOMEGA*TURBOMEGA
      TURBK3VW = TAU*SQRT(3.)
      TURBK4VW = TURBSIG*SQRT(TURBOMEGA**3/TSAMP)
!
!*******************************************************************!
!  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
!
      KFILD1  = 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
      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)
!
!**********************************************************************!
!
END    ! OF INITIAL !

!**********************************************************************!
   DYNAMIC
!**********************************************************************!
!**********************************************************************!
   DERIVATIVE
!**********************************************************************!
!
!    CONTINUOUS TURBULENCE MODEL    WTB, JCY 2-24-94 through 11-30-97
!
!**********************************************************************!
!    U - COMPONENT
!**********************************************************************!
!
!   used in eq.5
!
      TURBUD = - TURBOMEGA * TURBU + TURBK2U * FILNU
      TURBU  = INTVC (TURBUD, TURBU0)
!**********************************************************************!
!    V - COMPONENT
!**********************************************************************!
!
!   used in eq.2
!
      TURBXVDD = - TURBK1VW * TURBXVD - TURBK2VW * TURBXV     &
                 + TURBK4VW * FILNV
      TURBXVD  = INTVC (TURBXVDD, TURBXVD0)
      TURBXVDI = TURBXVD
      TURBXV   = INTVC (TURBXVDI, TURBXV0)
      TURBV    = TURBXV + TURBK3VW * TURBXVDI
      TURBVD   = TURBXVDI + TURBK3VW * TURBXVDD
!**********************************************************************!
!    W - COMPONENT
!**********************************************************************!
!
!   used in eq.2
!
      TURBXWDD = - TURBK1VW * TURBXWD - TURBK2VW * TURBXW    &
                 + TURBK4VW * FILNW
      TURBXWD   = INTVC (TURBXWDD, TURBXWD0)
      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
!**********************************************************************!
!    P - COMPONENT
!**********************************************************************!
!
!   used in eq.7
!
      TURBPD = - WP * TURBP + K2P * FILNP
      TURBP  = INTVC (TURBPD, TURBP0)
!**********************************************************************!
!    Q - COMPONENT
!**********************************************************************!
!
!   used in eq.12
!
      TURBQD = - TURBQ / TAUQ + TURBWD / (TAUQ * VTOT)
      TURBQ  = INTVC (TURBQD, TURBQ0)
!**********************************************************************!
!    R - COMPONENT
!**********************************************************************!
!
!   used in eq.15
!
      TURBRD = - TURBR / TAUR + TURBVD / (TAUR * VTOT)
      TURBR  = INTVC (TURBRD, TURBR0)
!**********************************************************************!
!
END  !OF DERIVATIVE!
!
!**********************************************************************!
!
!**********************************************************************!
!
!   DISCRETE  BLOCK
!
!**********************************************************************!
!
   DISCRETE GUST
!**********************************************************************!
!
```

```
         INTERVAL TSGAUS = 0.0125
!**********************************************************************!
!    GAUSSIAN RANDOM PROCESS
!
!**********************************************************************!
         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
!
!**********************************************************************!
!    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
!
!    eq.18
!
         GUFILK = -FILK1*GUFILKM1 + TWOPIOVTNU*FILK2*(UFILK + UFILKM1)

!**********************************************************************!
! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF U-COMPONENT
!**********************************************************************!
!
!    eq.27
!
            FILUD = (GUFILK - GUFILKM1)/TSAMP
!
!    eq.28
!
            FILUD2 = - KWD1 * filud2km1                    &
                     + (kfild1*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.
            FILVD2KM1 = 0.
            FILVD2KM2 = 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*FILVD2KM1                        &
                  - KWD1**2*FILVD2KM2                        &
                  + kwd2*((1.+CTWSR3)*VFILK                  &
                  - (2.*CTWSR3)*VFILKM1 -(1.-CTWSR3)*VFILKM2)   ! 5-12-97
!
            GVFILKM2 = GVFILKM1
            GVFILKM1 = GVFILK
            VFILKM2 = VFILKM1
            VFILKM1 = VFILK
            FILVD2KM2 = FILVD2KM1
            FILVD2KM1 = FILVD2
         FILVDKM1 = FILVD
         FILV = GVFILK
!**********************************************************************!
!    W-COMPONENT
!**********************************************************************!
         WFILK = FILNW
         IF (T .EQ. 0.)  THEN
            WFILKM1 = 0.
            WFILKM2 = 0.
            GWFILKM1 = 0.
            GWFILKM2 = 0.
            FILWDKM1 = 0.
            FILWD2KM1 = 0.
```

```
              FILWD2KM2 = 0.
           ENDIF
              FILWKM1 = GWFILKM1
!
!    eq.20
!
              GWFILK = - FILK4*GWFILKM1 - FILK5*GWFILKM2              &
                       + TWOPIOVTNU*FILK9*(FILK6*WFILK + FILK7*WFILKM1 &
                       + FILK8*WFILKM2)
!**********************************************************************!
! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF W-COMPONENT
!**********************************************************************!
!
!    eq.27
!
           FILWD = (GWFILK - GWFILKM1)/TSAMP

!
!    eq.29
!
           FILWD2 =  -2.*KWD1*FILWD2KM1                         &
                    - KWD1**2*FILWD2KM2                         &
                    + kwd2*((1.+CTWSR3)*WFILK                   &
                    - (2.*CTWSR3)*WFILKM1 -(1.-CTWSR3)*WFILKM2)  !5-12-97
!
              GWFILKM2 = GWFILKM1
              GWFILKM1 = GWFILK
              WFILKM2 = WFILKM1
              WFILKM1 = WFILK
              FILWD2KM2 = FILWD2KM1
              FILWD2KM1 = FILWD2
           FILWDKM1 = FILWD
           FILW = GWFILK
!**********************************************************************!
!  P, Q, R COMPONENT ADDED  5-15-95 JCY
!**********************************************************************!
!    P-COMPONENT
!**********************************************************************!
           PFILK = FILNP
           IF (T .EQ. 0.)  THEN
              PFILKM1 = 0.
              GPFILKM1 = 0.
           ENDIF
              GPFILK  = -PK1*GPFILKM1 + PK2*(PFILK + PFILKM1)   ! eq. 21
              PFILKM1 = PFILK                                   ! 5-2-97
              GPFILKM1 = GPFILK                                 ! 5-2-97
           FILP = GPFILK
!**********************************************************************!
!    Q-COMPONENT
!**********************************************************************!
           IF (T .EQ. 0.)  THEN
              FILQKM1 = 0.
           ENDIF
              FILQ    = -QK1*FILQKM1 + QK2*(FILW - FILWKM1)     ! eq. 23
```

```
            FILQKM1 = FILQ
!
!**********************************************************************!
!    R-COMPONENT
!**********************************************************************!
         IF (T .EQ. 0.)  THEN
            FILRKM1 = 0.
         ENDIF
            FILR    = -RK1*FILRKM1 + RK2*(FILV - FILVKM1)      ! 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
!                                                              8-28-97
!**********************************************************************!
         IF (T .EQ. 0.)  THEN
            MILUKM1 = 0.
            MILVKM1 = 0.
            MILWKM1 = 0.
            MILPKM1 = 0.
            MILQKM1 = 0.
            MILRKM1 = 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*MILQKM1 + piov4b*(milwk-milwkm1)         ! eq.34
      MILRK = milk8*MILRKM1 + piov3b*(milvk-milvkm1)         ! eq.35
!
!   Calc derivatives                                         9-18-97
!
      MILUDK = (MILUK - MILUKM1)/TSAMP
      MILVDK = (MILVK - MILVKM1)/TSAMP
      MILWDK = (MILWK - MILWKM1)/TSAMP
!
!   save past values
!
```

```
            MILUKM1 = MILUK
            MILVKM1 = MILVK
            MILWKM1 = MILWK
            MILPKM1 = MILPK
            MILQKM1 = MILQK
            MILRKM1 = MILRK
!
!**********************************************************************!
!
   END ! GUST DISCRETE
!
!**********************************************************************!
!
   DISCRETE DISCRMS
!
!**********************************************************************!
!     DISCRETE TO COMPUTE RMS OF TURBULENCE
!
         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(DRMSTURBWD, DMNTURBWD, TURBWD , 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(XCORTRV, MXMTRV, TURBR, 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 DISCRMS DISCRETE
!**********************************************************************!
!----------------------------------------------------------------------!
! ------ DISCRETE TO SAVE DATA TO AN ARRAY IN DRYDEN GETDATA FORMAT    !
!----------------------------------------------------------------------!
```

```
!*********************************************************************!
!
DISCRETE ASC
         INTERVAL TSASC = 0.0125
!
       CALL ASCFM(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,                               &
         NUMOUT,    TSTP)
!
    TERMT(T.GE.TSTP)

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

END  !OF DYNAMIC!
!
TERMINAL
!
      CALL WSTATS (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,                       &
                DMNTURBALP,DMNTURBBET,                               &
                DMNTURBALPD,DMNTURBBETD,                             &
                DMNFILA,DMNFILB,DMNFILAD,DMNFILBD,                   &
```

```
                 DMNFILAD2,DMNFILBD2,                              &
                 DRMSMILUK,DMNMILUK,DRMSMILVK,DMNMILVK,            &
                 DRMSMILWK,DMNMILWK,DRMSMILPK,DMNMILPK,            &
                 DRMSMILQK,DMNMILQK,DRMSMILRK,DMNMILRK,            &
                 DRMSMILUDK,DMNMILUDK,DRMSMILVDK,DMNMILVDK,        &
                 DRMSMILWDK,DMNMILWDK)
!
END ! END OF TERMINAL
!
!
END  !OF PROGRAM!
      SUBROUTINE WSTATS (T,
C
C *******  35  +2 = 37 rms   *******************************
C
     *   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,
C
C *******  35 + 2 = 37 means + 12 mils= 49 values
*******************************
C
     *   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,
     *   DMNTURBALP,    DMNTURBBET,
     *   DMNTURBALPD,   DMNTURBBETD,
     *   DMNFILA,       DMNFILB,       DMNFILAD,      DMNFILBD,
     *   DMNFILAD2,     DMNFILBD2,
     *   DRMSMILUK,     DMNMILUK,      DRMSMILVK,     DMNMILVK,
     *   DRMSMILWK,     DMNMILWK,      DRMSMILPK,     DMNMILPK,
     *   DRMSMILQK,     DMNMILQK,      DRMSMILRK,     DMNMILRK,
     *   DRMSMILUDK,    DMNMILUDK,     DRMSMILVDK,    DMNMILVDK,
     *   DRMSMILWDK,    DMNMILWDK)
C
C*****************************************************************
C*  SUBROUTINE WSTAT
```

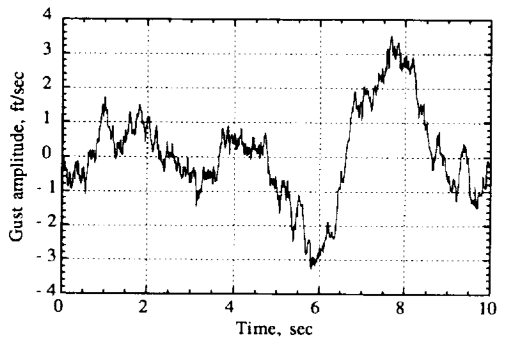
```
C*  WRITES AN OUTPUT FILE IN THE GETDATA FORMAT ASC. to IMPORT TO EXCEL
or KG
C*******************************************************************
c
      DIMENSION XLAB2(93)
      CHARACTER*16 XLAB1, XLAB2
C
      DATA XLAB1 /'names'/
      DATA NSIMCH2 / 93 /
C*******************************************************************
c
Cjcy       INCLUDE 'dpguststats.inc'
c
C*******************************************************************
C
      DATA XLAB2/  'names',
C
c *******  35 rms   ********************************
c
     *    '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',
c
c *******  37 means + 2 rms +12 mil ********************************
c
     *    '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',
     *    'DMNTURBALP', 'DMNTURBBET',
     *    'DMNTURBALPD','DMNTURBBETD',
     *    'DMNFILA',    'DMNFILB',    'DMNFILAD',    'DMNFILBD',
     *    'DRMSFILAD2', 'DRMSFILBD2', 'DMNFILAD2',   'DMNFILBD2',
     *    'DRMSMILUK',  'DMNMILUK',   'DRMSMILVK',
     *    'DMNMILVK',   'DRMSMILWK',  'DMNMILWK',
     *    'DRMSMILPK',  'DMNMILPK',   'DRMSMILQK',
     *    'DMNMILQK',   'DRMSMILRK',  'DMNMILRK',
     *    'DRMSMILUDK', 'DMNMILUDK',  'DRMSMILVDK',
     *    'DMNMILVDK',  'DRMSMILWDK', 'DMNMILWDK' /
C
```

```
C  OPEN A DATA CHANNEL FOR WRITING THE GETDATA ASC FORMAT FILE.
C
      OPEN (33,FILE='KGACSL.ASC',STATUS='UNKNOWN')
C
C  WRITE OUT DATA IN TO FILE FOR USE IN GETDATA.
C
      WRITE(33,'(''format  asc 2   .1'',/,''nChans'',t14,i3)')NSIMCH2-1
      WRITE(33,'(6a13)') XLAB1,(XLAB2(I),I=2,NSIMCH2)
      WRITE(33,'(''data001'')')
C
     WRITE(33,'(6G13.7)') 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,
c *******  37 means + 2  rms + 12 mils ********************************
    *   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,
    *   DMNTURBALP,    DMNTURBBET,
    *   DMNTURBALPD,   DMNTURBBETD,
    *   DMNFILA,       DMNFILB,       DMNFILAD,       DMNFILBD,
    *   DRMSFILAD2,    DRMSFILBD2,    DMNFILAD2,      DMNFILBD2,
    *   DRMSMILUK,     DMNMILUK,      DRMSMILVK,      DMNMILVK,
    *   DRMSMILWK,     DMNMILWK,      DRMSMILPK,      DMNMILPK,
    *   DRMSMILQK,     DMNMILQK,      DRMSMILRK,      DMNMILRK,
    *   DRMSMILUDK,    DMNMILUDK,     DRMSMILVDK,     DMNMILVDK,
    *   DRMSMILWDK,    DMNMILWDK
c
     CLOSE(33)
c
     RETURN
     END
     SUBROUTINE ASCFM(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,
     *   NUMOUT,   TSTP)

C
C*******************************************************************
C*  SUBROUTINE ASCFM
C*  WRITES AN OUTPUT FILE IN THE GETDATA FORMAT ASC.
C*******************************************************************
C
cjcy   IMPLICIT REAL*8 (a-h,o-z)
c
      DIMENSION XLABEL2(47)
      CHARACTER*16 XLABEL1, XLABEL2
C
cjcy was real*8
C
      REAL MILUK, MILVK, MILWK, MILPK, MILQK, MILRK,
     *       MILUDK, MILVDK, MILWDK
C
      DATA XLABEL1 /'names'/
      DATA NSIMCH2 / 47/
c
      DATA XLABEL2/'names',
     *   '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' /

C
C  OPEN A DATA CHANNEL FOR WRITING THE GETDATA ASC FORMAT FILE.
C
      IF((T .EQ. 0.0) ) THEN
       OPEN (3,FILE='ACSL.ASC',STATUS='UNKNOWN')
C
C  WRITE OUT DATA IN TO FILE FOR USE IN GETDATA.
C
       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
```

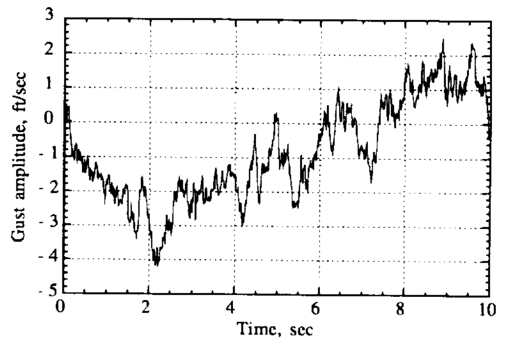
```
C
      IF(T .GE. 0.0 .AND. T .LT. TSTP) THEN
         IF (INTERVA .EQ. NUMOUT) THEN
      WRITE(3,'(6G13.7)') T,
     *   TURBU,     TURBV,     TURBW,     FILU,
     *   FILV,      FILW,      TURBUD,    TURBVD,
     *   TURBWD,    FILUD,     FILVD,     FILWD,
     *   FILNU,     FILNV,     FILNW,     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
C
             INTERVA = 1
         ELSE
             INTERVA = INTERVA + 1
         ENDIF
      ENDIF
      IF(T .GT. TSTP) CLOSE(3)
      RETURN
      END
```

Time History Plots

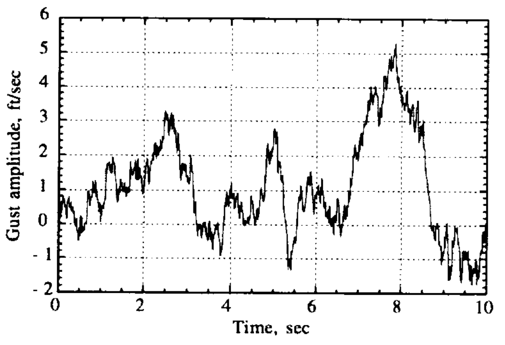
Numerous 100-second runs were made with the GUSTMDL test program to produce turbulence sequences for analysis of the continuous, Tustin, and MIL STD turbulence models. Simulated aircraft velocities of 100 ft/sec and 1000 ft/sec were used. Sample time history plots of the first 10 seconds of these sequences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



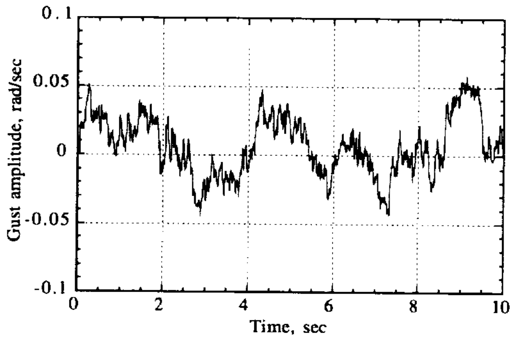
(a).- u-component



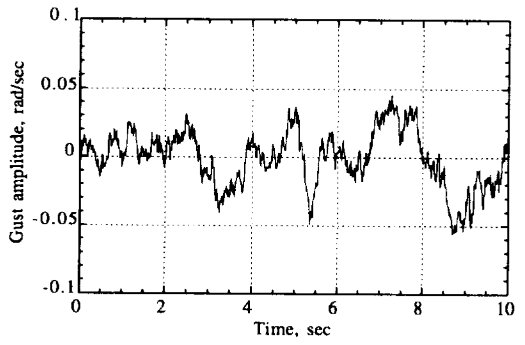
(b).- v-component



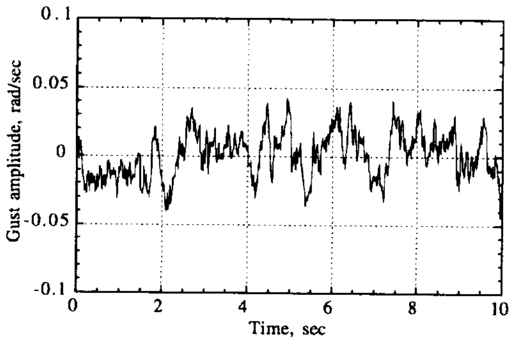
(c).- w-component



(d).- p-component

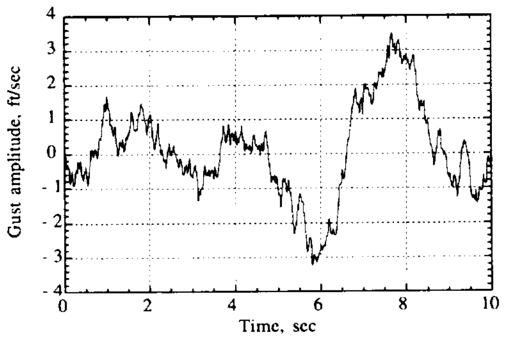


(e).- q-component

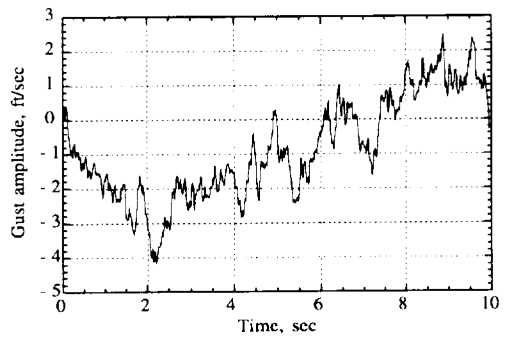


(f).- r-component

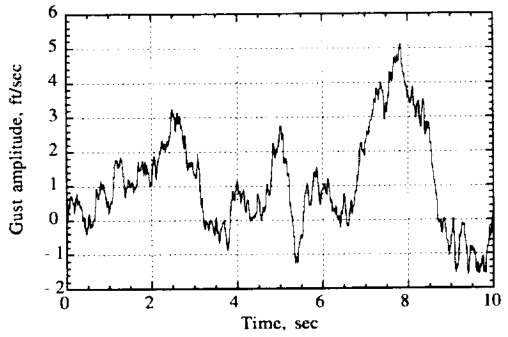
Figure 1. Sample time histories; V =100 ft/sec - continuous model.



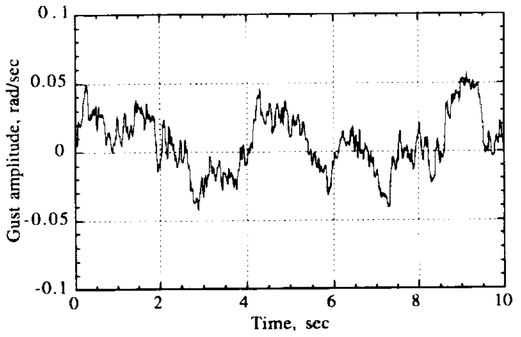
(a).- u-component



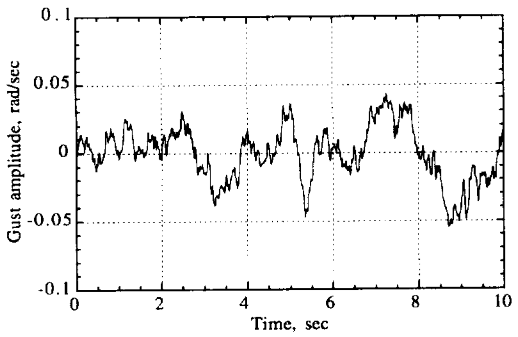
(b).- v-component



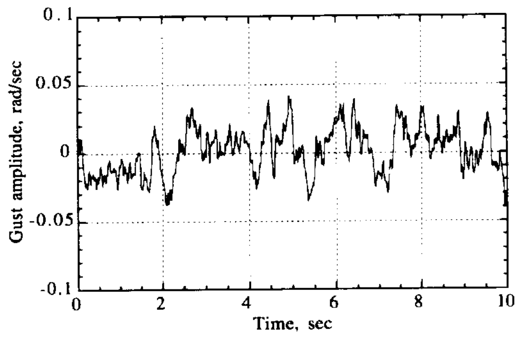
(c).- w-component



(d).- p-component

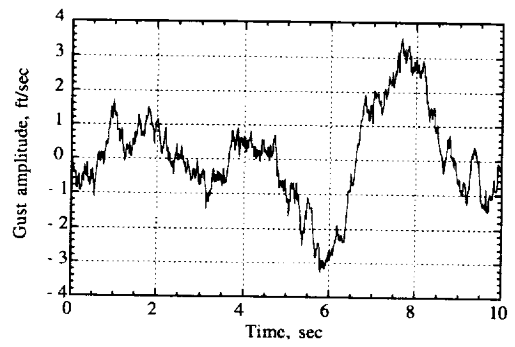


(e).- q-component

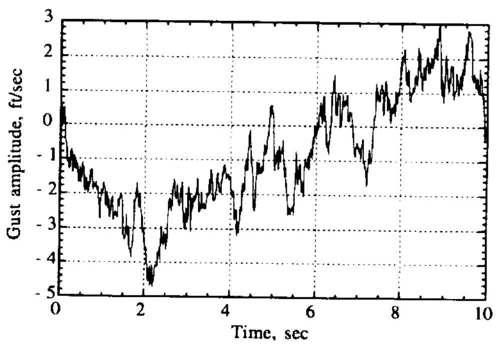


(f).- r-component

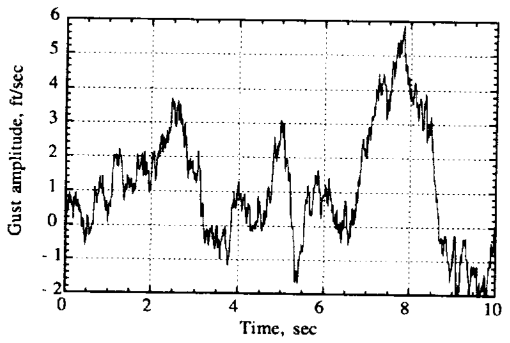
Figure 2. Sample time histories; V =100 ft/sec - Tustin discrete model.



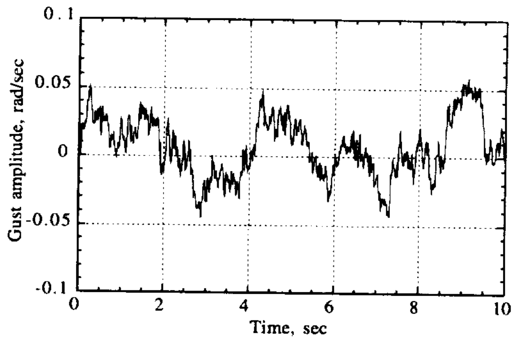
(a).- u-component



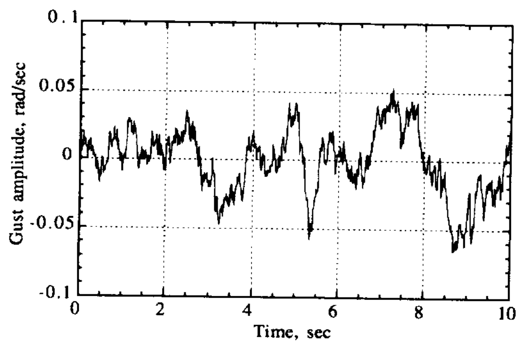
(b).- v-component



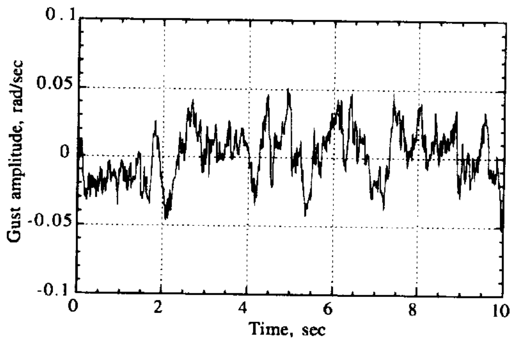
(c).- w-component



(d).- p-component

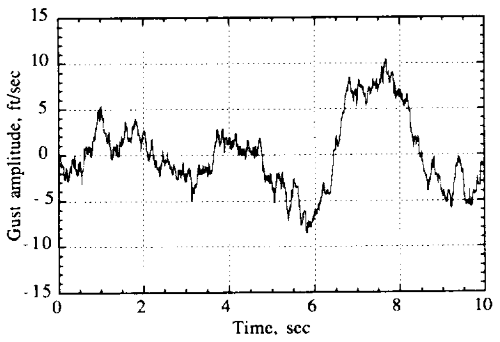


(e).- q-component

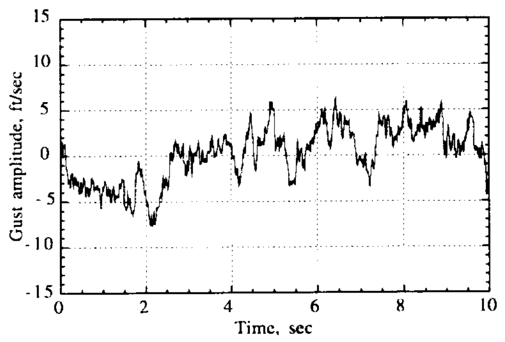


(f).- r-component

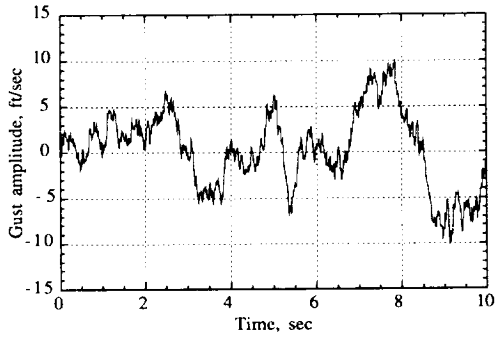
Figure 3. Sample time histories; V =100 ft/sec - MIL STD discrete model.



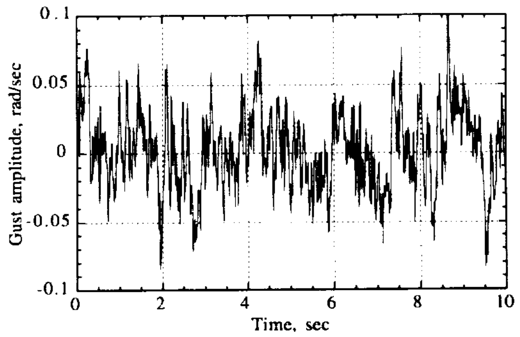
(a).- u-component



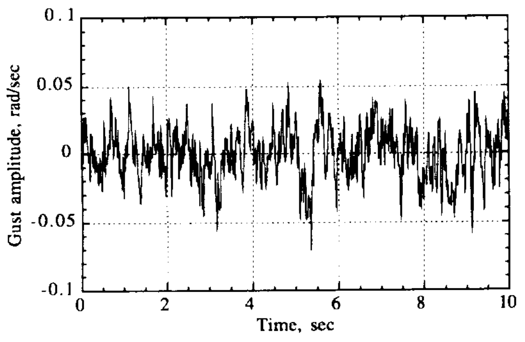
(b).- v-component



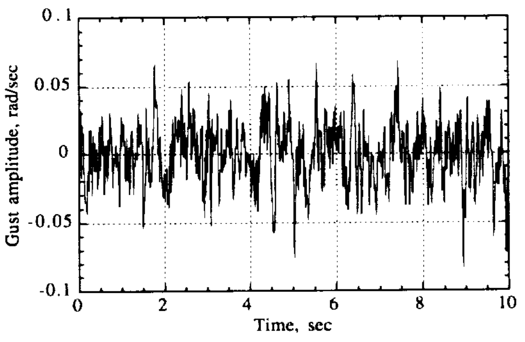
(c).- w-component



(d).- p-component

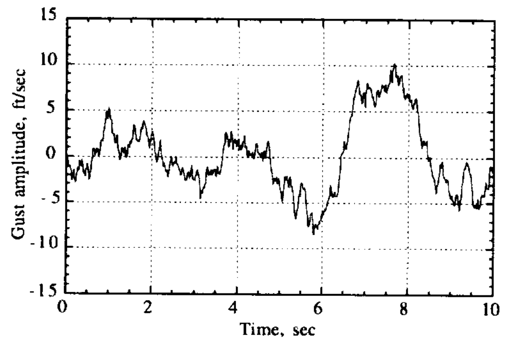


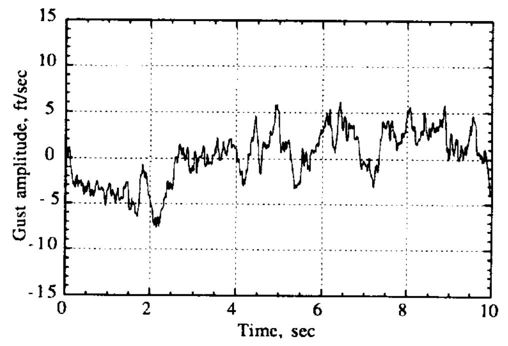
(e).- q-component

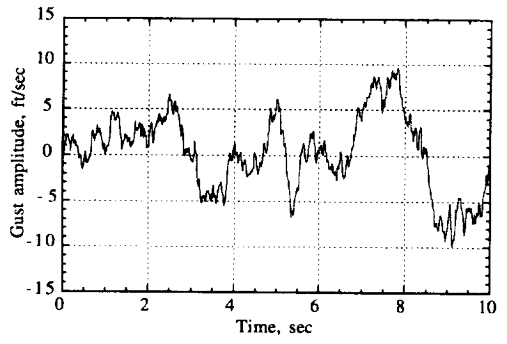


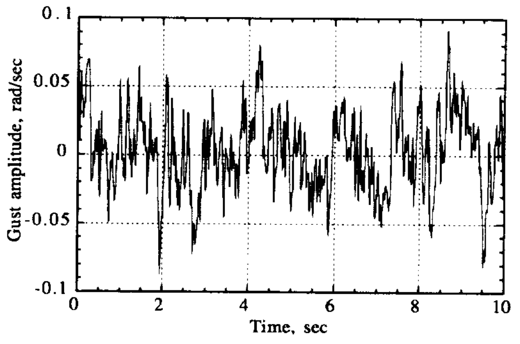
(f).- r-component

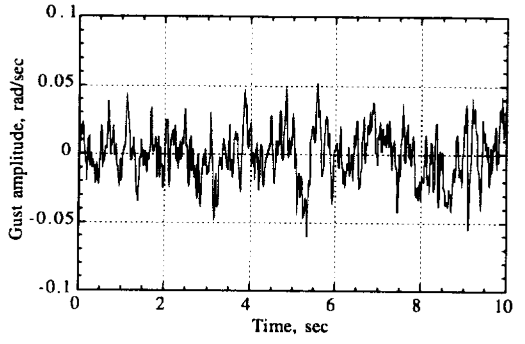
Figure 4. Sample time histories; V =1000 ft/sec - continuous model.


(a).- u-component


(b).- v-component


(c).- w-component


(d).- p-component


(e).- q-component

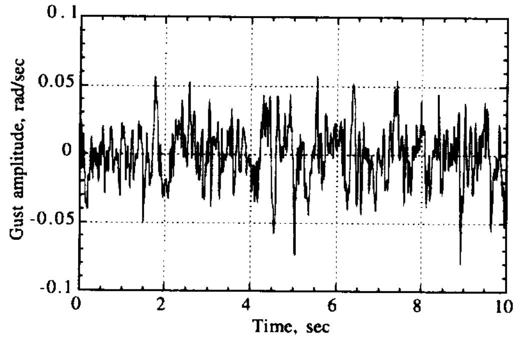
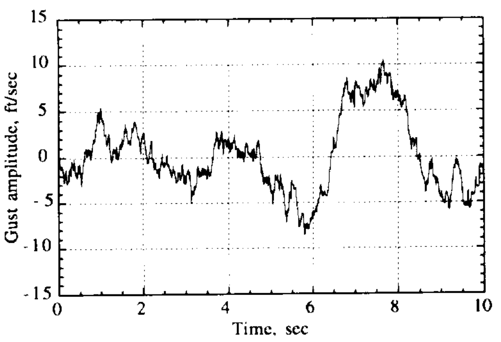
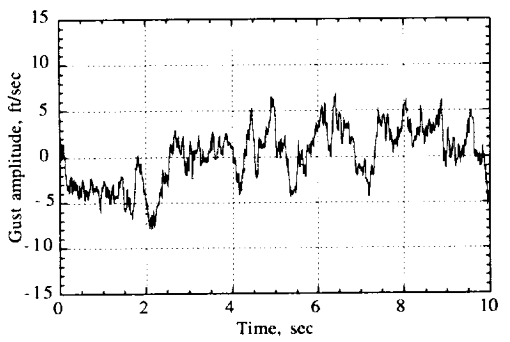

(f).- r-component

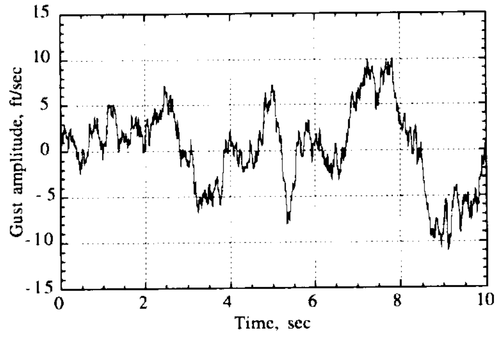
Figure 5. Sample time histories; V =1000 ft/sec - Tustin discrete model.



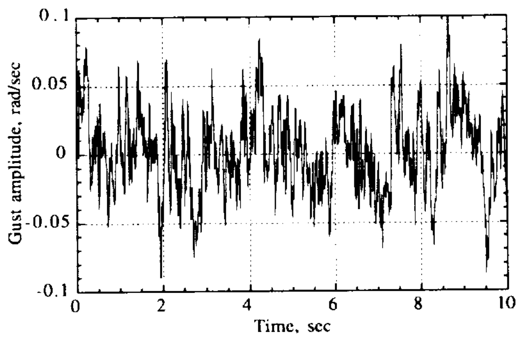
(a).- u-component



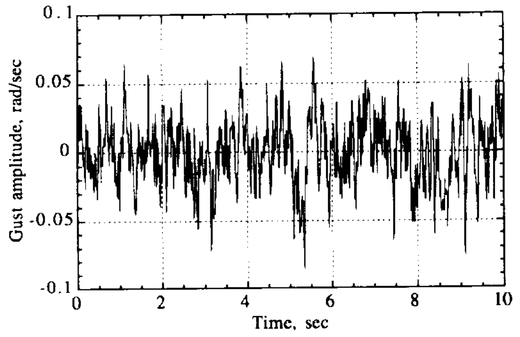
(b).- v-component



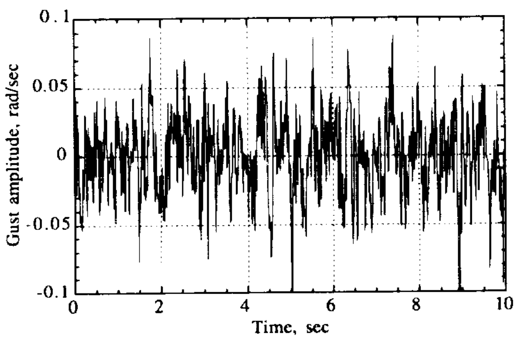
(c).- w-component



(d).- p-component



(e).- q-component



(f).- r-component

Figure 6. Sample time histories; V =1000 ft/sec - MIL STD discrete model.

Power Spectral Densities

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).

$$S_u(\omega) = \frac{\sigma_u^2 \tau_u}{\pi} \frac{1}{1+(\tau_u \omega)^2} \tag{36}$$

$$S_v(\omega) = \frac{1}{2}\tilde{\Phi}_v(\omega) = \frac{\sigma_v^2 \tau_v}{2\pi} \frac{1+3(\tau_v \omega)^2}{\left(1+(\tau_v \omega)^2\right)^2} \tag{37}$$

$$S_w(\omega) = \frac{1}{2}\tilde{\Phi}_w(\omega) = \frac{\sigma_w^2 \tau_w}{2\pi} \frac{1+3(\tau_w \omega)^2}{\left(1+(\tau_w \omega)^2\right)^2} \tag{38}$$

$$S_p(\omega) = \frac{\sigma_p^2 \tau_p}{\pi} \frac{1}{1+(\tau_p \omega)^2} \tag{39}$$

$$S_q(\omega) = \frac{(\omega/V)^2}{1+\left[(4b_w/\pi)(\omega/V)\right]^2} S_w(\omega) \tag{40}$$

$$S_r(\omega) = \frac{(\omega/V)^2}{1+\left[(3b_w/\pi)(\omega/V)\right]^2} S_v(\omega) \tag{41}$$

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

$$G_{du}(\omega) = \frac{T_V}{2\pi}\left|H_u\left(e^{j\omega T_V}\right)\right|^2 = \frac{\sigma_u^2 \omega_u}{\pi} \frac{\left[1+\cos(\omega T_V)\right]}{\left(\omega_u^2 + C_{BL}^2\right) + \left(\omega_u^2 - C_{BL}^2\right)\cos(\omega T_V)} \tag{42}$$

$$G_{di}(\omega) = \frac{T_V}{2\pi}\left|H_i\left(e^{j\omega T_V}\right)\right|^2 \qquad \text{for} \quad i = v, w$$

$$= \frac{K_i^2 \sigma_i^2\left[a^2+b^2+c^2+2(ab+bc)\cos(\omega T_V)+2ac\cos(2\omega T_V)\right]}{d^2+e^2+f^2+2(de+ef)\cos(\omega T_V)+2df\cos(2\omega T_V)} \tag{43}$$

where
$$\left.\begin{aligned} \omega_i &= V/L_i \\ K_i &= \sqrt{\frac{3\omega_i}{2\pi}} \\ a &= C_{BL} + \omega_i \sqrt{3} \\ b &= 2\omega_i/\sqrt{3} \\ c &= \omega_i/\sqrt{3} - C_{BL} \\ d &= (\omega_i + C_{BL})^2 \\ e &= 2(\omega_i^2 - C_{BL}^2) \\ f &= (\omega_i - C_{BL})^2 \end{aligned}\right\} \quad \text{for} \quad i = v, w \tag{44}$$

$$G_{dp}(\omega) = \frac{T_V}{2\pi}\left|H_p\left(e^{j\omega T_V}\right)\right|^2 = \frac{\sigma_p^2\omega_p}{\pi}\frac{\left[1+\cos(\omega T_V)\right]}{\left(\omega_p^2 + C_{BL_p}^2\right) + \left(\omega_p^2 - C_{BL_p}^2\right)\cos(\omega T_V)} \tag{45}$$

$$G_{dq}(\omega) = \left|H_q\left(e^{j\omega T_V}\right)\right|^2 G_{dw}(\omega) \tag{46}$$

where
$$\left|H_q\left(e^{j\omega T_V}\right)\right|^2 = \frac{2C_{BL_q}^2}{V^2}\frac{1-\cos(\omega T_V)}{M_{q+}^2 + M_{q-}^2 + 2M_{q+}M_{q-}\cos(\omega T_V)} \tag{47}$$

and
$$\left.\begin{aligned} M_{q+} &= 1 + \tau_q C_{BL_q} \\ M_{q-} &= 1 - \tau_q C_{BL_q} \end{aligned}\right\} \tag{48}$$

$$G_{dr}(\omega) = \left|H_r\left(e^{j\omega T_V}\right)\right|^2 G_{dv}(\omega) \tag{49}$$

where
$$\left|H_r\left(e^{j\omega T_V}\right)\right|^2 = \frac{2C_{BL_r}^2}{V^2}\frac{1-\cos(\omega T_V)}{M_{r+}^2 + M_{r-}^2 + 2M_{r+}M_{r-}\cos(\omega T_V)} \tag{50}$$

and
$$\left.\begin{aligned} M_{r+} &= 1 + \tau_r C_{BL_r} \\ M_{r-} &= 1 - \tau_r C_{BL_r} \end{aligned}\right\} \tag{51}$$

The theoretical power spectral densities for the MIL STD model were provided by the Government and are listed below as equations (52) through (57).

$$G_{di}(\omega) = \frac{T_v}{2\pi}\left|H_i\left(e^{j\omega T_v}\right)\right|^2$$

$$= \frac{\frac{T_v^2}{\pi\tau_i}\sigma_i^2}{1+(1-a_i)^2 - 2(1-a_i)\cos(\omega T_v)} \qquad \text{for } i = u, v, w, p \qquad (52)$$

where

$$H_i\left(e^{j\omega T_v}\right) = \frac{\sigma_u\sqrt{2a_i}}{1-(1-a_i)e^{-j\omega T_v}}$$

$$= \frac{\sigma_u\sqrt{2a_i}}{\left[1-(1-a_i)\cos(\omega T_v)\right] - j(1-a_i)\sin(\omega T_v)} \qquad \text{for } i = u, v, w, p \qquad (53)$$

and

$$\left.\begin{aligned} a_u &= \frac{T_v}{\tau_u} \\ a_v &= \frac{2T_v}{\tau_v} \\ a_w &= \frac{2T_v}{\tau_w} \\ a_p &= \frac{T_v}{\tau_p} \end{aligned}\right\} \qquad (54)$$

$$G_{di}(\omega) = \frac{T_v}{2\pi}\left|H_i\left(e^{j\omega T_v}\right)\right|^2 \qquad \text{for } i,j = q,w \text{ or } r,v$$

$$= \frac{\frac{T_v}{2\pi}k_i^2k_j^2\left[2-2\cos(\omega T_v)\right]}{\left(1+X^2+Y^2\right) - 2X(1+Y)\cos(\omega T_v) + 2Y\cos(2\omega T_v)} \qquad (55)$$

where

$$H_i(z) = \frac{\xi_i(z)}{v_j(z)} \qquad \text{for } i,j = q, w \text{ or } r, v$$

$$= \frac{k_ik_j\left(1-z^{-1}\right)}{1-\left(2-a_i-a_j\right)z^{-1} + (1-a_i)\left(1-a_j\right)z^{-2}} \qquad (56)$$

$$= \frac{k_ik_j\left(1-z^{-1}\right)}{1-Xz^{-1}+Yz^{-2}}$$

and
$$\left.\begin{aligned} a_q &= \frac{T_v}{\tau_q} \\ a_r &= \frac{T_v}{\tau_r} \\ k_q &= \frac{\pi}{4b_w} \\ k_r &= \frac{\pi}{3b_w} \\ k_v &= \sigma_v\sqrt{\frac{4T_v}{\tau_v}} \\ k_w &= \sigma_w\sqrt{\frac{4T_v}{\tau_w}} \\ X &= 2 - a_i - a_j \\ Y &= (1-a_i)(1-a_j) \end{aligned}\right\} \quad \text{for} \quad i,j = q,w \text{ or } r,v \qquad (57)$$

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.

```
% 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(Lwv*bwing)*sigma;
Npts = 200;

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

tauu = Lu/V;
tauwv = Lwv/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));
%
Gmilu = zeros(size(w));
Gmilwv = zeros(size(w));
Gmilp = zeros(size(w));
Gmilq = zeros(size(w));
Gmilr = zeros(size(w));

% * * * * * * * * * * * * * * * * * * * * * *
%  Calculate psd's
% * * * * * * * * * * * * * * * * * * * * * *

for i = 1:Npts,

  Snu(i) = 10.*log10(sin(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));
  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 = 1./tauu;
   cfil = wud/tan(wud*Tnu/2.);
   sdnois = 1.;
   kud = sigma*sqrt(tauu/pi);
   cud = sdnois*sqrt(pi*(wud + cfil));
   ufacl = kud^2*wud^2;
   ud1 = wud^2+cfil^2;
   ud2 = wud^2-cfil^2;
%
%  Mil constants  - u
   au = Tnu/tauu;
   onemau = 1. - au;
   onemausq = onemau^2;
   Tnusq = Tnu^2;
   sigusq=sigma^2;
   milunum= Tnusq/(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;
   wvfac1 = kwvd^2*sigma^2;
   abcsum = 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;

%
% * * * * * * * * * * * * * * * * * * * * * *
%   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);
%
%
%  Mil constants  - p
   ap = Tnu/taup;
   onemap = 1. - ap;
   onemapsq = onemap^2;
   sigpsq = sigp^2;
   milpnum= Tnusq/(pi*taup)*sigpsq;

% * * * * * * * * * * * * * * * * * * * * * *
%   q-component
% * * * * * * * * * * * * * * * * * * * * * *
%
   wqd = 1./tauq;
   cfilq = wqd/tan(Tnu/(2.*tauq));
   maqt = (1. + cfilq*tauq);
   mmqt = (1. - cfilq*tauq);
   kqd = cfilq/V;
%
%
%  Mil constants  - q
   aq = Tnu/tauq;
   onemaq = 1. - aq;
% onemawv defined above
   Xqw = 2. - aq - awv;
   Yqw = onemaq*onemawv;
   milqwk1 = 1. + Xqw^2 + Yqw^2;
   milqwk2 = 2.*Xqw*(1. + Yqw);
   milqwk3 = 2.* Yqw;
   kq = pi/(4.*bwing);
   kw = sigma*sqrt(2.*awv);
   Tnuov2pi = Tnu/(2.*pi);
   kqsq=kq^2;
   kwsq=kw^2;
   milqwnk1= Tnuov2pi*kqsq*kwsq;
%   milqwnk1= 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;
   milrvnk1= Tnuov2pi*krsq*kvsq;
%   milrvnk1= Tnuov2pi^3*krsq*kvsq;
%
% * * * * * * * * * * * * * * * * * * * * * *
%   Calculate psd's
%
%   u-component
% * * * * * * * * * * * * * * * * * * * * * *
%
for i = 1:Npts,

   coswt = cos(w(i)*Tnu);
   cos2wt = cos(2*w(i)*Tnu);
   UN = ufac1*(1. + coswt);
   UD = (ud1 + (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
% * * * * * * * * * * * * * * * * * * * * * *
```

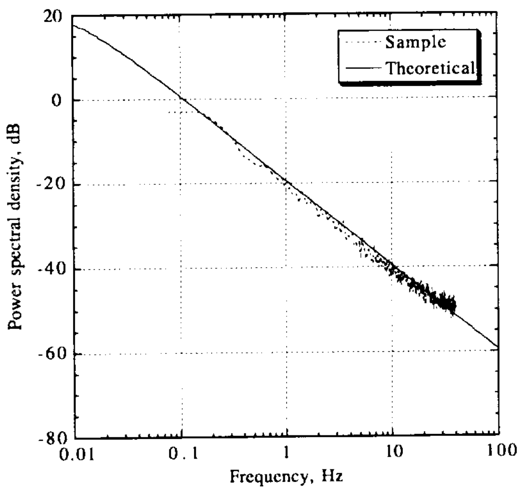
```
%
    wvfac2 = 2.*(a*b + b*c)*coswt;
    wvfac3 = 2.*a*c*cos2wt;
    wvfac4 = 2.*(d*e + e*f)*coswt;
    wvfac5 = 2.*d*f*cos2wt;
%

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

   Gdwv(i) = 10.*log10(WVN/WVD);
%
%
%  Mil specs - w,v
%
   milwvd = 1. + onemawvsq - 2.*onemawv*coswt;
   Gmilwv(i) = 10.*log10(milwvnum/milwvd);
%
% * * * * * * * * * * * * * * * * * * * * * * *
%   p-component
% * * * * * * * * * * * * * * * * * * * * * * *
%
    PN = kpd^2*2.*(1. + coswt);
    PD = mapt^2 + mmpt^2 + 2.*mapt*mmpt*(coswt);

   Gdp(i) = 10.*log10(PN/PD);
%
%
%  Mil specs - p
%
   milpd = 1. + onemapsq - 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
% * * * * * * * * * * * * * * * * * * * * * * *
%
```

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

  Gdr(i) = 10.*log10(RN/RD) + Gdwv(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);
```

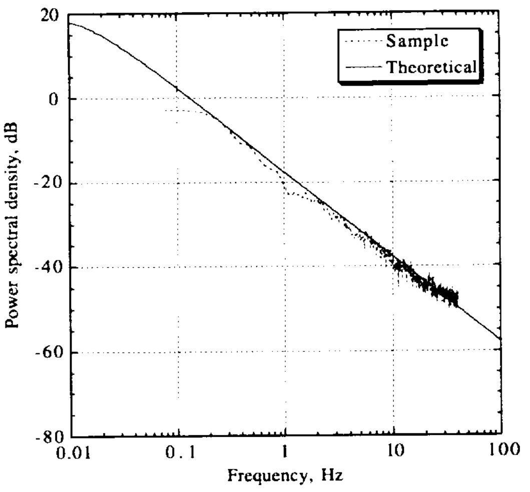
```
Pud = 10.*log10(Pud*SF);
%
[Pvd,fm] = psd(filv,nfft,fs,win,nov);
Pvd = 10.*log10(Pvd*SF);
%
[Pwd,fm] = psd(filw,nfft,fs,win,nov);
Pwd = 10.*log10(Pwd*SF);
%
[Ppd,fm] = psd(filp,nfft,fs,win,nov);
Ppd = 10.*log10(Ppd*SF);
%
[Pqd,fm] = psd(filq,nfft,fs,win,nov);
Pqd = 10.*log10(Pqd*SF);
%
[Prd,fm] = psd(filr,nfft,fs,win,nov);
Prd = 10.*log10(Prd*SF);
%
%
%  Now do discrete mil spec psd's
%
[Pmilud,fm] = psd(miluk,nfft,fs,win,nov);
Pmilud = 10.*log10(Pmilud*SF);
%
[Pmilvd,fm] = psd(milvk,nfft,fs,win,nov);
Pmilvd = 10.*log10(Pmilvd*SF);
%
[Pmilwd,fm] = psd(milwk,nfft,fs,win,nov);
Pmilwd = 10.*log10(Pmilwd*SF);
%
[Pmilpd,fm] = psd(milpk,nfft,fs,win,nov);
Pmilpd = 10.*log10(Pmilpd*SF);
%
[Pmilqd,fm] = psd(milqk,nfft,fs,win,nov);
Pmilqd = 10.*log10(Pmilqd*SF);
%
[Pmilrd,fm] = psd(milrk,nfft,fs,win,nov);
Pmilrd = 10.*log10(Pmilrd*SF);
% * * * * * * * * * * * * * * * * * * * * * * *
%  Write results
% * * * * * * * * * * * * * * * * * * * * * * *
%
outvect = ['ft Snu Su Swv Hqsq Hrsq Sp Sq Sr Gdu Gdwv Gdp Gdq Gdr Gmilu
Gmilwv Gmilp Gmilq Gmilr'];
gdwrite('fig12dattj.asc2 asc2', outvect)
gdwrite('fig12datmj.asc2 asc2','fm Puc,Pvc,Pwc
Ppc,Pqc,Prc,Pud,Pvd,Pwd,Ppd,Pqd,Prd,Pmilud,Pmilvd,
Pmilwd,Pmilpd,Pmilqd,Pmilrd')
```

Plots

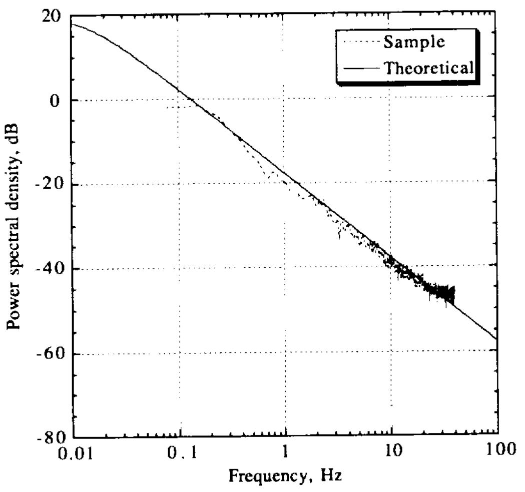
PSD's of sample sequences of each turbulence component were produced using *fig12dat.m* for each of the three models. These data were input into a commercial plotting program, and the resulting plots are shown below in figures 7 through 12. These plots compare the sample PSD and theoretical PSD for each of the turbulence sequences that were shown in figures 1 through 6.



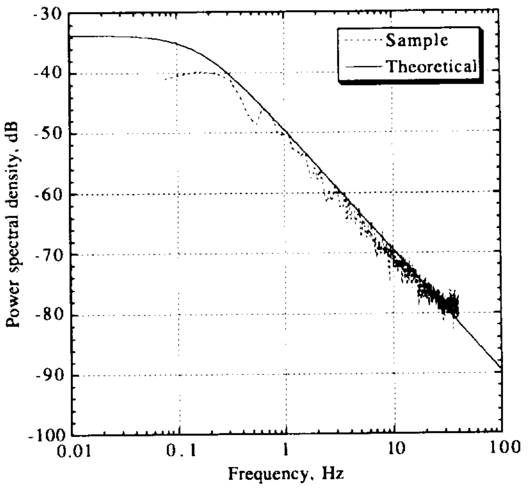
(a).- u-component



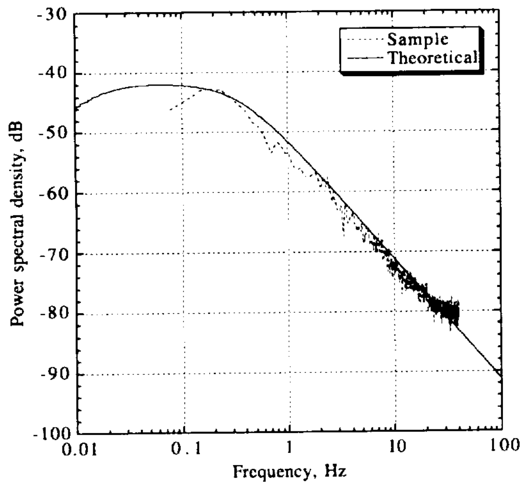
(b).- v-component



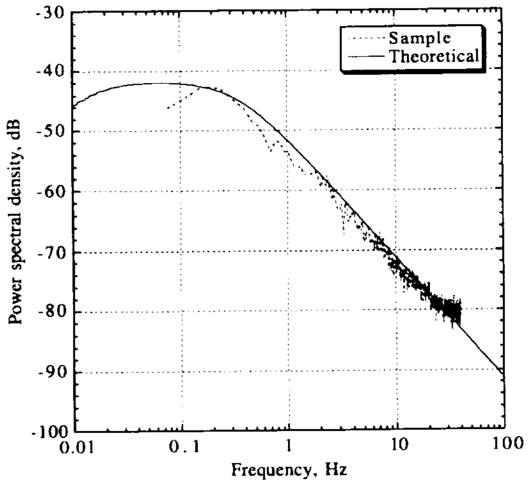
(c).- w-component



(d).- p-component

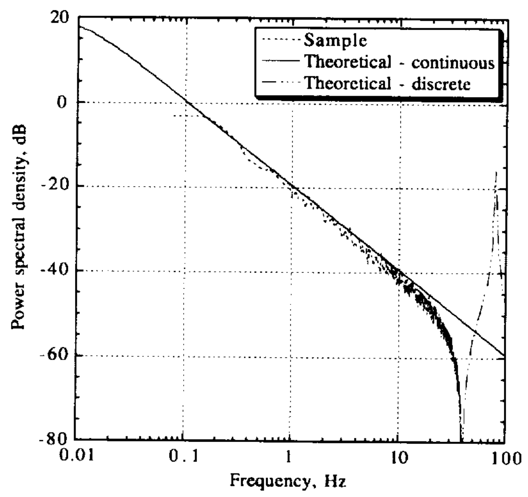


(e).- q-component

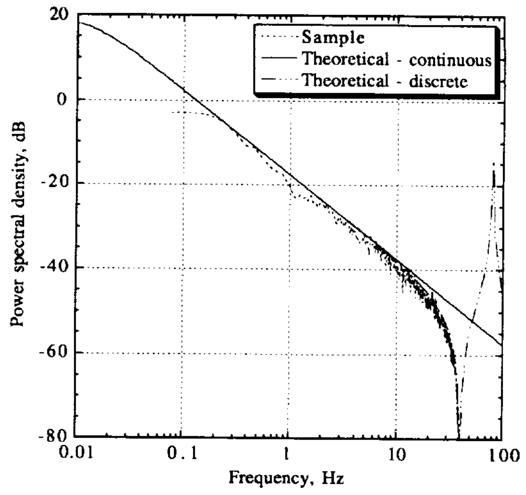


(f).- r-component

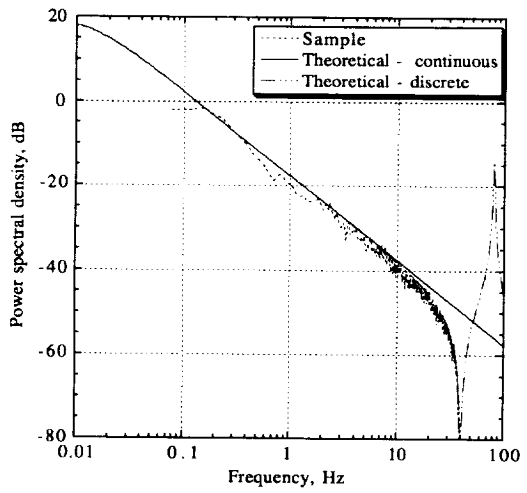
Figure 7. Power spectral densities, V = 100 ft/sec, continuous model.



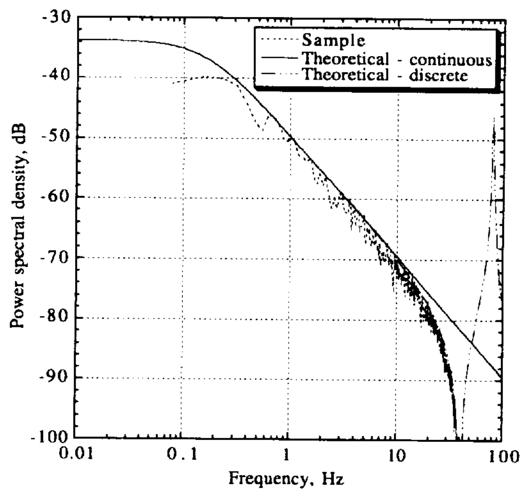
(a).- u-component



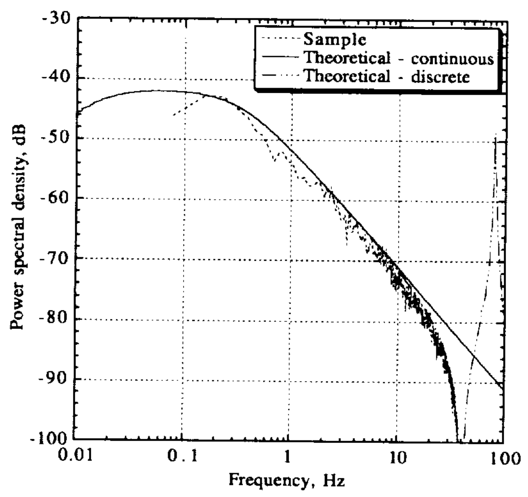
(b).- v-component



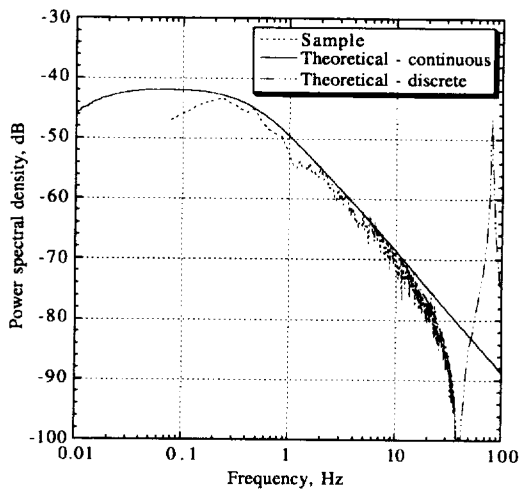
(c).- w-component



(d).- p-component

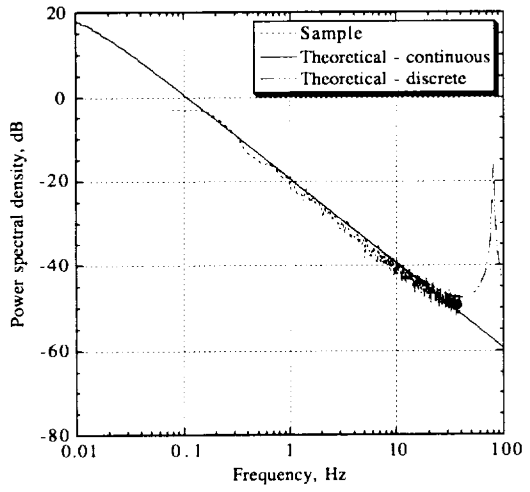


(e).- q-component

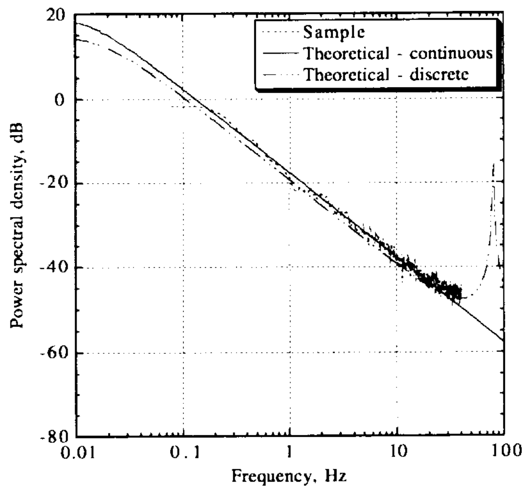


(f).- r-component

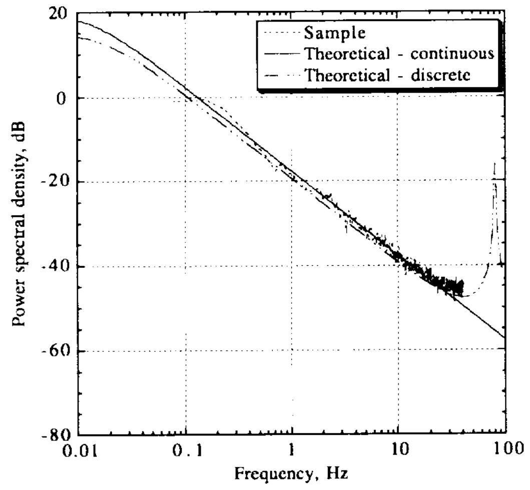
Figure 8. Power spectral densities, V = 100 ft/sec, Tustin model.



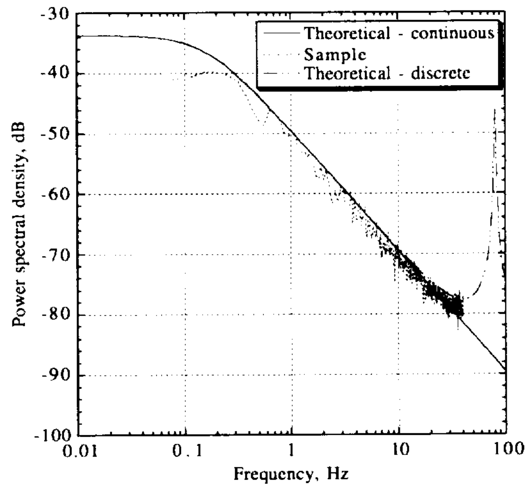
(a).- u-component



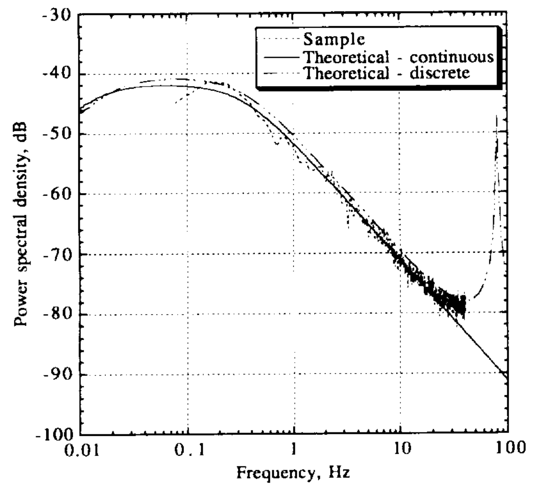
(b).- v-component



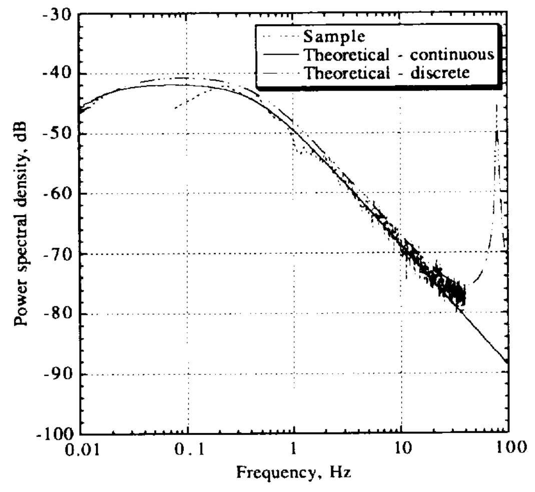
(c).- w-component



(d).- p-component

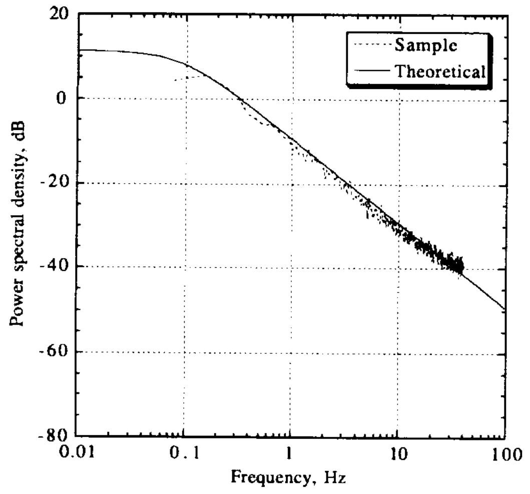


(e).- q-component

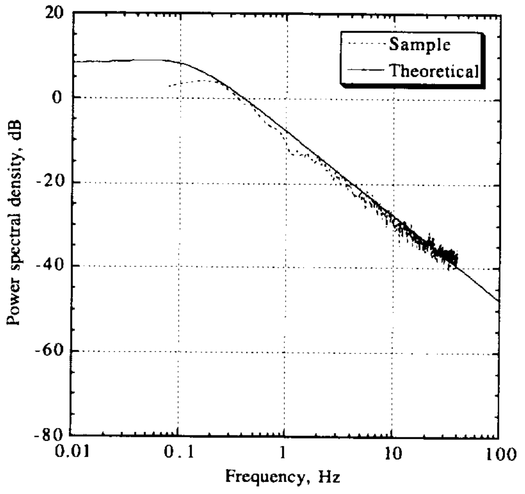


(f).- r-component

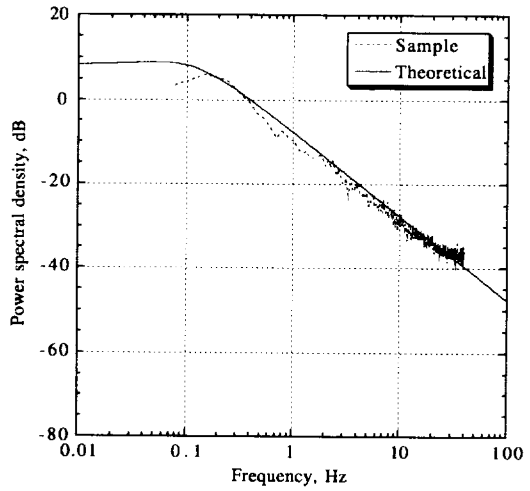
Figure 9. Power spectral densities, V = 100 ft/sec, MIL STD model.



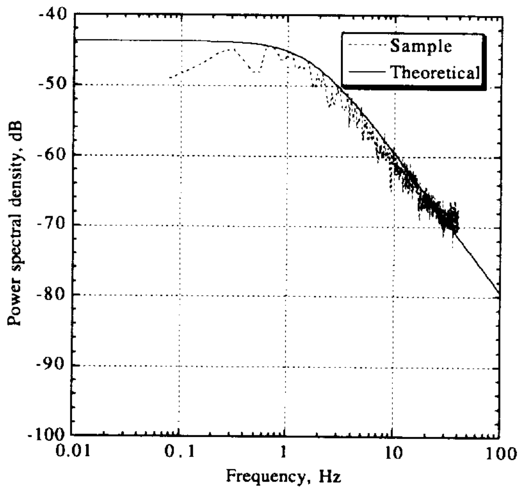
(a).- u-component



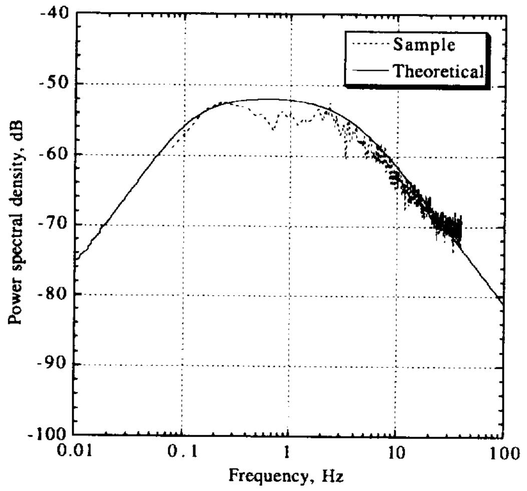
(b).- v-component



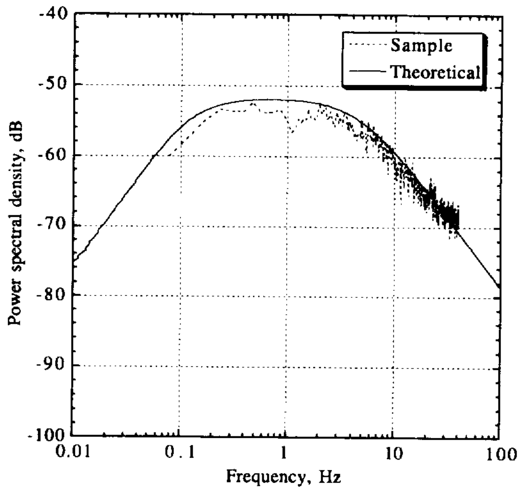
(c).- w-component



(d).- p-component

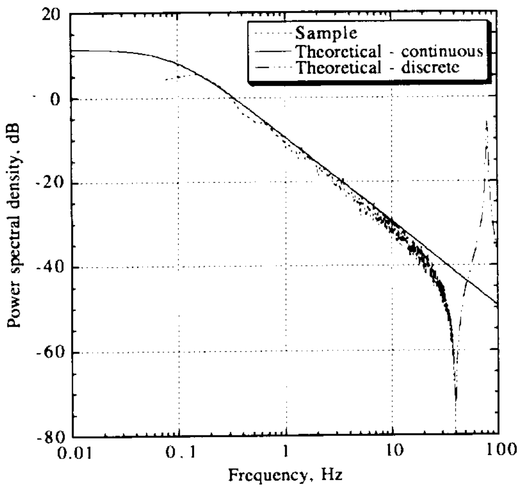


(e).- q-component

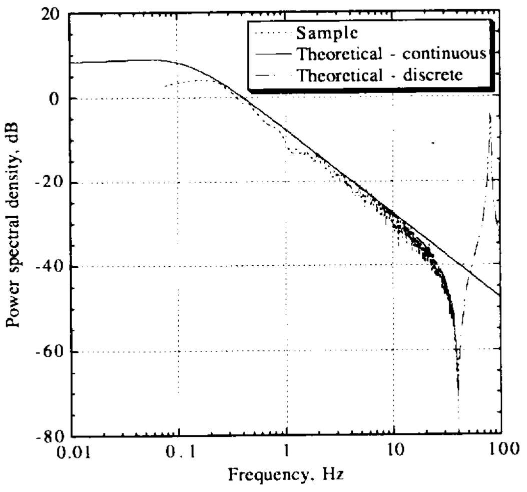


(f).- r-component

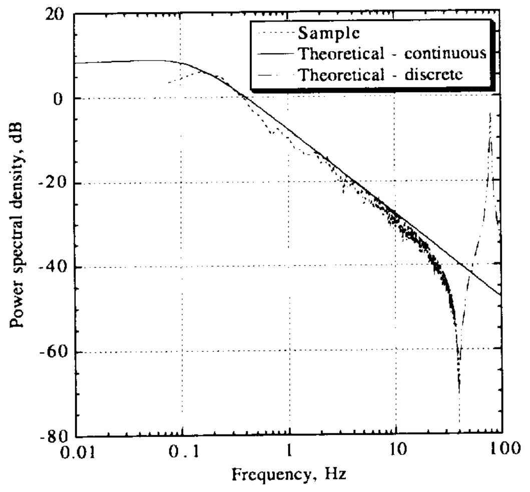
Figure 10. Power spectral densities, V = 1000 ft/sec, continuous model.



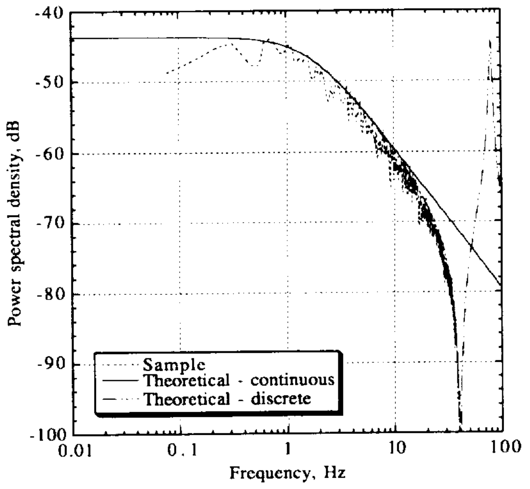
(a).- u-component



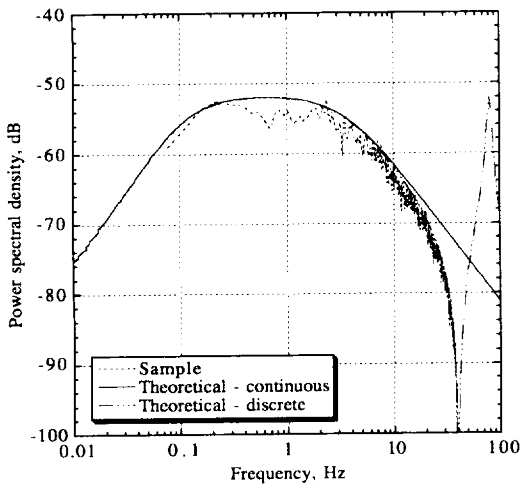
(b).- v-component



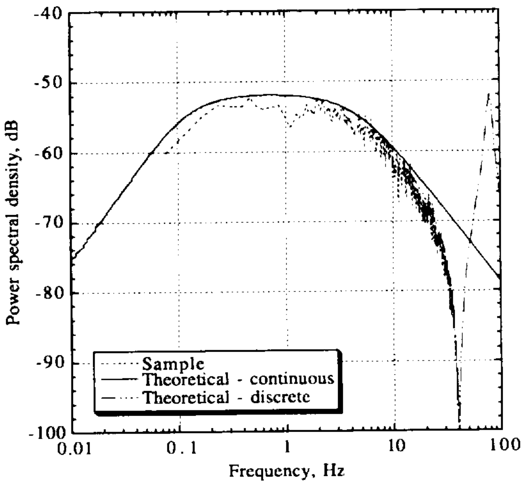
(c).- w-component



(d).- p-component

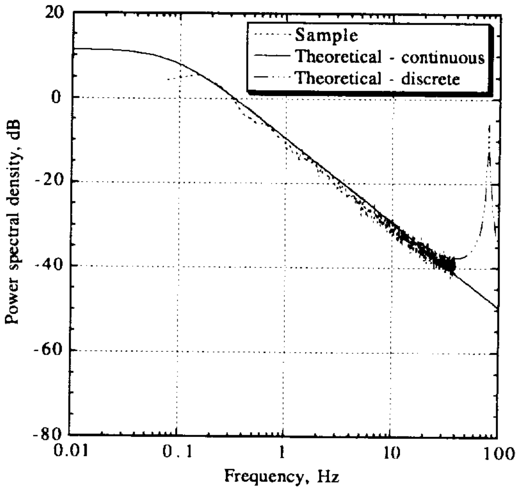


(e).- q-component

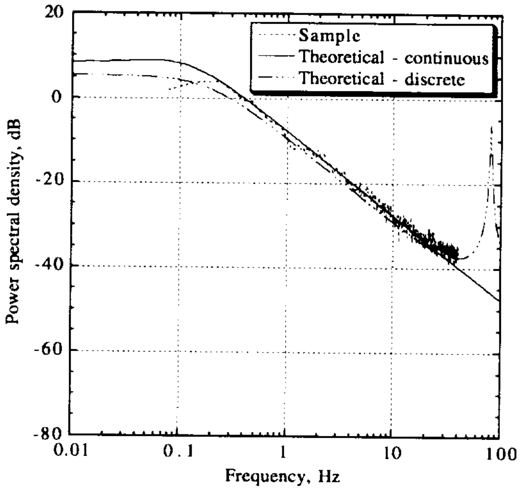


(f).- r-component

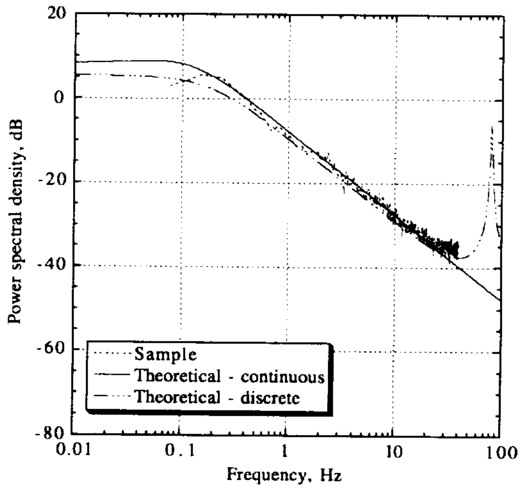
Figure 11. Power spectral densities, V = 1000 ft/sec, Tustin model.



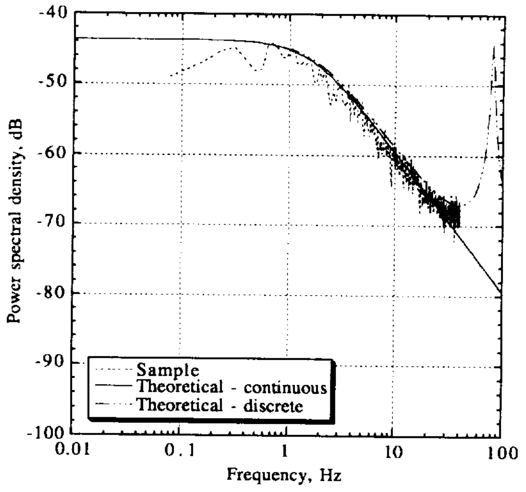
(a).- u-component



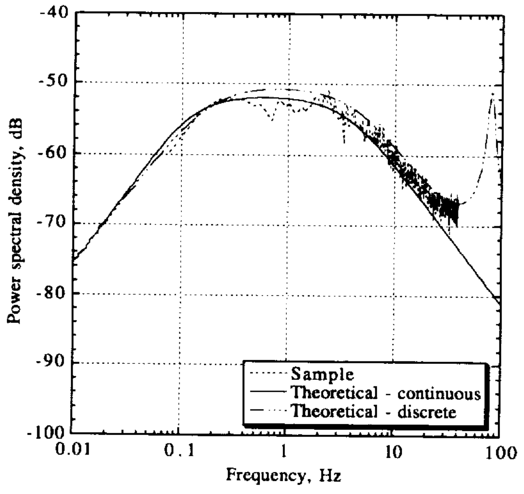
(b).- v-component



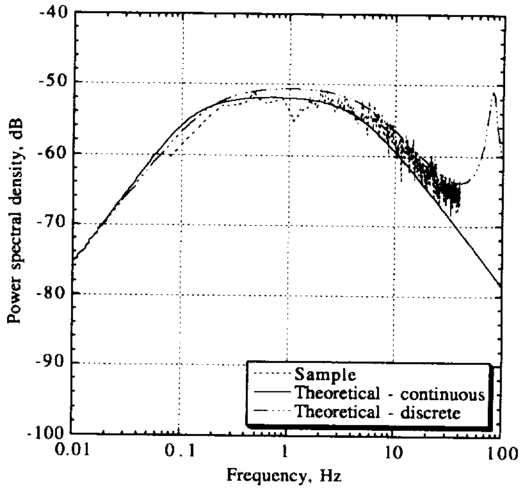
(c).- w-component



(d).- p-component



(e).- q-component



(f).- r-component

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

$$\hat{m}_{\xi_i}^{(j)} = \frac{1}{N}\sum_{k=1}^{N}\xi_i(k); \qquad i = u,v,w,p,q,r; \quad j = 1,2,\ldots,10 \tag{58}$$

$$\hat{\sigma}_{\xi_i}^{(j)} = \sqrt{\frac{1}{(N-1)}\sum_{k=1}^{N}\left\{\xi_i^{(j)}(k) - m_{\xi_i}^{(j)}\right\}^2} \tag{59}$$

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:

$$\hat{M}_g = \frac{1}{10}\sum_{j=1}^{10} g^{(j)} \tag{60}$$

$$\hat{\Sigma}_g = \sqrt{\frac{1}{(10-1)}\sum_{j=1}^{10}\left\{g^{(j)} - m_g\right\}^2} \tag{61}$$

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

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; $V = 100$ ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
34	4.95	5.27	4.90	0.0371	0.0212	0.0242
36	4.80	5.51	5.83	0.0375	0.0214	0.0245
37	5.27	5.39	4.82	0.0373	0.0209	0.0239
38	4.84	4.20	5.16	0.0366	0.0209	0.0236
39	5.27	4.77	4.51	0.0379	0.0214	0.0243
40	5.24	5.32	4.76	0.0394	0.0207	0.0236
41	5.11	4.88	5.00	0.0366	0.0210	0.0245
42	5.46	5.11	4.78	0.0372	0.0209	0.0240
43	5.36	5.20	5.07	0.0368	0.0208	0.0236
44	4.40	5.23	5.24	0.0360	0.0215	0.0243
Mean	5.07	5.09	5.01	0.0372	0.0211	0.0241
Std. Dev.	0.321	0.382	0.358	0.000912	0.000273	0.000369
Run	**Mean**					
	u	v	w	p	q	r
34	0.296	-0.112	-0.463	0.000106	-6.42e-05	6.54e-05
36	-1.23	-0.951	-0.388	-0.000551	3.45e-06	2.08e-05
37	-0.261	-0.740	0.214	-0.00264	-2.90e-05	-0.000109
38	0.259	1.11	-0.418	-0.000202	-5.56e-05	5.06e-05
39	1.77	-0.499	-0.185	-0.00267	8.01e-06	-5.22e-06
40	1.11	0.849	0.606	-6.02e-05	2.72e-05	1.67e-05
41	-0.419	0.229	-0.351	0.000188	-7.82e-05	-3.97e-05
42	-1.09	0.887	-0.284	-0.000544	-4.67e-05	-2.52e-05
43	0.425	-1.26	0.443	-0.000526	1.81e-05	-6.10e-05
44	-0.660	-0.258	-0.923	-0.00361	-4.15e-05	5.71e-05
Mean	0.0204	-0.0743	-0.175	-0.00105	-2.58e-05	-2.96e-06
Std. Dev.	0.950	0.822	0.463	0.00138	3.73e-05	5.63e-05

Table 3. Statistics for Ten 1000-Second Simulation Runs;
Tustin Case; V = 100 ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
34	4.95	5.23	4.89	0.0370	0.0211	0.0240
36	4.80	5.44	5.75	0.0374	0.0213	0.0243
37	5.26	5.35	4.79	0.0371	0.0207	0.0237
38	4.84	4.18	5.11	0.0365	0.0208	0.0233
39	5.26	4.74	4.51	0.0377	0.0212	0.0241
40	5.23	5.27	4.73	0.0393	0.0206	0.0234
41	5.11	4.84	4.98	0.0364	0.0208	0.0243
42	5.45	5.08	4.77	0.0371	0.0208	0.0238
43	5.36	5.15	5.07	0.0367	0.0207	0.0234
44	4.40	5.21	5.22	0.0359	0.0214	0.0241
Mean	5.07	5.05	4.98	0.0372	0.0209	0.0238
Std. Dev.	0.320	0.373	0.341	0.000919	0.000276	0.000372

Run	Mean					
	u	v	w	p	q	r
34	0.295	-0.0971	-0.450	0.000106	-6.22e-05	6.32e-05
36	-1.24	-0.897	-0.380	-0.000551	5.48e-06	1.95e-05
37	-0.258	-0.697	0.205	-0.00264	-3.03e-05	-0.000111
38	0.260	1.06	-0.408	-0.000198	-5.59e-05	4.93e-05
39	1.77	-0.474	-0.176	-0.00267	7.45e-06	-5.30e-06
40	1.11	0.812	0.585	-6.05e-05	2.74e-05	1.45e-05
41	-0.413	0.217	-0.342	0.000191	-7.60e-05	-3.90e-05
42	-1.08	0.854	-0.275	-0.000548	-4.70e-05	-2.78e-05
43	0.422	-1.18	0.418	-0.000532	1.82e-05	-6.23e-05
44	-0.661	-0.239	-0.877	-0.00362	-4.21e-05	5.65e-05
Mean	0.0207	-0.0646	-0.170	-0.00105	-2.55e-05	-4.22e-06
Std. Dev.	0.948	0.781	0.444	0.00138	3.70e-05	5.62e-05

Table 4. Statistics for Ten 1000-Second Simulation Runs;
MIL STD Case; V = 100 ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
34	4.95	5.29	4.90	0.0372	0.0245	0.0280
36	4.80	5.47	5.76	0.0376	0.0247	0.0284
37	5.27	5.33	4.85	0.0374	0.0241	0.0277
38	4.84	4.22	5.11	0.0368	0.0241	0.0273
39	5.26	4.80	4.59	0.0380	0.0246	0.0282
40	5.24	5.28	4.81	0.0395	0.0239	0.0274
41	5.11	4.91	4.93	0.0367	0.0242	0.0284
42	5.46	5.09	4.80	0.0373	0.0241	0.0278
43	5.36	5.11	5.03	0.0369	0.0240	0.0274
44	4.40	5.18	5.23	0.0361	0.0248	0.0282
Mean	5.07	5.07	5.00	0.0373	0.0243	0.0279
Std. Dev.	0.320	0.357	0.320	0.000914	0.000312	0.000421
Run	Mean					
	u	v	w	p	q	r
34	0.295	-0.121	-0.451	0.000106	-6.13e-05	5.09e-05
36	-1.24	-0.956	-0.368	-0.000551	5.42e-06	3.08e-06
37	-0.258	-0.752	0.203	-0.00264	-2.34e-05	-0.000121
38	0.260	1.10	-0.423	-0.000197	-5.72e-05	4.72e-05
39	1.77	-0.491	-0.191	-0.00267	9.76e-06	-1.80e-05
40	1.11	0.835	0.617	-6.16e-05	1.64e-05	1.69e-05
41	-0.413	0.232	-0.341	0.000191	-6.41e-05	-5.00e-05
42	-1.08	0.866	-0.295	-0.000548	-4.24e-05	-2.16e-05
43	0.422	-1.27	0.443	-0.000532	2.34e-05	-6.00e-05
44	-0.661	-0.262	-0.930	-0.00362	-2.94e-05	6.86e-05
Mean	0.0206	-0.0819	-0.174	-0.00105	-2.23e-05	-8.37e-06
Std. Dev.	0.948	0.821	0.464	0.00138	3.39e-05	5.83e-05

Table 5. Statistics for Ten 1000-Second Simulation Runs;
Continuous Case; V = 1000 ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
35	5.07	5.03	5.07	0.0373	0.0207	0.0238
45	5.17	5.20	5.22	0.0372	0.0207	0.0238
46	5.16	4.94	4.98	0.0373	0.0206	0.0237
47	5.10	4.79	5.05	0.0370	0.0207	0.0237
48	5.28	5.07	5.16	0.0373	0.0208	0.0240
49	5.31	4.91	4.92	0.0376	0.0208	0.0237
50	5.26	5.03	5.03	0.0370	0.0206	0.0240
51	5.22	4.96	5.03	0.0372	0.0207	0.0239
52	4.92	4.85	5.01	0.0370	0.0206	0.0238
53	4.91	5.10	5.20	0.0368	0.0207	0.0238
Mean	5.14	4.99	5.07	0.0372	0.0207	0.0238
Std. Dev.	0.141	0.122	0.0973	0.000228	8.38e-05	0.000115

Run	Mean					
	u	v	w	p	q	r
35	0.0393	-0.0217	-0.159	3.68e-05	1.63e-08	-4.04e-06
45	-0.417	-0.302	-0.115	-0.000174	-3.16e-06	-2.90e-07
46	-0.0929	-0.266	0.0601	-0.000824	-8.60e-06	-7.35e-06
47	0.0400	0.362	-0.147	-5.01e-05	-9.49e-07	-1.41e-06
48	0.551	-0.157	-0.0590	-0.000864	1.58e-06	-9.09e-06
49	0.391	0.271	0.199	-4.39e-05	-6.69e-06	-6.65e-06
50	-0.125	0.0640	-0.128	5.42e-05	4.89e-06	-9.71e-06
51	-0.367	0.270	-0.104	-0.000167	-1.82e-07	-3.87e-06
52	0.143	-0.416	0.143	-0.000171	8.47e-06	-6.60e-06
53	-0.210	-0.0693	-0.303	-0.00115	2.91e-06	8.59e-06
Mean	-0.00478	-0.0266	-0.0612	-0.000335	-1.72e-07	-4.04e-06
Std. Dev.	0.308	0.266	0.152	0.000437	5.12e-06	5.40e-06

Table 6. Statistics for Ten 1000-Second Simulation Runs;
Tustin Case; V = 1000 ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
35	5.06	5.02	5.06	0.0363	0.0195	0.0221
45	5.16	5.18	5.20	0.0362	0.0195	0.0221
46	5.15	4.93	4.97	0.0362	0.0194	0.0220
47	5.08	4.77	5.03	0.0359	0.0195	0.0219
48	5.26	5.05	5.15	0.0363	0.0196	0.0223
49	5.30	4.90	4.91	0.0366	0.0196	0.0220
50	5.25	5.02	5.02	0.0360	0.0194	0.0222
51	5.21	4.95	5.02	0.0361	0.0195	0.0221
52	4.91	4.84	5.00	0.0359	0.0194	0.0220
53	4.90	5.08	5.19	0.0358	0.0195	0.0220
Mean	5.13	4.97	5.05	0.0361	0.0195	0.0221
Std. Dev.	0.141	0.122	0.0977	0.000235	8.50e-05	0.000120
Run	Mean					
	u	v	w	p	q	r
35	0.0389	-0.0219	-0.160	3.69e-05	-5.64e-08	-4.11e-06
45	-0.418	-0.303	-0.115	-0.000174	-3.61e-06	-5.62e-08
46	-0.0919	-0.266	0.0605	-0.000824	-8.72e-06	-7.21e-06
47	0.0404	0.362	-0.149	-4.84e-05	-7.64e-07	-1.44e-06
48	0.551	-0.157	-0.0594	-0.000865	1.49e-06	-9.28e-06
49	0.390	0.271	0.201	-4.46e-05	-6.83e-06	-6.68e-06
50	-0.123	0.0647	-0.127	5.50e-05	4.66e-06	-9.81e-06
51	-0.365	0.272	-0.106	-0.000168	-1.10e-07	-3.78e-06
52	0.142	-0.416	0.143	-0.000173	8.33e-06	-6.40e-06
53	-0.211	-0.0689	-0.304	-0.00115	2.69e-06	8.24e-06
Mean	-0.00473	-0.0262	-0.0615	-0.000335	-2.93e-07	-4.05e-06
Std. Dev.	0.308	0.267	0.153	0.000437	5.12e-06	5.34e-06

Table 7. Statistics for Ten 1000-Second Simulation Runs;
MIL STD Case; V = 1000 ft/sec

Run	Standard deviation					
	u	v	w	p	q	r
35	5.07	5.06	5.10	0.0386	0.0258	0.0308
45	5.17	5.19	5.23	0.0385	0.0258	0.0308
46	5.17	4.97	5.01	0.0385	0.0257	0.0307
47	5.10	4.81	5.05	0.0383	0.0258	0.0307
48	5.28	5.06	5.15	0.0386	0.0260	0.0310
49	5.31	4.93	4.95	0.0389	0.0260	0.0307
50	5.27	5.06	5.05	0.0383	0.0258	0.0310
51	5.23	4.98	5.03	0.0385	0.0258	0.0309
52	4.92	4.88	5.03	0.0383	0.0257	0.0308
53	4.92	5.10	5.21	0.0381	0.0259	0.0308
Mean	5.15	5.00	5.08	0.0385	0.0258	0.0308
Std. Dev.	0.141	0.112	0.0906	0.000227	0.000102	0.000132
Run	Mean					
	u	v	w	p	q	r
35	0.0387	-0.0229	-0.159	3.71e-05	-6.71e-07	-3.69e-06
45	-0.418	-0.301	-0.115	-0.000175	-4.36e-06	5.25e-07
46	-0.0919	-0.264	0.0600	-0.000823	-9.53e-06	-4.69e-06
47	0.0403	0.360	-0.148	-4.77e-05	-2.11e-07	-1.92e-06
48	0.551	-0.157	-0.0590	-0.000865	6.75e-07	-9.44e-06
49	0.390	0.270	0.201	-4.53e-05	-7.01e-06	-8.05e-06
50	-0.123	0.0644	-0.126	5.54e-05	4.44e-06	-9.36e-06
51	-0.365	0.270	-0.104	-0.000167	-3.17e-07	-3.44e-06
52	0.142	-0.414	0.144	-0.000173	8.52e-06	-6.00e-06
53	-0.211	-0.0691	-0.303	-0.00115	2.65e-06	7.13e-06
Mean	-0.00476	-0.0263	-0.0609	-0.000335	-5.82e-07	-3.89e-06
Std. Dev.	0.308	0.266	0.153	0.000437	5.33e-06	5.04e-06

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.

$$\Sigma(\hat{m}_i) \approx \sigma_i \sqrt{2\left(\frac{\tau_i}{T} - \frac{\tau_i^2}{T^2}\right)} \qquad i = u, p \tag{62}$$

$$\Sigma(\hat{m}_i) \approx \sigma_i \sqrt{\frac{\tau_i}{T}} \qquad i = v, w \tag{63}$$

$$\Sigma(\hat{m}_i) \approx \frac{\sigma_j}{V\tau_i T}\left\{\begin{matrix} T\tau_j\left(R_{j-} + R_{j+}\right) + T\tau_i\left(R_{i-} + R_{i+}\right) + T\tau_j^2\left(R_{j+\tau} - R_{j-\tau}\right) \\ -\tau_j^2\left(R_{j+} + R_{j-}\right) - \tau_i^2\left(R_{i+} + R_{i-}\right) + 2\tau_j^3\left(R_{j-\tau} - R_{j+\tau}\right) \end{matrix}\right\}^{\frac{1}{2}} \qquad i,j = q,w \text{ or } r,v \tag{64}$$

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

$$\Sigma(\hat{\sigma}_i) \approx \sigma_i \left\{\left(\frac{\tau_i}{2T} - \frac{9\tau_i^2}{4T^2} + \frac{4\tau_i^3}{T^3} - \frac{2\tau_i^4}{T^4}\right)\right\}^{\frac{1}{2}} \qquad i = u, p \tag{65}$$

$$\Sigma(\hat{\sigma}_i) \approx \frac{\sigma_i}{2}\left\{\left(\frac{13\tau_i}{4T} - \frac{83\tau_i^2}{8T^2}\right)\right\}^{\frac{1}{2}} \qquad i = v, w \tag{66}$$

$$\Sigma(\hat{\sigma}_i) \approx \frac{\sigma_j^2}{\sigma_i V^2 \tau_i^2 T\sqrt{2}}\left\{S_{00} - 2\begin{bmatrix} T\tau_j\left(R_{j-} + R_{j+}\right) + T\tau_i\left(R_{i-} + R_{i+}\right) \\ +T\tau_j^2\left(R_{j+\tau} - R_{j-\tau}\right) - \tau_j^2\left(R_{j+} + R_{j-}\right) \\ -\tau_i^2\left(R_{i+} + R_{i-}\right) + 2\tau_j^3\left(R_{j-\tau} - R_{j+\tau}\right) \end{bmatrix}^2\right\}^{\frac{1}{2}} \tag{67}$$

where

$$\begin{aligned} S_{00} = {} & \frac{T\tau_j}{2}\left(R_{j-}^2 + R_{j+}^2\right) + \frac{T\tau_i}{2}\left(R_{i-}^2 + R_{i+}^2\right) + \frac{2T}{\left(1/\tau_i + 1/\tau_j\right)}\left(R_{j-}R_{i-} + R_{j+}R_{i+}\right) \\ & + \frac{\tau_j^2}{4}\left(-R_{j-}^2 - 2R_{j-}R_{j-\tau}T - R_{j+}^2 + 2R_{j+}R_{j+\tau}T\right) - \frac{\tau_i^2}{4}\left(R_{i-}^2 + R_{i+}^2\right) \\ & + \frac{2R_{i+}\left(R_{j+\tau}T - R_{j+}\right) - 2R_{i-}\left(R_{j-\tau}T + R_{j-}\right)}{\left(1/\tau_i + 1/\tau_j\right)^2} + \frac{4\left(R_{j+\tau}R_{i+} + R_{j-\tau}R_{i-}\right)}{\left(1/\tau_i + 1/\tau_j\right)^3} \\ & + \frac{\tau_j^3}{4}\left[R_{j+\tau}\left(R_{j+\tau}T - 2R_{j+}\right) + R_{j-\tau}\left(R_{j-\tau}T + 2R_{j-}\right)\right] - \frac{3\tau_j^4}{8}\left(R_{j-\tau}^2 + R_{j+\tau}^2\right) \end{aligned} \quad (68)$$

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 σ, τ, 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 inplemented in the MATLAB m-file *newA41.m*. Comments were added to the file to identify equation numbers listed in the previous section. The variables *Smhatuu, Smhatpp, Smhatqq, Smhatrr,* and *Smhatwv* defined in file *newA41.m* are the standard deviations of the sample means. The variables *Ssighatuu, Ssighatpp, Ssighatqq, Ssighatrr,* and *Ssighatwv* defined in file *newA41.m* are the standard deviations of sample standard deviations.

```
% * * * * * * * * * * * * * * * * * * * * * * * * *          file = newA41.m
% 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 = sigv = 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;
%
```

```
% 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;
tauwv = 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 selqr, etc for equations
% * * * * * * * * * * * * * * * * * * * * * *
%
% * * * * * * * * * * * * * * * * * * * * * *
%  u, p Logic selection
% * * * * * * * * * * * * * * * * * * * * * *
%
% * * * * * * * * * * * * * * * * * * * * * *
%  Selected sigma and tau
% * * * * * * * * * * * * * * * * * * * * * *
%
  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    %  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);
      Smhatuu = Smhat46;
      Ssighatuu = Ssighat53;
      Smhatpp = 'undef';
      Ssighatpp = 'undef';
  elseif selup == 'p'
%      Rpp(n) = Rii(n);
      Smhatpp = Smhat46;
      Ssighatpp = Ssighat53;
      Smhatuu = 'undef';
      Ssighatuu = '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;
     taui = tauq;
     tauj = tauwv;
  elseif selqr == 'r'
     sigi = sigr;
     sigj = sigma;
     taui = taur;
     tauj = tauwv;
  end

%
% * * * * * * * * * * * * * * * * * * * * * * *
%
%  eq. 71 - 77 constants for Rii autocorrelation function
%                    for q,r sigma(mean-hat)
%
% * * * * * * * * * * * * * * * * * * * * * * *
%  for eq. 64 also- use tisq, tjsq
tisq = taui^2;
```

```
tjsq = tauj^2;
%
tauij = tauj*taui;
%
oneovtaui = 1./taui;
oneovtauj = 1./tauj;
RAk3 = oneovtaui+oneovtauj;
RAk12 = oneovtaui/RAk3;

RAk4 = 1./(2.*tauij);
RAk3sq =RAk3^2;
RA4ov3sq = RAk4/RAk3sq;

RAk5 = oneovtaui-oneovtauj;
RAk13 = oneovtaui/RAk5;
RAk5sq =RAk5^2;
RA4ov5sq = RAk4/RAk5sq;

oneovtauisq = 1./tisq;
oneovtaujsq = 1./tjsq;
RAk6 = oneovtauisq-oneovtaujsq;
RAk11 = oneovtauisq/RAk6;

RAk7 = RAk4*oneovtaui;
RAk5by3sq = RAk5*RAk3sq;
RA5sqby3 = RAk5sq*RAk3;

RAk8 = 1./(2.*tauj);
RAk4ov3 = RAk4/RAk3;
RAk4ov5 = RAk4/RAk5;
RAk7ov6 = RAk7/RAk6;

RAk9 = 1./(2.*taui);
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/RA5sqby3 ;

% * * * * * * * * * *
% Eq 73
% * * * * * * * * * *
Rjnegt = RAk8 - RAk4ov3 - RAk4ov5 + RAk7ov6;

% * * * * * * * * * *
% Eq 74
% * * * * * * * * * *
Rineg  = RAk9ov5 -RAk9ov3 + RAk10/RAk5sq + RAk10/RAk3sq;

% * * * * * * * * * *
% Eq 75
% * * * * * * * * * *
Rjpos  = 1. - RAk13 - RA4ov5sq - RAk12 + RA4ov3sq  ...
           + RAk11 + RAk7/RA5sqby3 - RAk7/RAk5by3sq ;

% * * * * * * * * * *
% Eq 76
% * * * * * * * * * *
Rjpost = - RAk8 + RAk4ov5 + RAk4ov3 - RAk7ov6;

% * * * * * * * * * *
% Eq 77
% * * * * * * * * * *
Ripos  = - RAk9ov3 + RAk9ov5 + RAk10/RAk3sq + RAk10/RAk5sq;
%
Ripsq  = Ripos^2;
Rinsq  = Rineg^2;
Rjpsq  = Rjpos^2;
Rjnsq  = Rjneg^2;
Rjptsq = Rjpost^2;
Rjntsq = Rjnegt^2;
%
% * * * * * * * * * * * * * * * * * * * * * * *
% eq. 71 eq. Rii  autocorrelation function for q,r
% * * * * * * * * * * * * * * * * * * * * * * *
%
% * * * * * * * * * * * * * * * * * * * * * * *
%  start tau  - 'for loop'  for 71
% * * * * * * * * * * * * * * * * * * * * * * *
%
%  constant for 71
   Riik1 = sigj^2/(V^2*tisq);
%
%
    for n = 1:Ntau

       A37pwr1 = tau(n)/tauj;
       A37pwr2 = tau(n)/taui;
%
%  tau < 0
       if tau(n) < 0
          Rii(n) = Riik1*(Rjneg*exp(A37pwr1) ...
                 + Rjnegt*tau(n)*exp(A37pwr1)+ Rineg*exp(A37pwr2));
       else
%  tau(n) >= 0
          Rii(n) = Riik1*(Rjpos*exp(-A37pwr1) ...
                 + Rjpost*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
tjfour = 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*(Rjpost - Rjnegt);
QRk5 = tjsq*(Rjpos + Rjneg);
QRk6 = tisq*(Ripos + Rineg);
QRk7 = 2.*tjcube*(Rjnegt - Rjpost);
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*Rineg + Rjpos*Ripos);
Sk4 = tjsq/4.*(-Rjnsq - 2.*Rjneg*Rjnegt*T - Rjpsq + 2.*Rjpos*Rjpost*T);
Sk5 = tisq/4.*(Rinsq + Ripsq);
```

```
 Sk6num = 2.*Ripos*(Rjpost*T - Rjpos) - 2.*Rineg*(Rjnegt*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;
       Smhatrr = 'undef';
       Ssighatrr = 'undef';
  elseif selqr == 'r'
 %      Rrr(n) = Rii(n);
       Smhatrr = Smhat50;
       Ssighatrr = Ssighat62;
       Smhatqq = 'undef';
       Ssighatqq = 'undef';
  end
 %
 % * * * * * * * * * * * * * * * * * * * * * *
 %  w or v  calculation
 % * * * * * * * * * * * * * * * * * * * * * *
 %
 % * * * * * * * * * * * * * * * * * * * * * *
% eq.70 Rwv  autocorrelation function for v,w
% * * * * * * * * * * * * * * * * * * * * * * *
%
% * * * * * * * * * * * * * * * * * * * * * * *
%  tau  - 'for loop' calculations
% * * * * * * * * * * * * * * * * * * * * * * *
%
    for n = 1: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 - WVk5);
%
   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:

$$\begin{aligned} R_{ii}(\tau) &= F^{-1}\{S_{ii}(\omega)\} \\ &= F^{-1}\left\{\frac{\sigma_i^2\tau_i}{\pi}\frac{1}{1+(\tau_i\omega)^2}\right\} \qquad i = u, p \\ &= \sigma_i^2 e^{-|\tau|/\tau_i} \end{aligned} \tag{69}$$

The autocorrelation function for the w-component is

$$R_{ww}(\tau) = \frac{3\sigma_w^2}{\tau_w}\left[\frac{\tau_w}{3}e^{-|\tau|/\tau_w} - \frac{1}{6}|\tau|e^{-|\tau|/\tau_w}\right] \tag{70}$$

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

$$R_{ii}(\tau) = \begin{cases} \dfrac{\sigma_j^2}{V^2\tau_i^2}\left[R_{j-}e^{\tau/\tau_j} + R_{j-\tau}\tau e^{\tau/\tau_j} + R_{i-}e^{\tau/\tau_i}\right] & \text{for } \tau < 0 \\ \dfrac{\sigma_j^2}{V^2\tau_i^2}\left[R_{j+}e^{-\tau/\tau_j} + R_{j+\tau}\tau e^{-\tau/\tau_j} + R_{i+}e^{-\tau/\tau_i}\right] & \text{for } \tau \geq 0 \end{cases} \tag{71}$$

$$\text{where } R_{j-} = \begin{cases} 1 - \dfrac{1/\tau_i}{(1/\tau_i + 1/\tau_j)} + \dfrac{1/(2\tau_j\tau_i)}{(1/\tau_i + 1/\tau_j)^2} - \dfrac{1/\tau_i}{(1/\tau_i - 1/\tau_j)} - \dfrac{1/(2\tau_j\tau_i)}{(1/\tau_i - 1/\tau_j)^2} \\ + \dfrac{1/\tau_i^2}{(1/\tau_i^2 - 1/\tau_j^2)} - \dfrac{1/(2\tau_j\tau_i^2)}{(1/\tau_i + 1/\tau_j)^2(1/\tau_i - 1/\tau_j)} + \dfrac{1/(2\tau_j\tau_i^2)}{(1/\tau_i - 1/\tau_j)^2(1/\tau_i + 1/\tau_j)} \end{cases} \tag{72}$$

$$R_{j-\tau} = \frac{1}{2\tau_j} - \frac{1/(2\tau_j\tau_i)}{(1/\tau_i + 1/\tau_j)} - \frac{1/(2\tau_j\tau_i)}{(1/\tau_i - 1/\tau_j)} + \frac{1/2\tau_j\tau_i^2}{(1/\tau_i^2 - 1/\tau_j^2)} \tag{73}$$

$$R_{i-} = \frac{1/(2\tau_i)}{(1/\tau_i - 1/\tau_j)} - \frac{1/(2\tau_i)}{(1/\tau_i + 1/\tau_j)} + \frac{1/(4\tau_j\tau_i)}{(1/\tau_i - 1/\tau_j)^2} + \frac{1/(4\tau_j\tau_i)}{(1/\tau_i + 1/\tau_j)^2} \tag{74}$$

$$R_{j+} = \begin{cases} 1 - \dfrac{1/\tau_i}{\left(1/\tau_i - 1/\tau_j\right)} - \dfrac{1/\left(2\tau_j\tau_i\right)}{\left(1/\tau_i - 1/\tau_j\right)^2} - \dfrac{1/\tau_i}{\left(1/\tau_i + 1/\tau_j\right)} + \dfrac{1/\left(2\tau_j\tau_i\right)}{\left(1/\tau_i + 1/\tau_j\right)^2} \\ + \dfrac{1/\tau_i^2}{\left(1/\tau_i^2 - 1/\tau_j^2\right)} + \dfrac{1/\left(2\tau_j\tau_i^2\right)}{\left(1/\tau_i - 1/\tau_j\right)^2\left(1/\tau_i + 1/\tau_j\right)} - \dfrac{1/\left(2\tau_j\tau_i^2\right)}{\left(1/\tau_i + 1/\tau_j\right)^2\left(1/\tau_i - 1/\tau_j\right)} \end{cases} \tag{75}$$

$$R_{j+\tau} = -\frac{1}{2\tau_j} + \frac{1/\left(2\tau_j\tau_i\right)}{\left(1/\tau_i - 1/\tau_j\right)} + \frac{1/\left(2\tau_j\tau_i\right)}{\left(1/\tau_i + 1/\tau_j\right)} - \frac{1/2\tau_j\tau_i^2}{\left(1/\tau_i^2 - 1/\tau_j^2\right)} \tag{76}$$

$$R_{i+} = -\frac{1/(2\tau_i)}{\left(1/\tau_i + 1/\tau_j\right)} + \frac{1/(2\tau_i)}{\left(1/\tau_i - 1/\tau_j\right)} + \frac{1/\left(4\tau_j\tau_i\right)}{\left(1/\tau_i + 1/\tau_j\right)^2} + \frac{1/\left(4\tau_j\tau_i\right)}{\left(1/\tau_i - 1/\tau_j\right)^2} \tag{77}$$

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 ***Ruu***, ***Rpp***, ***Rqq***, ***Rrr***, and ***Rwv*** are the autocorrelation functions from equations (69) through (71) for the turbulence components. The variables ***Rjneg***, ***Rjnegt***, ***Rineg***, ***Rjpos***, ***Rjpost***, and ***Ripos*** 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.

```
%   * * * * * * * * * * * * *  * * * * * *  * * * * *        file A41plts.m
%
%                    autocorrelation plots
%
whitebg
%
%             Theorectical 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 V100')
 xlabel('tau, sec')
 ylabel('Autocorrelation')
 %
 %pause
 %print Rpp
 %
     end
 %
 %    Select q or r component
 %
     if selqr == 'q'
 %
 plot(tau,Rqq)
 title('Theoretical autocorrelation Rqq V100')
 xlabel('tau, sec')
 ylabel('Autocorrelation')
 %
 %pause
 %print Rqq
 %
     elseif selqr == 'r'
 %
 plot(tau,Rrr)
 title('Theoretical autocorrelation Rrr V100')
 xlabel('tau, sec')
 ylabel('Autocorrelation')
 %
 %pause
 %print Rrr
 %
     end
 %
 %    Select w or v component
 %
 plot(tau,Rwv)
 title('Theoretical autocorrelation Rwv V100')
 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 selvw == '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')
```

```
        axis ([-2. 2. -10. 25.])
     end
   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');
```

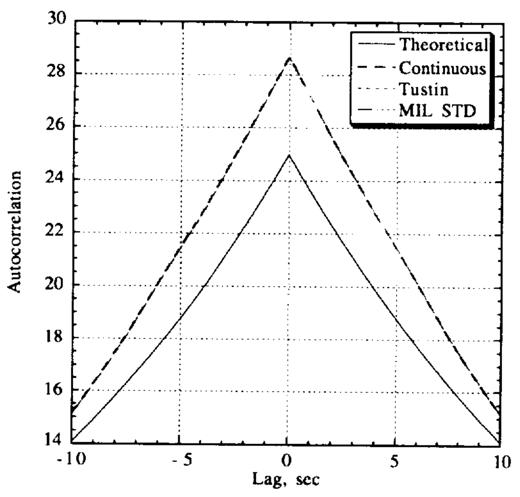
```
Ruucplt = Ruuc(nx-800:nx+800);
RuuTplt = RuuT(nx-800:nx+800);
RuuMplt = RuuM(nx-800:nx+800);

%
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.
```

```
%
        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');
```

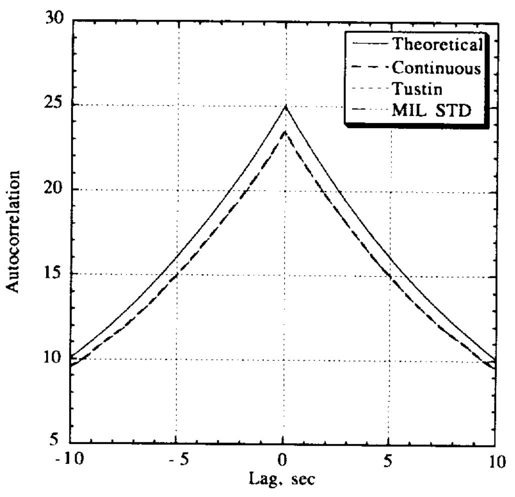
```
Rrrcplt = Rrrc(nx-800:nx+800);
RrrTplt = RrrT(nx-800:nx+800);
RrrMplt = RrrM(nx-800:nx+800);

%
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
%
    end

    if selup == 'u' & selvw == 'v' & selqr == 'q'
outvect1 = ['tau Ruu Rwv Rqq tplt Ruucplt RuuTplt RuuMplt ' ];
outvect2 = ['Rvvcplt RvvTplt RvvMplt Rqqcplt RqqTplt RqqMplt' ];
outvect = [outvect1 outvect2];
gdwrite('cor55uvq.asc2 asc2',outvect)

    elseif selup == 'p' & selvw == 'w' & selqr == 'r'
outvect1 = ['tau Rpp Rwv Rrr tplt Rppcplt RppTplt RppMplt ' ];
outvect2 = ['Rwwcplt RwwTplt RwwMplt Rrrcplt RrrTplt RrrMplt' ];
outvect = [outvect1 outvect2];
gdwrite('cor55pwr.asc2 asc2',outvect)

    end
```

Plots

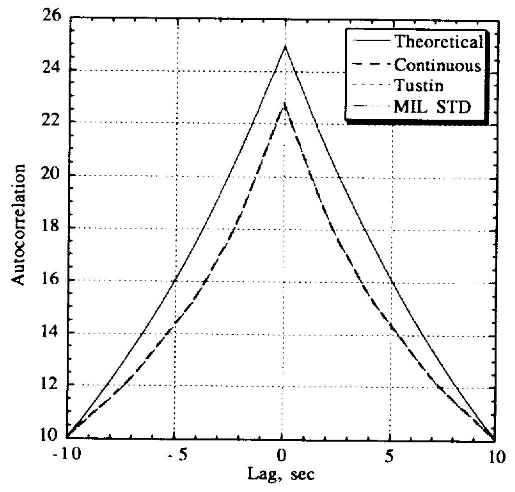
Example plots of autocorrelation functions produced by *A41plts.m* for $V = 100$ ft/sec and $V = 1000$ ft/sec are found in this section. The sample autocorrelation functions are for 819-sec turbulence sequences produced using GUSTMDL.



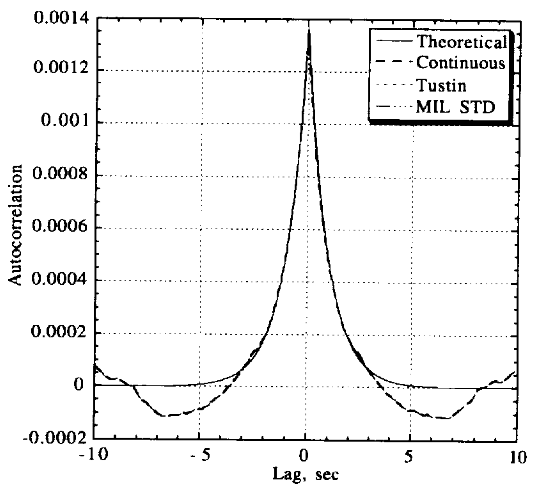
(a).- u-component



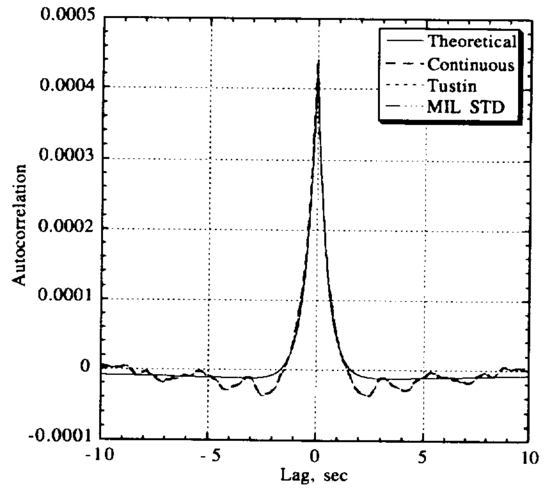
(b).- v-component



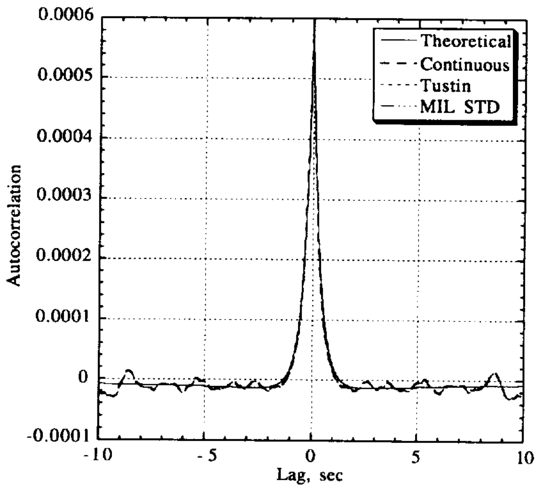
(c).- w-component



(d).- p-component

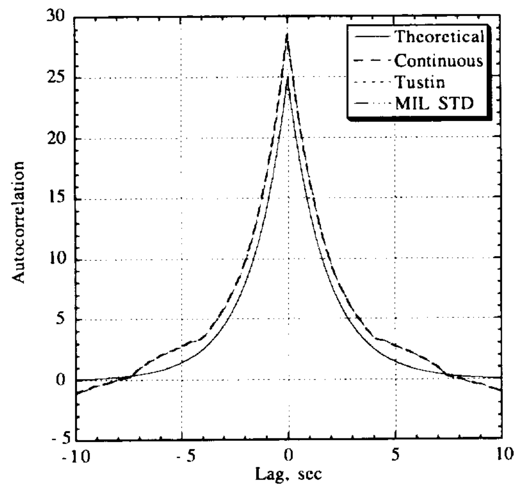


(e).- q-component

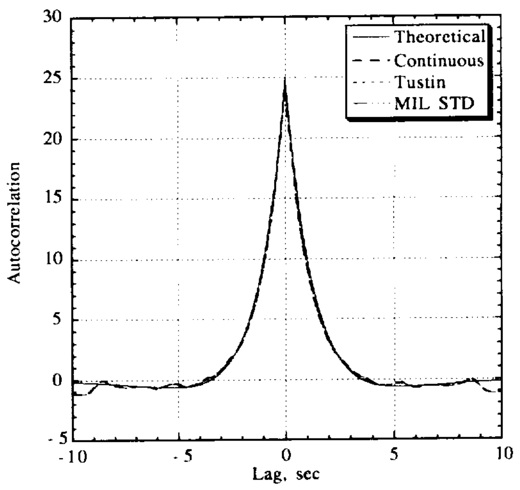


(f).- r-component

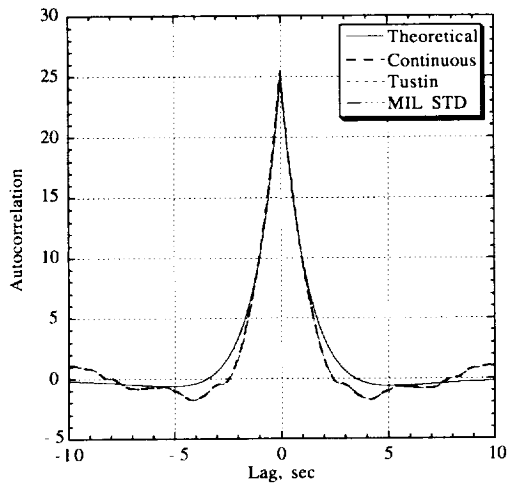
Figure 13. Autocorrelation functions for $V = 100$ ft/sec.



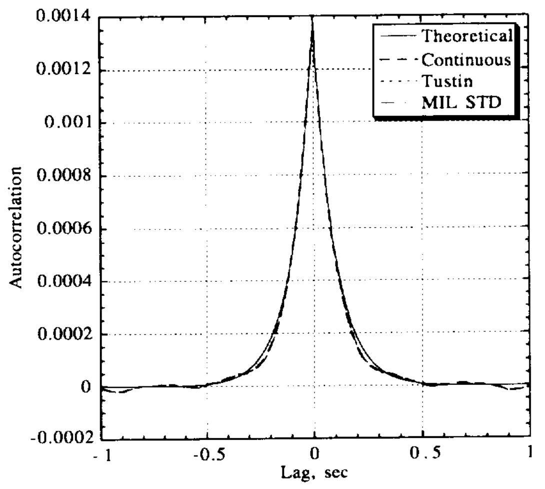
(a).- u-component



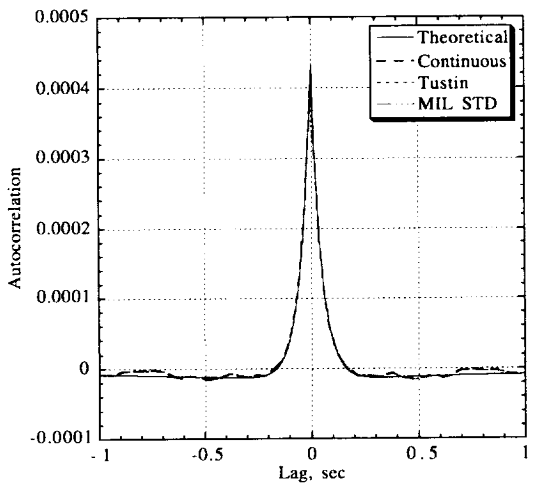
(b).- v-component



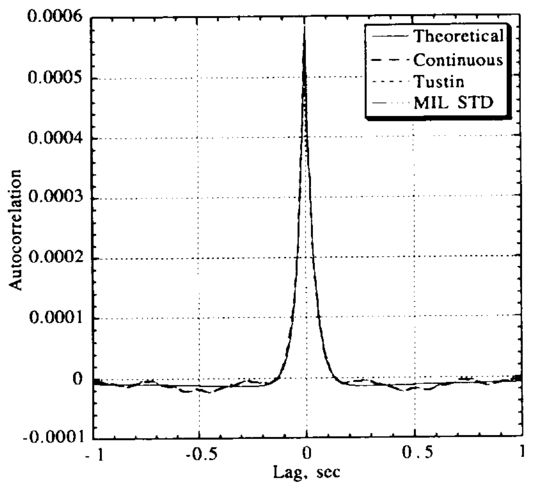
(c).- w-component



(d).- p-component



(e).- q-component



(f).- r-component

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
!              WTB, JCY 2-24-94
!
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'.

# #

```
      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.,     &
               TURBXVD0  = 0.,     &
               TURBXV0   = 0.,     &
               TURBXWD0  = 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
```

# #

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
!
   END$ !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.
         XDDWND  = VWNDRT*(HD/1000.)*COS(HEADWND*DG2RA)           &
                  - VELWND*SIN(HEADWND*DG2RA)*HDWRT*(HD/1000.)*DG2RA
         YDDWND  = - VWNDRT*(HD/1000.)*SIN(HEADWND*DG2RA)         &
                  -VELWND*COS(HEADWND*DG2RA)*HDWRT*(HD/1000.)*DG2RA
         ZDDWND  = 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 turbl= 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 * BWING) / 2.6                          ! eq.10
      TAUP = LP/VTOT                                           ! eq.13

      TURBOMEGA = VTOT/TURBL        ! same as 1./TAU,1./TAUV,1./TAUW
      WP = VTOT/LP                  ! same as 1./TAUP
!
!  CONSTANTS FOR CALCULATIONS OF CONTINUOUS W AND V            ! eq. 1,2
!
      TURBK2U  = TURBSIG*SQRT(2.*TURBOMEGA/TSAMP)
          K2P  = SIGP * SQRT(2. * WP/TSAMP)
      TURBK1VW = 2.*TURBOMEGA
      TURBK2VW = TURBOMEGA*TURBOMEGA
      TURBK3VW = TAU*SQRT(3.)
      TURBK4VW = TURBSIG*SQRT(TURBOMEGA**3/TSAMP)
!
!*********************************************************************!
!
!    TURBULENCE MODEL    WTB, JCY 2-24-94
!*********************************************************************!
!    U - COMPONENT
!*********************************************************************!
!
!   used in eq.5
!
      TURBUD = - TURBOMEGA * TURBU + TURBK2U * FILNU
      TURBU  = INTVC (TURBUD, TURBU0)
!*********************************************************************!
!    V - COMPONENT
!*********************************************************************!
!
!   used in eq.2
!
      TURBXVDD = - TURBK1VW * TURBXVD - TURBK2VW * TURBXV     &
                 + TURBK4VW * FILNV
      TURBXVD  = INTVC (TURBXVDD, TURBXVD0)
      TURBXVDI = TURBXVD
      TURBXV   = INTVC (TURBXVDI, TURBXV0)
      TURBV    = TURBXV + TURBK3VW * TURBXVDI
      TURBVD   = TURBXVDI + TURBK3VW * TURBXVDD
!*********************************************************************!
!    W - COMPONENT
!*********************************************************************!
!
!   used in eq.2
!
      TURBXWDD = - TURBK1VW * TURBXWD - TURBK2VW * TURBXW     &
                 + TURBK4VW * FILNW
      TURBXWD  = INTVC (TURBXWDD, TURBXWD0)
      TURBXWDI = TURBXWD
      TURBXW   = INTVC (TURBXWDI, TURBXW0)
      TURBW    = TURBXW + TURBK3VW * TURBXWDI
      TURBWD   = TURBXWDI + TURBK3VW * TURBXWDD
!*********************************************************************!
!    P, Q, R COMPONENTS  ADDED          5-15-95 JCY
```

```
!**********************************************************************!
!    P - COMPONENT
!**********************************************************************!
!
!   used in eq.7
!
      TURBPD = - WP * TURBP + K2P * FILNP
      TURBP  = INTVC (TURBPD, TURBP0)
!**********************************************************************!
!    Q - COMPONENT
!**********************************************************************!
!
!   used in eq.12
!
      TURBQD = - TURBQ / TAUQ + TURBWD / (TAUQ * VTOT)
      TURBQ  = INTVC (TURBQD, TURBQ0)
!**********************************************************************!
!    R - COMPONENT
!**********************************************************************!
!
!   used in eq.15
!
      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.

# #

```
      PROCEDURAL(wwg=wwnd, TURBU, TURBV, TURBW, TURBP, TURBQ, TURBR)
       IF (TURBFLG .EQ. 'TUS' .AND. T .GT. 0.) THEN
          UWG = FILU
          VWG = FILV
          WWG = FILW
          PWG = FILP
          QWG = FILQ
          RWG = FILR
         IF  (.not. PQRFLG)  THEN
             PWG = 0.
             QWG = 0.
             RWG = 0.
          END IF
          UDWG = FILUD2
          VDWG = FILVD2
          WDWG = FILWD2
       ELSE IF  (TURBFLG .EQ. 'MIL' .AND. T .GT. 0.)  THEN
          UWG = MILUK
          VWG = MILVK
          WWG = MILWK
          PWG = MILPK
          QWG = MILQK
```

```
            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)
!------- d/dt Vb = - [omega] Vb    + [L](d/dt Ve)
!------------------------------------------------------------------------
        UDWND =  r*vwg - q*wwg + CXX*XDDWND + CXY*YDDWND + CXZ*ZDDWND
        VDWND = -r*uwg + p*wwg + CYX*XDDWND + CYY*YDDWND + CYZ*ZDDWND
        WDWND =  q*uwg - p*vwg + CZX*XDDWND + CZY*YDDWND + CZZ*ZDDWND
        UA  = U - UWND - UWG
        VA  = V - VWND - VWG
        WA  = W - WWND - WWG
        UDA = UD - UDWND - UDWG
        VDA = VD - VDWND - VDWG
        WDA = WD - WDWND - WDWG
```

```
         UA2 = UA*UA
         VA2 = VA*VA
         WA2 = WA*WA
```

# #

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
!
         PA = P + PWG
         QA = Q + QWG
         RA = R + RWG
!
     PROCEDURAL(TMPK=H)
          CALL ATMAT62(TMPK,H,3)
!
     TMPKOLD = TMPK
     TMPK = TMPK + DELTK
!
     END $!OF PROCEDURAL TO CALCULATE TEMPERATURE IN KELVIN!
!
! ------ COMPUTE WIND AXES VARIABLES !

         CSALF = COS(ALF)
         CSBET = COS(BET)
         SNALF = SIN(ALF)
         SNBET = SIN(BET)

        PW = ( P*CSALF*CSBET + R*SNALF*CSBET + (Q-ALFD)*SNBET )
        QW = (-P*CSALF*SNBET - R*SNALF*SNBET + (Q-ALFD)*CSBET )
        RW = (-P*SNALF       + R*CSALF       + BETD           )
        PWDG = PW * RA2DG
        QWDG = QW * RA2DG
        RWDG = RW * RA2DG
        AYW =  VT * RW
        AZW = -VT * QW
```

# #

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 ,CZRFSF ,C1RFSF ,CMRFSF ,CNRFSF              &
  ,CDRFSF ,CLRFSF ,TVJPAVG,TVJYAVG,TVJRAVG                     &
  ,cno    ,cnbeta ,cnlef  ,cntef  ,cnail  ,cnrud  ,cnraero&
  ,cnp    ,clo    ,clbeta ,cllef  ,cltef  ,clail  ,clrud   &
```

```
     ,clr     ,clp     ,dclflx                                        &
     =ALDGRF  ,BEDGRF  ,MACHRF  ,PA      ,QA      ,RA      ,HRF      &
     ,HGC     ,QBAR    ,CWRF    ,BWRF    ,VTRF                       &
     ,DSR     ,DSL     ,DAR     ,DAL     ,DRR     ,DRL     ,DFR      &
     ,DFL     ,DNR     ,DNL     ,DSB     ,DLG                        &
     ,LDBA    ,LQSE    ,LRTE    ,SWRF    ,                           &
      FGAERO  ,TVJANP  ,TVJANY  ,XSTRAKE,DG2RA   ,DNSL    ,DNSR      &
     ,H)


! ------ CALL SFAERRF TO USE THE R/T AERO DATA     E FOR RIGID MOTION. !
!           SFAERRF INTERPRETS LEFT RUDDER AS POS T.E.LEFT.  THIS IS  !
!           IN ACCORDANCE WITH MCAIR CONVENTION.                      !
!
! Altitude input to SFAERRF has been changed from pressure altitude
! to true altitude.  This was done to match the implementations in
! the DMS, ACES, and Dryden simulations.
!
      CALL SFAERRF(                                                  &
      CDRFSF ,CYRFSF ,CLRFSF ,C1RFSF ,CMRFSF ,CNRFSF                 &
     ,ALDGRF ,BEDGRF ,MACHRF ,PA     ,QA     ,RA     ,H              &
     ,HGC    ,QBAR   ,CWRF   ,BWRF   ,VTRF                           &
     ,DSR    ,DSL    ,DAR    ,DAL    ,DRR    ,DRL    ,DFR            &
     ,DFL    ,DNR    ,DNL    ,DSB    ,DLG                            &
     ,LDBA   ,LQSE   ,LRTE   ,cno    ,cnbeta ,cnlef  ,cntef          &
     ,cnail  ,cnrud  ,cnraero,cnp    ,clo    ,clbeta ,cllef          &
     ,cltef  ,clail  ,clrud  ,clr    ,clp    ,dclflx)
!                                                                   !
      CALL AEROINC (                                                  &
        CDRFSF ,CYRFSF ,CLRFSF ,C1RFSF , CMRFSF , CNRFSF,             &
        ALDGRF ,BEDGRF ,MACHRF ,                                      &
        QBAR   ,SWRF   ,FGAERO ,TVJANP , TVJANY ,                     &
        DSR    ,DSL    ,DRR    ,DRL    , TVJPAVG, TVJYAVG,            &
        TVJRAVG,PA     ,QA     ,RA     , CWRF   , BWRF   ,            &
        VTRF   ,XSTRAKE,DG2RA  ,DNSL   , DNSR   ,                     &
        DCNFS  )
!
# # # # # # # # # # # # # # # # # # # # # # #  # # # # # # # # # # # # #
```

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).

```
# # # # # # # # # # # # # # # # # # # # # # #  # # # # # # # # # # # # #


!
!*********************************************************************!
!
   DISCRETE GUST
!*********************************************************************!
!
         INTERVAL TSGAUS = 0.0125
!*********************************************************************!
!    GAUSSIAN RANDOM PROCESS
```

```
!
!**********************************************************************!
         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
!
!**********************************************************************!
!
!  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
!
      KFILD1  = 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
      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
!
!    eq.18
!
         GUFILK = -FILK1*GUFILKM1 + TWOPIOVTNU*FILK2*(UFILK + UFILKM1)


!**********************************************************************!
! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF U-COMPONENT
!**********************************************************************!
!
!    eq.27
!
            FILUD = (GUFILK - GUFILKM1)/TSAMP
!
!    eq.28
!
            FILUD2 = - KWD1 * filud2km1                    &
                     + (kfild1*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.
            FILVD2KM1 = 0.
            FILVD2KM2 = 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*FILVD2KM1                          &
                  - KWD1**2*FILVD2KM2                          &
                  + kwd2*((1.+CTWSR3)*VFILK                    &
                  - (2.*CTWSR3)*VFILKM1 -(1.-CTWSR3)*VFILKM2)   ! 5-12-97
!
            GVFILKM2 = GVFILKM1
            GVFILKM1 = GVFILK
            VFILKM2 = VFILKM1
            VFILKM1 = VFILK
            FILVD2KM2 = FILVD2KM1
            FILVD2KM1 = FILVD2
         FILVDKM1 = FILVD
         FILV = GVFILK
!*********************************************************************!
!    W-COMPONENT
!*********************************************************************!
         WFILK = FILNW
         IF (T .EQ. 0.)  THEN
            WFILKM1 = 0.
            WFILKM2 = 0.
            GWFILKM1 = 0.
            GWFILKM2 = 0.
            FILWDKM1 = 0.
            FILWD2KM1 = 0.
            FILWD2KM2 = 0.
         ENDIF
            FILWKM1 = GWFILKM1
!
!    eq.20
!
            GWFILK = - FILK4*GWFILKM1 - FILK5*GWFILKM2                 &
                     + TWOPIOVTNU*FILK9*(FILK6*WFILK + FILK7*WFILKM1 &
                     + FILK8*WFILKM2)
!*********************************************************************!
! DIGITAL IMPLEMENTATION OF DERIVATIVE CALCULATIONS OF W-COMPONENT
!*********************************************************************!
!
!    eq.27
!
         FILWD = (GWFILK - GWFILKM1)/TSAMP
```

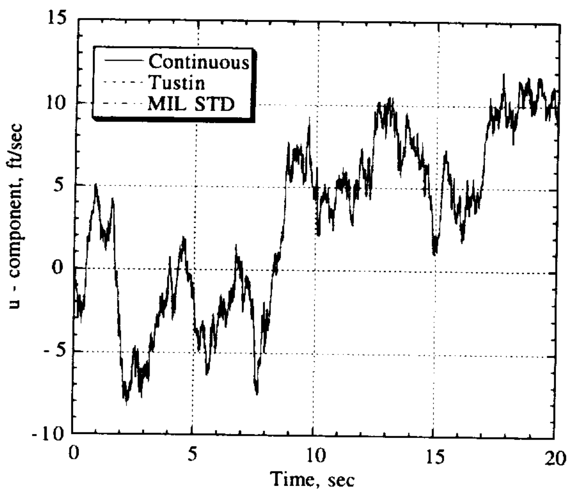
```
!
!    eq.29
!
        FILWD2 =  -2.*KWD1*FILWD2KM1                        &
                - KWD1**2*FILWD2KM2                         &
                + kwd2*((1.+CTWSR3)*WFILK                   &
                - (2.*CTWSR3)*WFILKM1 -(1.-CTWSR3)*WFILKM2)  !5-12-97
!
           GWFILKM2 = GWFILKM1
           GWFILKM1 = GWFILK
           WFILKM2 = WFILKM1
           WFILKM1 = WFILK
           FILWD2KM2 = FILWD2KM1
           FILWD2KM1 = FILWD2
        FILWDKM1 = FILWD
        FILW = GWFILK
!*********************************************************************!
!  P, Q, R COMPONENT ADDED  5-15-95 JCY
!*********************************************************************!
!    P-COMPONENT
!*********************************************************************!
        PFILK = FILNP
        IF (T .EQ. 0.)  THEN
           PFILKM1 = 0.
           GPFILKM1 = 0.
        ENDIF
           GPFILK  = -PK1*GPFILKM1 + PK2*(PFILK + PFILKM1)   ! eq. 21
           PFILKM1 = PFILK                                   ! 5-2-97
           GPFILKM1 = GPFILK                                 ! 5-2-97
        FILP = GPFILK
!*********************************************************************!
!    Q-COMPONENT
!*********************************************************************!
        IF (T .EQ. 0.)  THEN
           FILQKM1 = 0.
        ENDIF
           FILQ    = -QK1*FILQKM1 + QK2*(FILW - FILWKM1)     ! eq. 23
           FILQKM1 = FILQ
!
!*********************************************************************!
!    R-COMPONENT
!*********************************************************************!
        IF (T .EQ. 0.)  THEN
           FILRKM1 = 0.
        ENDIF
           FILR    = -RK1*FILRKM1 + RK2*(FILV - FILVKM1)     ! 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
!                                                              8-28-97
!*********************************************************************!
        IF (T .EQ. 0.)  THEN
           MILUKM1 = 0.
           MILVKM1 = 0.
           MILWKM1 = 0.
           MILPKM1 = 0.
           MILQKM1 = 0.
           MILRKM1 = 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*MILQKM1 + piov4b*(milwk-milwkm1)          ! eq.34
     MILRK = milk8*MILRKM1 + piov3b*(milvk-milvkm1)          ! 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
           MILQKM1 = MILQK
           MILRKM1 = 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(DRMSPWG, DMNPWG, PWG , SAMPS)
         DRMS(DRMSQWG, DMNQWG, QWG , SAMPS)
         DRMS(DRMSRWG, DMNRWG, RWG , SAMPS)
!
         DRMS(DRMSUDWG, DMNUDWG, UDWG , SAMPS)
         DRMS(DRMSVDWG, DMNVDWG, VDWG , SAMPS)
         DRMS(DRMSWDWG, DMNWDWG, 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
!**********************************************************************!
```

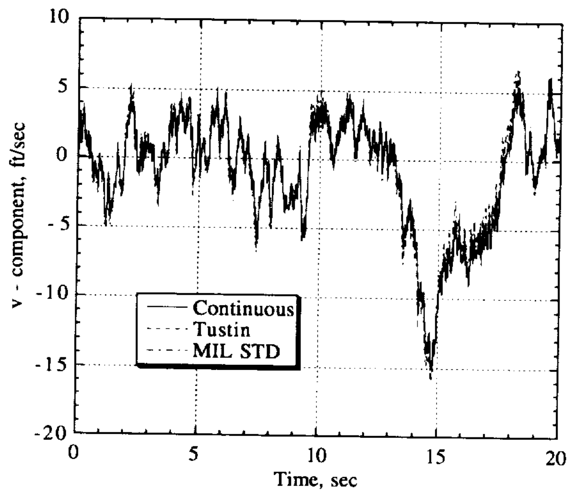
# #

Time History Plots

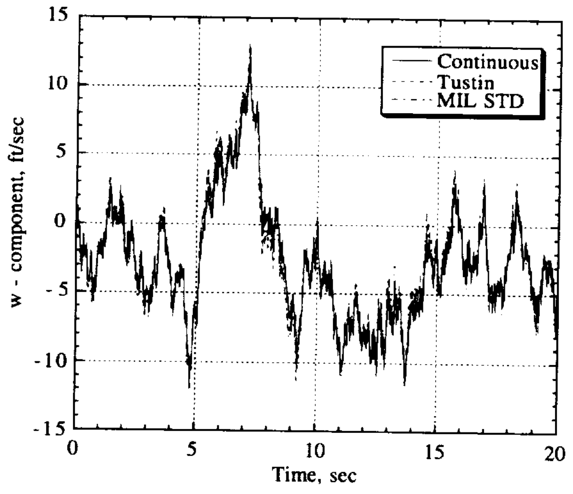
Numerous runs were made with the modified HARV aircraft simulation to evaluate the performance of the implemented turbulence models. The 20-second runs were made with five trim cases. Some runs set the *PQRFLG* variable to .True. to use effects of p, q, and r gust effects, while other runs had a value 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 α values of 5°, 25°, 35°, 55°, and 60°. Figures 15 through 23 are the time histories for the α = 5° trim case. Figures 24 through 32 are the time histories for the α = 25° trim case. Figures 33 through 41 are the time histories for the α = 35° trim case. Figures 42 through 50 are the time histories for the α = 55° trim case. Figures 51 through 59 are the time histories for the α = 60° trim case.



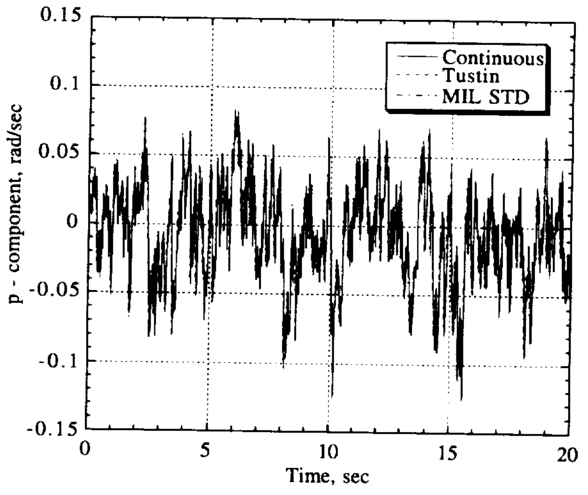
(a).- u-component.



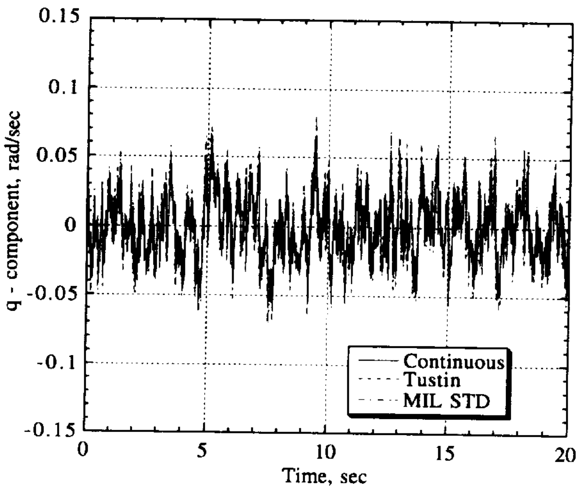
(b).- v-component.



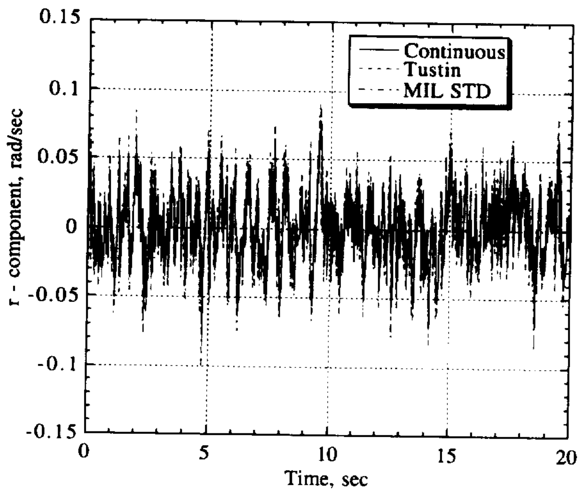
(c).- w-component.



(d).- p-component.

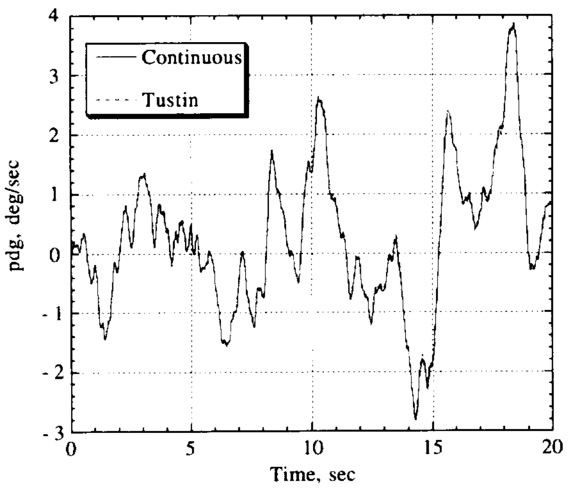


(e).- q-component.

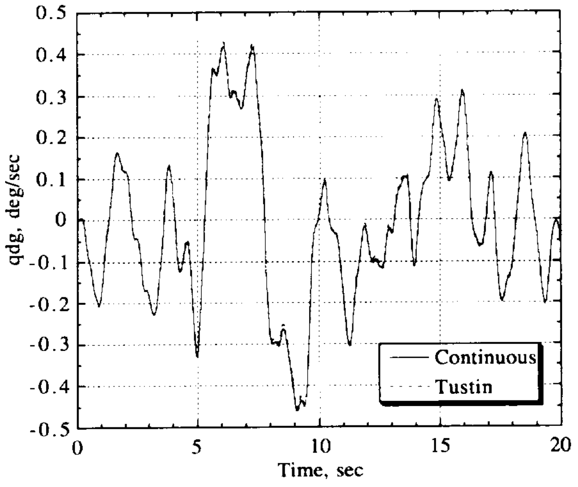


(f).- r-component.

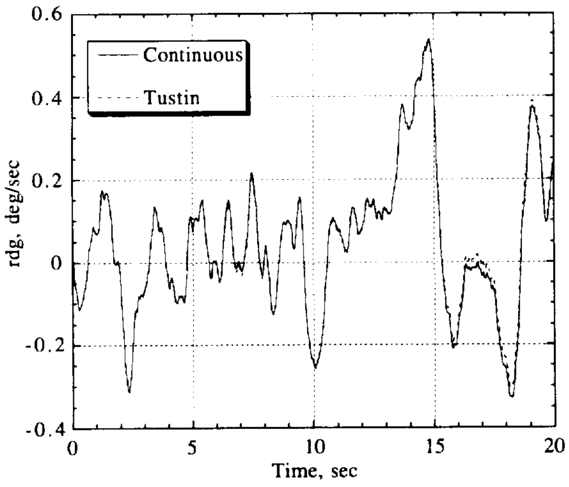
Figure 15. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 5°$, seed no. 1.



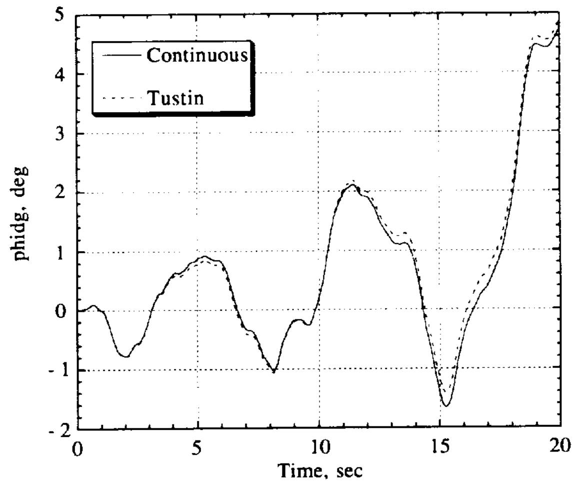
(a).- Roll rate.



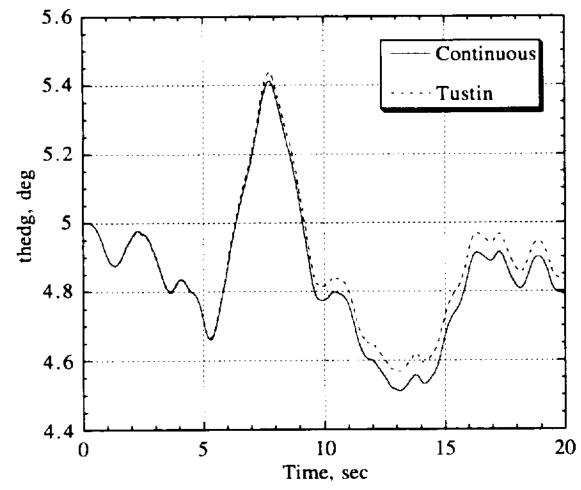
(b).- Pitch rate.



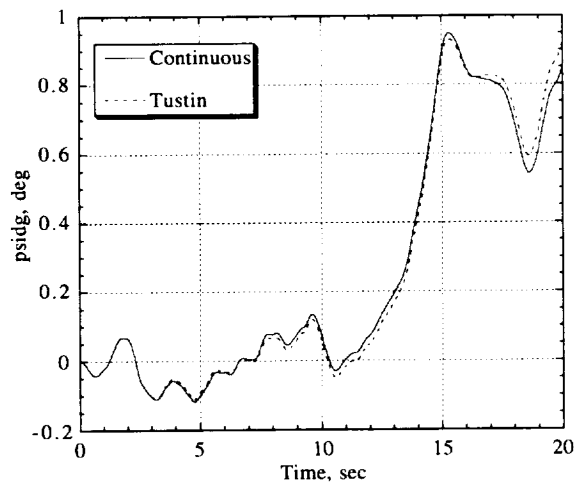
(c).- Yaw rate.



(d).- Bank angle.

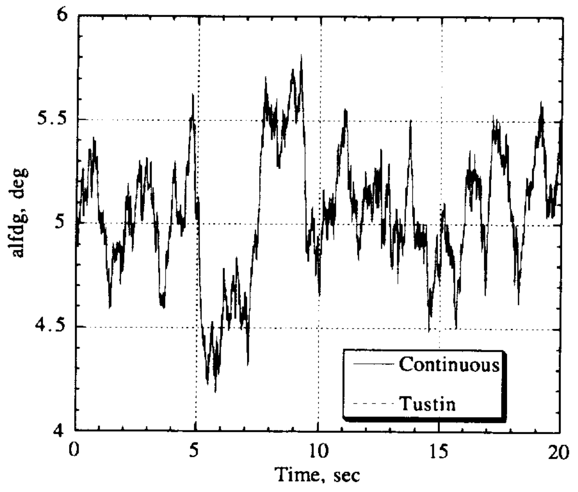


(e).-Pitch angle.

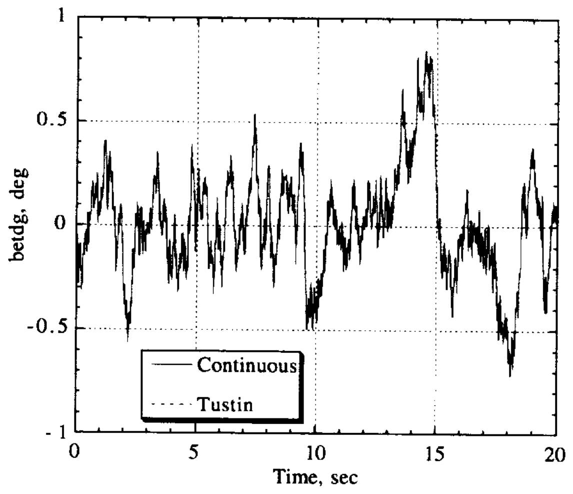


(f).-Heading.

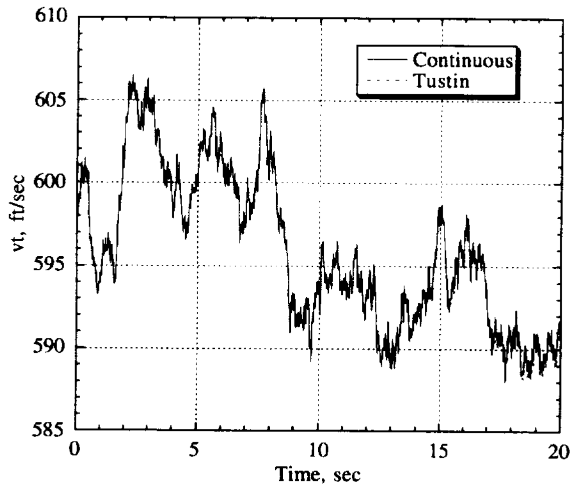
Figure 16. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 5°$, seed no. 1.



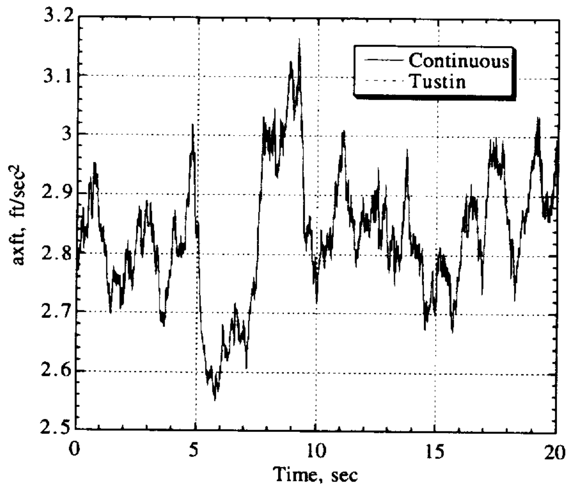
(a).- Angle of attack.



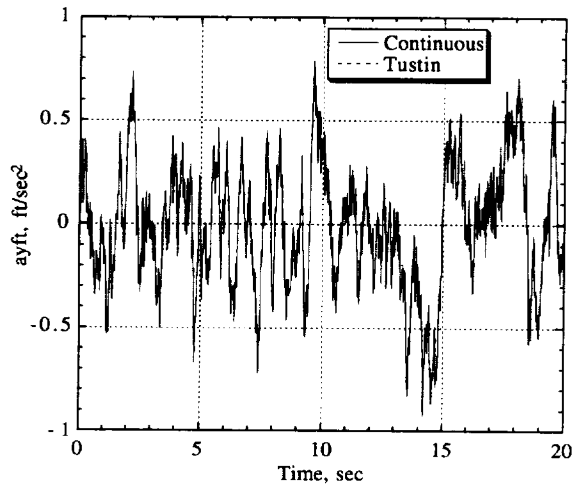
(b).- Sideslip angle.



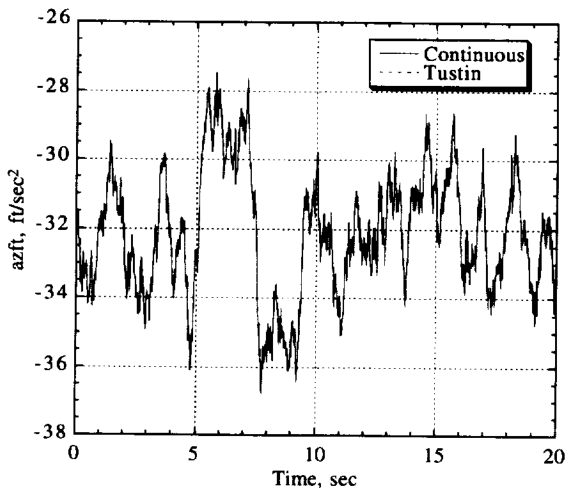
(c).- True airspeed.



(d).- X-axis acceleration.

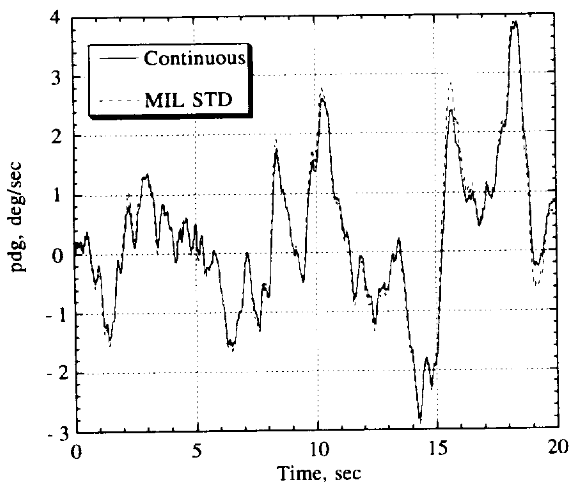


(e).- Y-axis acceleration.

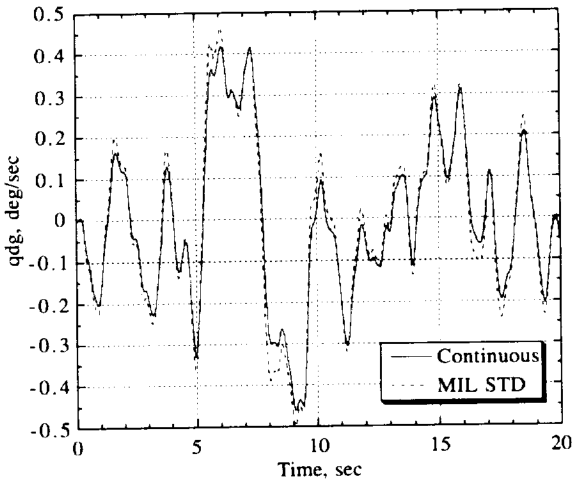


(f).- Z-axis acceleration.

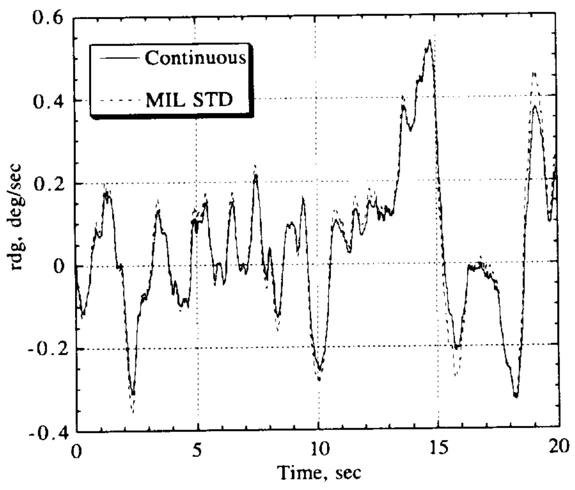
Figure 17. Comparison of continuous and Tustin air data and accelerations for $\alpha = 5°$, seed no. 1.



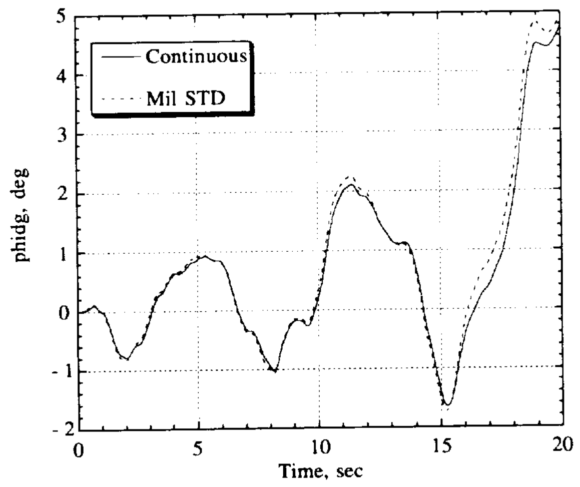
(a).- Roll rate.



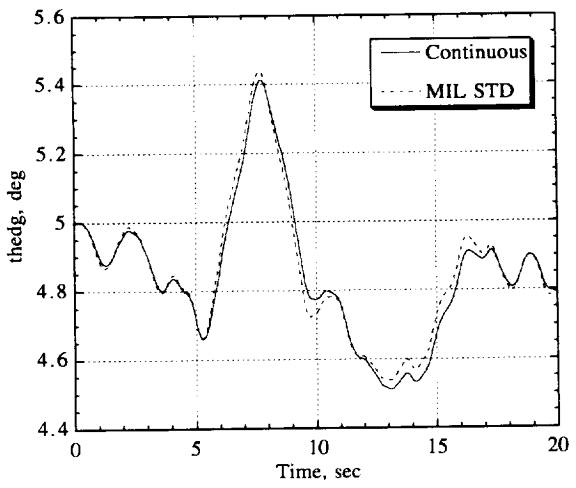
(b).- Pitch rate.



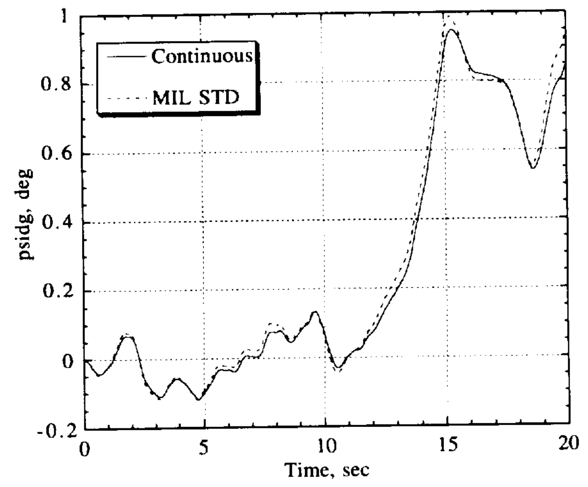
(c).- Yaw rate.



(d).- Bank angle.

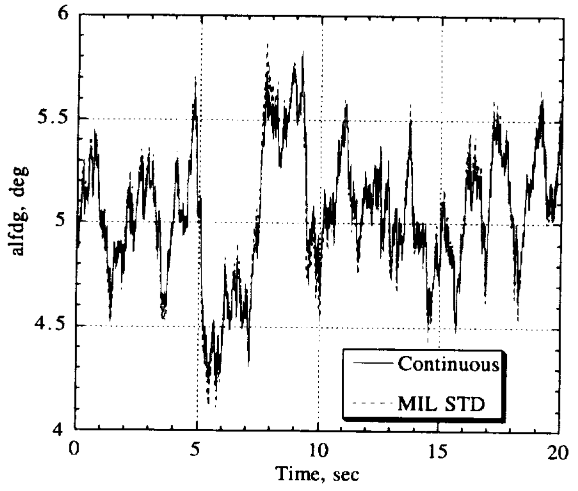


(e).- Pitch angle.

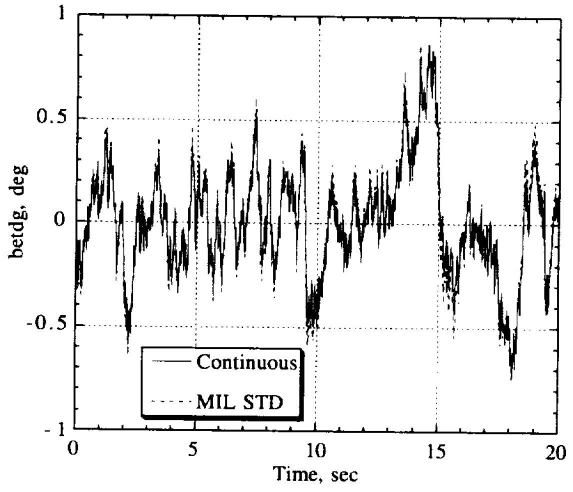


(f).- Heading.

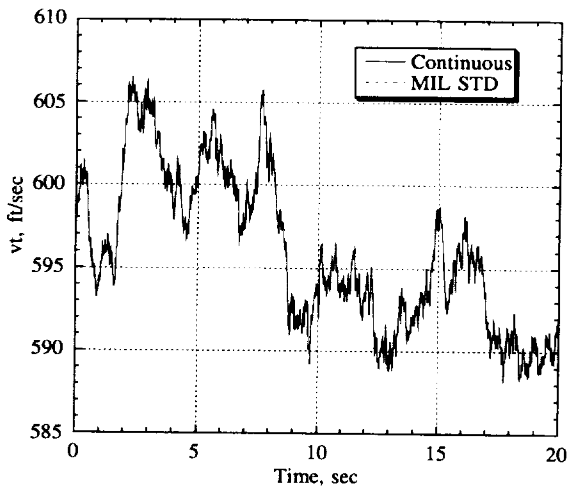
Figure 18. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 5°$, seed no. 1.



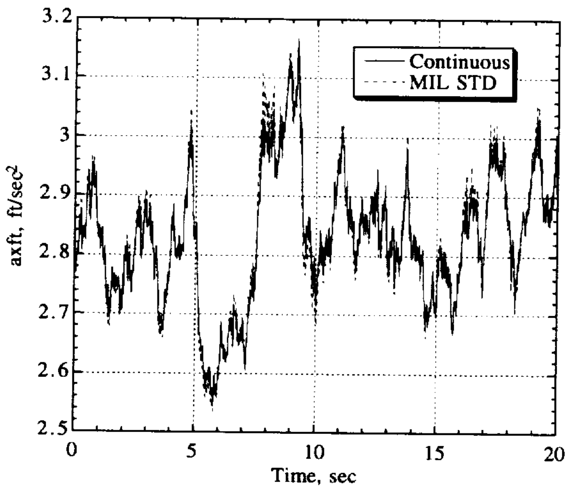
(a).- Angle of attack.



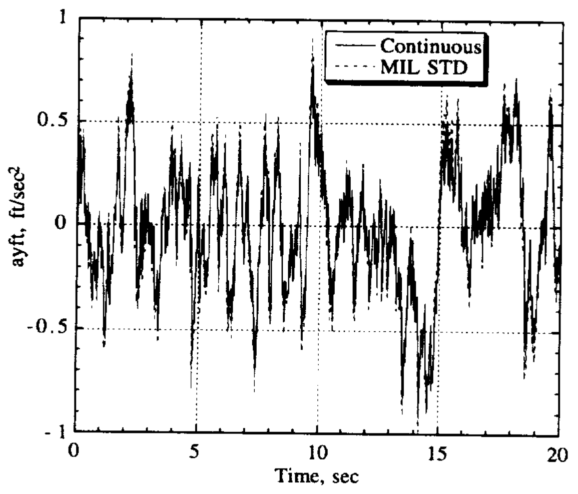
(b).- Sideslip angle.



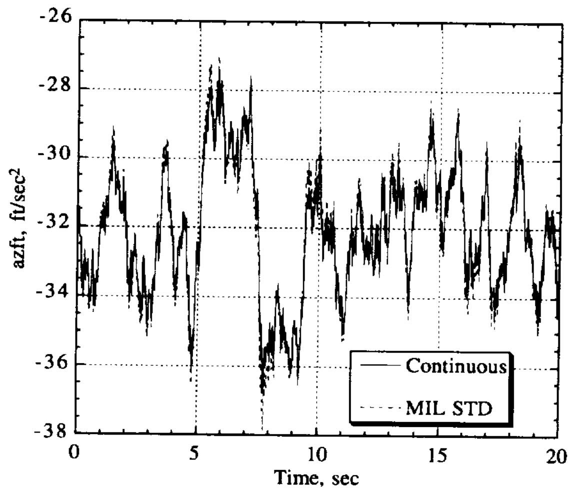
(c).- True airspeed.



(d).- X-axis acceleration.

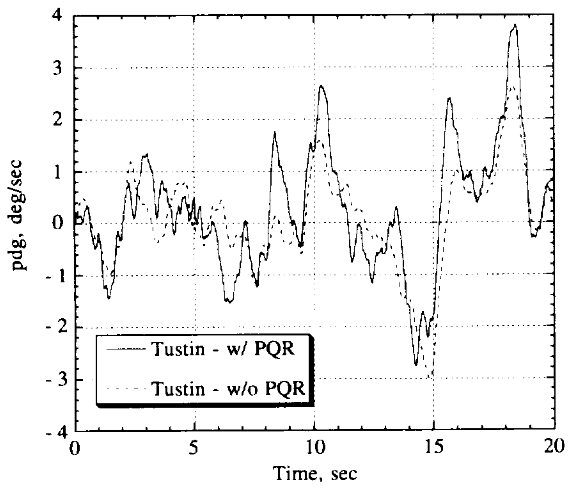


(e).- Y-axis acceleration.

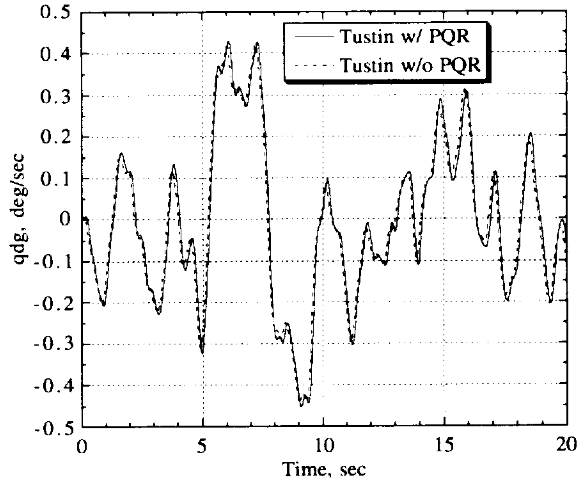


(f).- Z-axis acceleration.

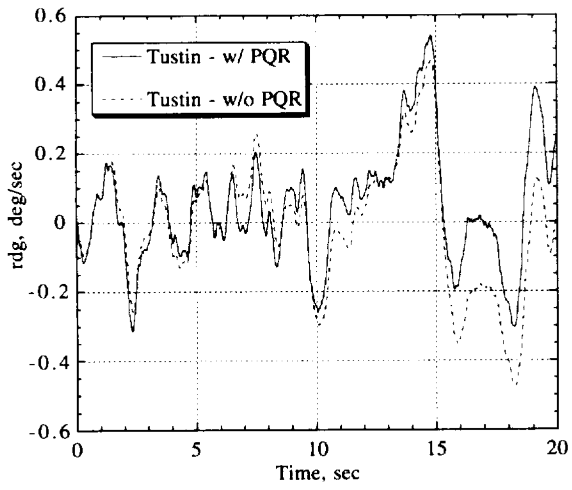
Figure 19. Comparison of continuous and Tustin air data and accelerations for $\alpha = 5°$, seed no. 1.



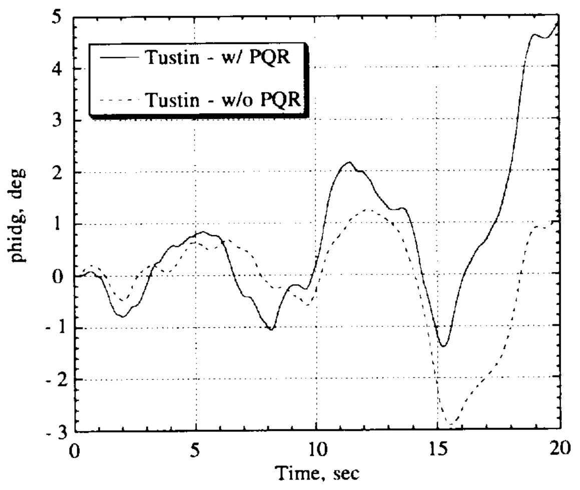
(a).- Roll rate.



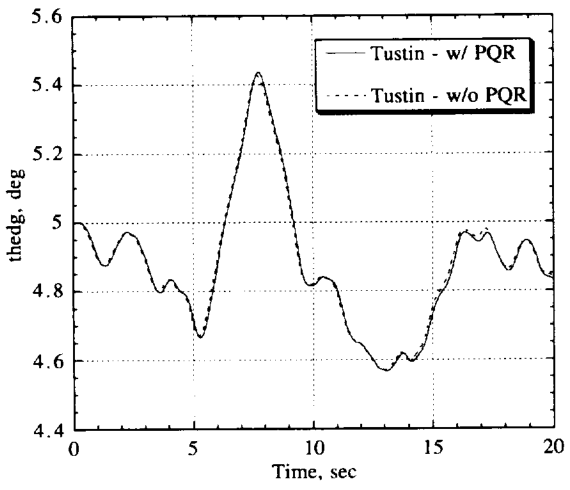
(b).- Pitch rate.



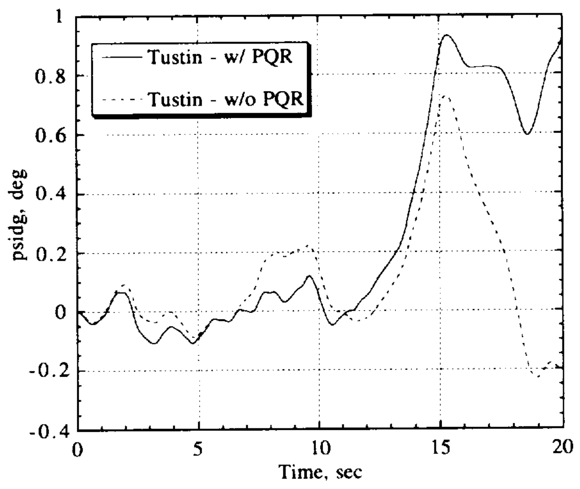
(c).- Yaw rate.



(d).- Bank angle.

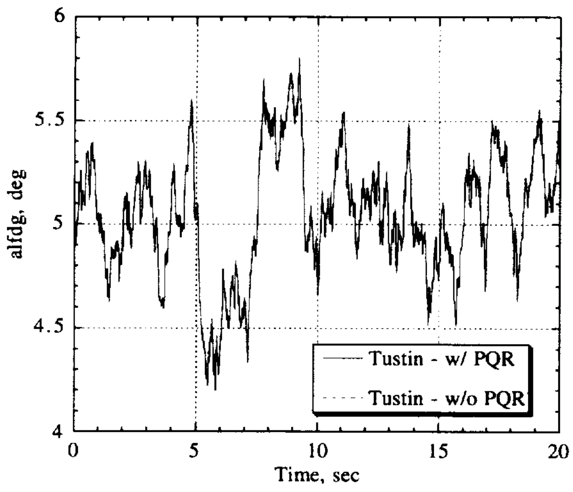


(e).- Pitch angle.

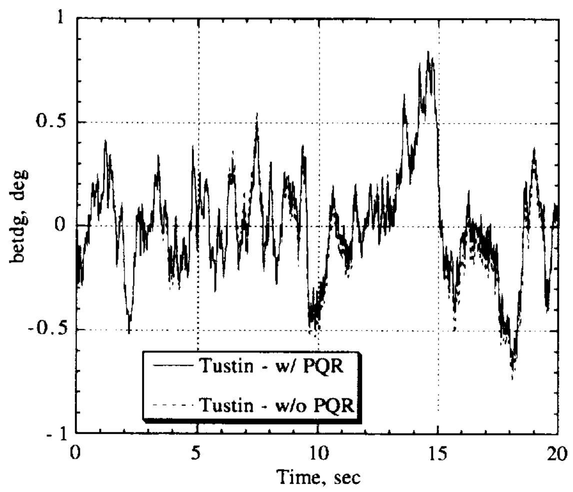


(f).- Heading.

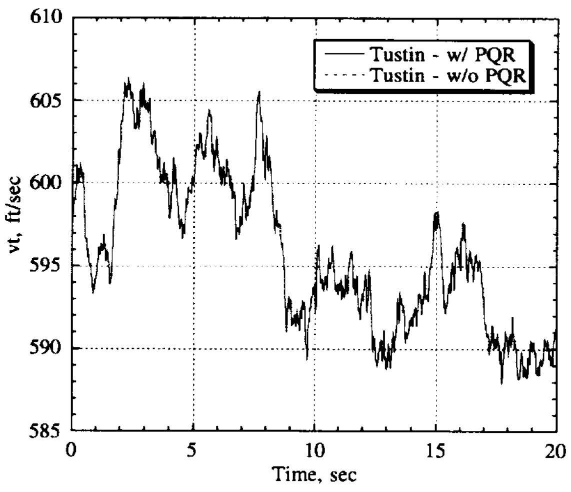
Figure 20. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 5°$, seed no. 1.



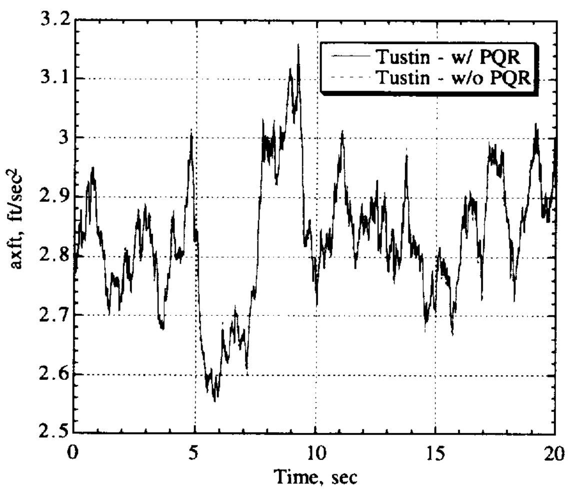
(a).- Angle of attack.



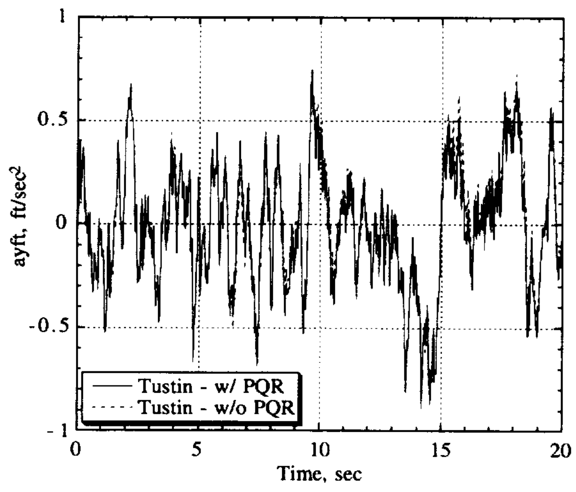
(b).- Sideslip angle.



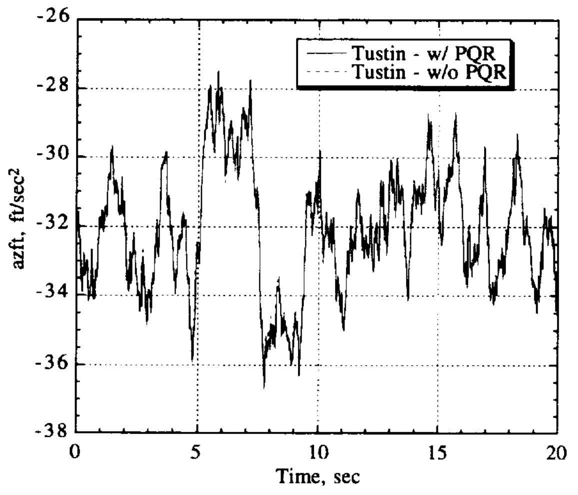
(c).- True airspeed.



(d).- X-axis acceleration.

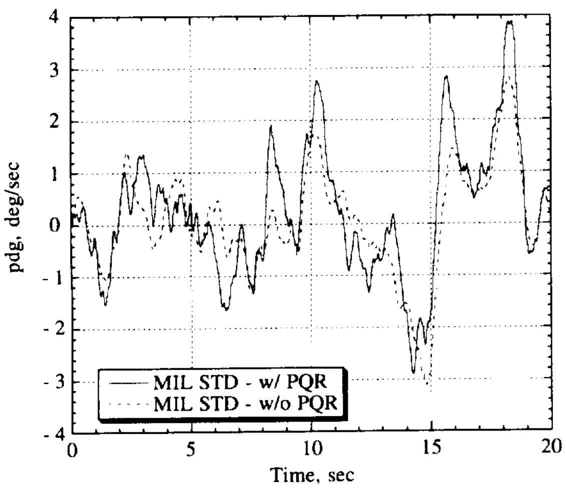


(e).- Y-axis acceleration.

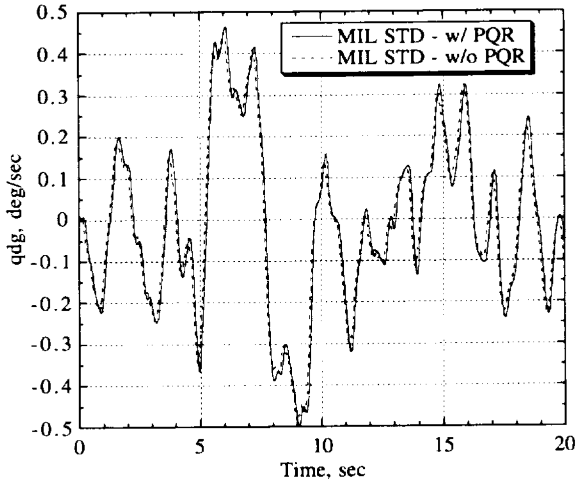


(f).- Z-axis acceleration.

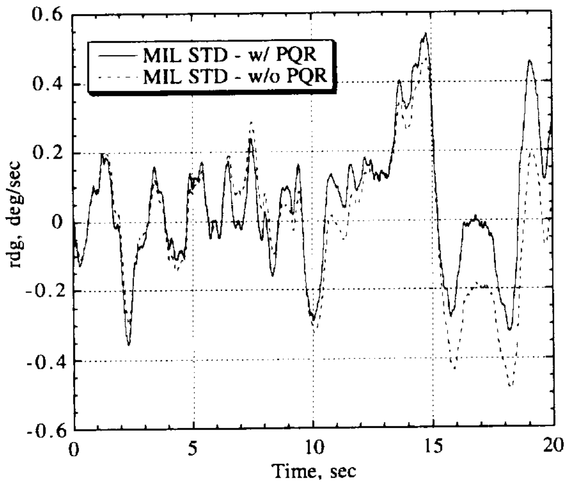
Figure 21. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 5°$, seed no. 1.



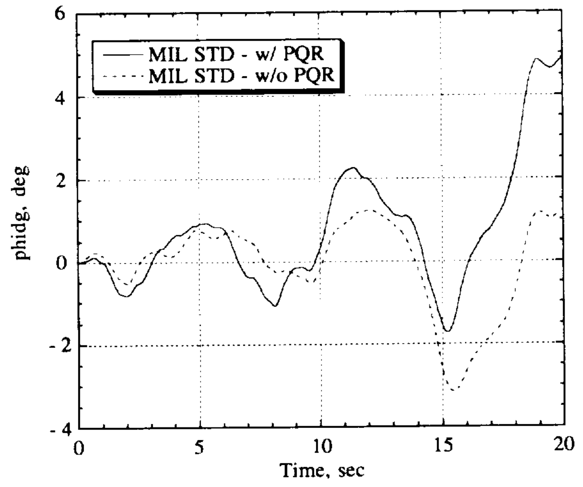
(a).- Roll rate.



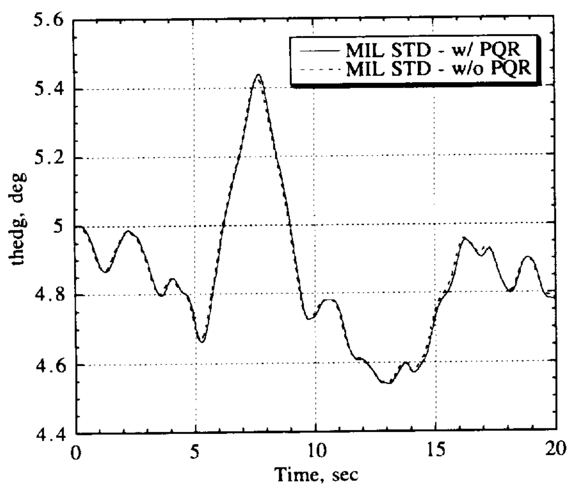
(b).- Pitch rate.



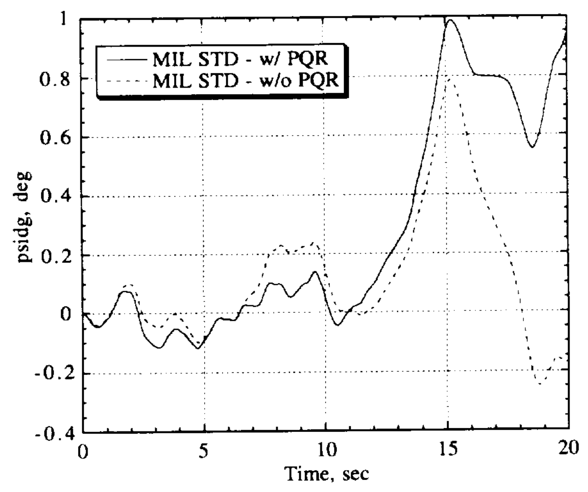
(c).- Yaw rate.



(d).- Bank angle.

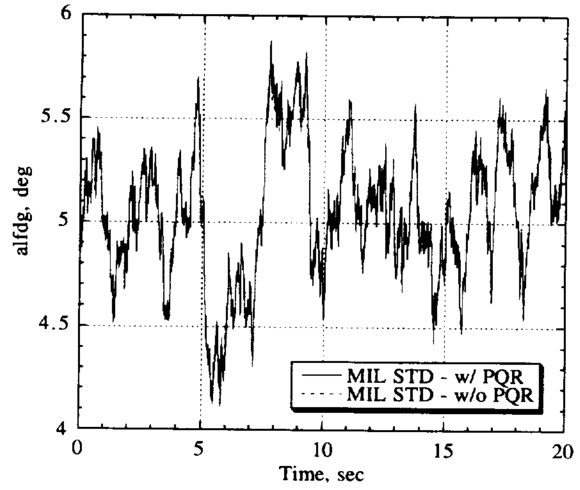


(e).- Pitch angle.

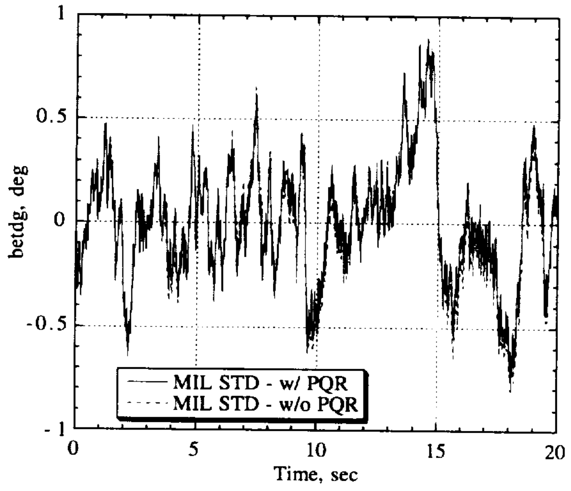


(f).- Heading.

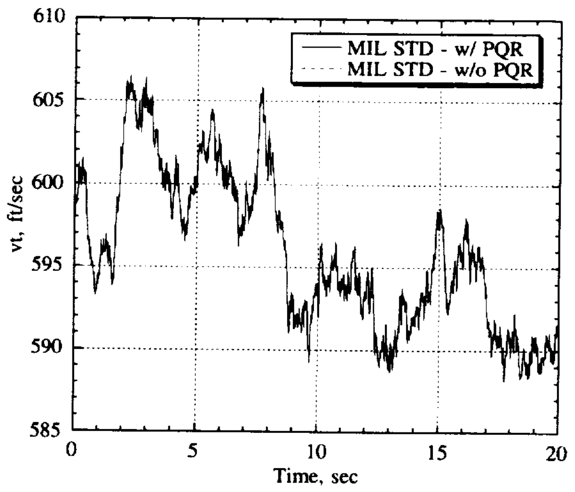
Figure 22. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 5°$, seed no. 1.



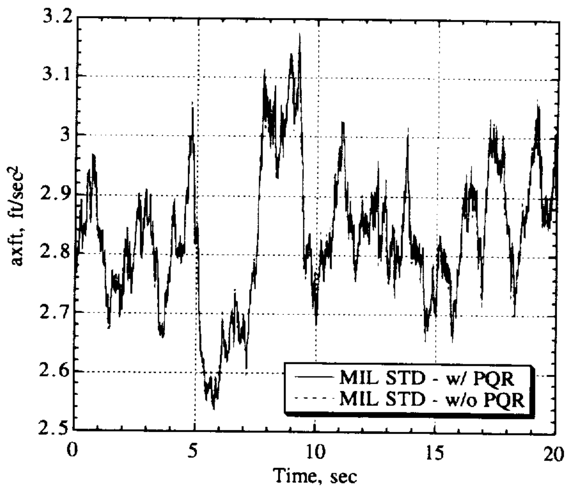
(a).- Angle of attack.



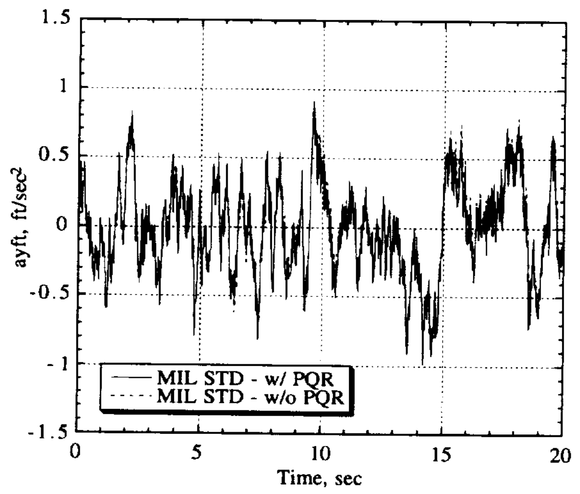
(b).- Sideslip angle.



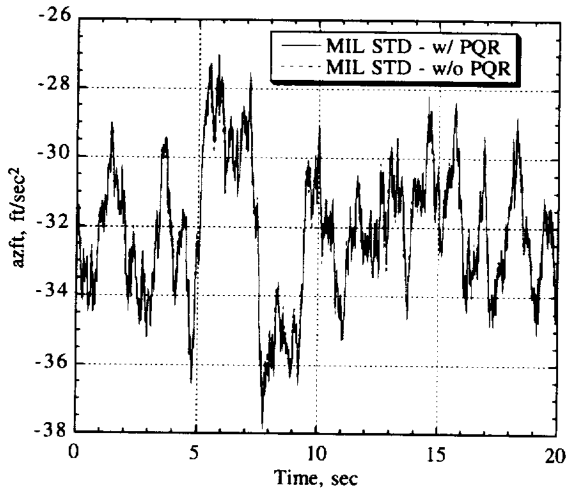
(c).- True airspeed.



(d).- X-axis acceleration.

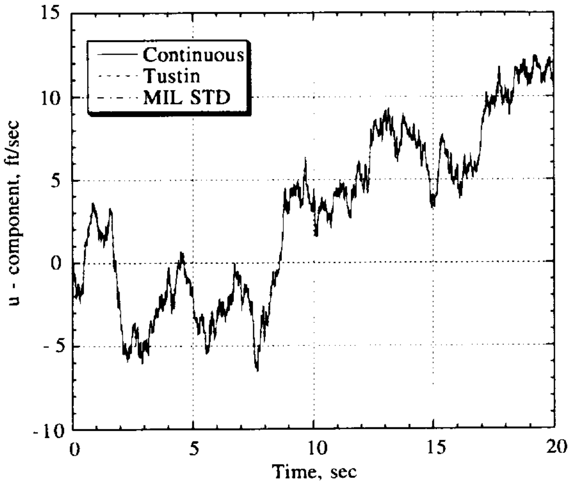


(e).- Y-axis acceleration.

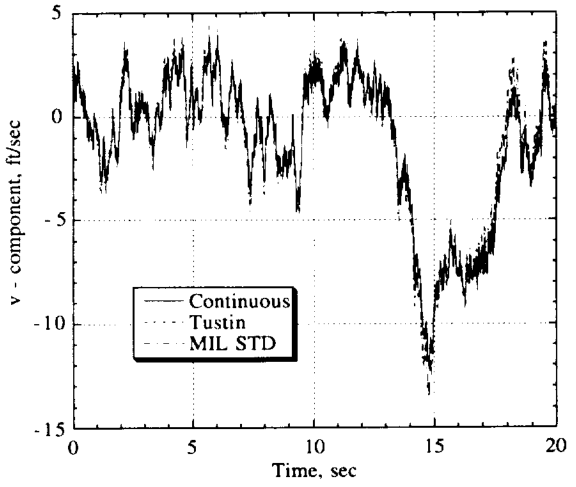


(f).- Z-axis acceleration.

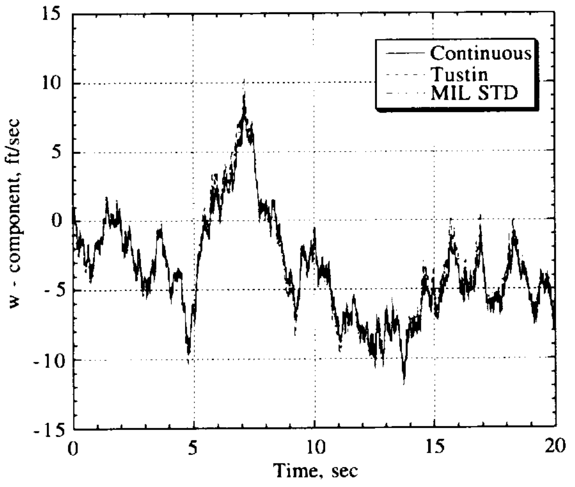
Figure 23. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 5°$, seed no. 1.



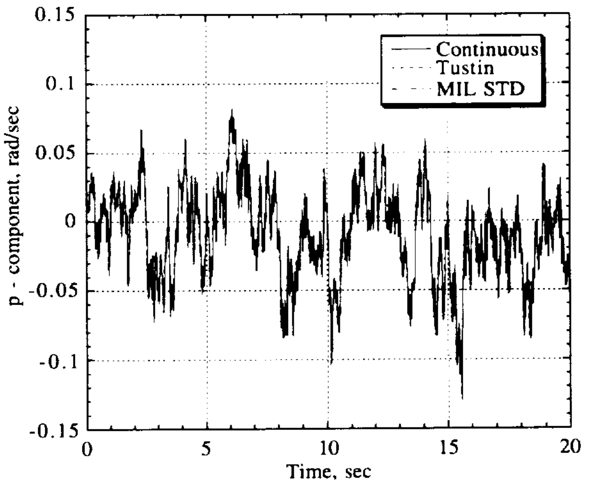
(a).- u-component.



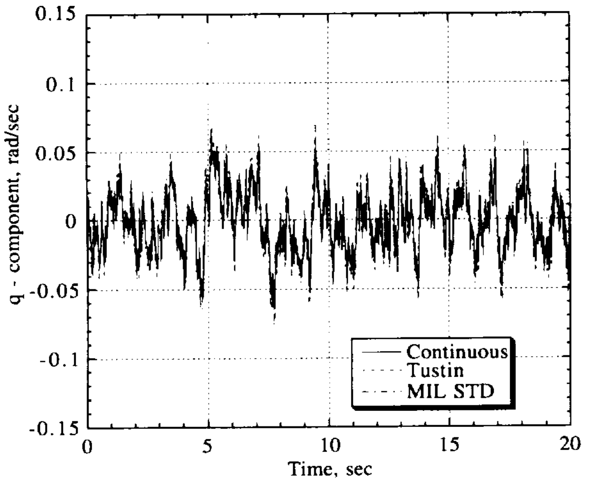
(b).- v-component.



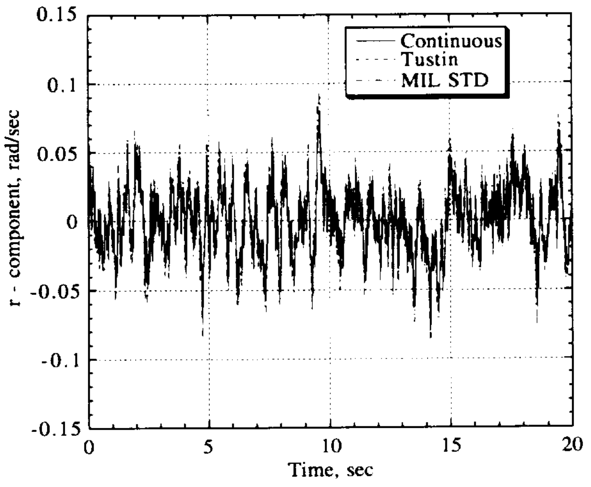
(c).- w-component.



(d).- p-component.

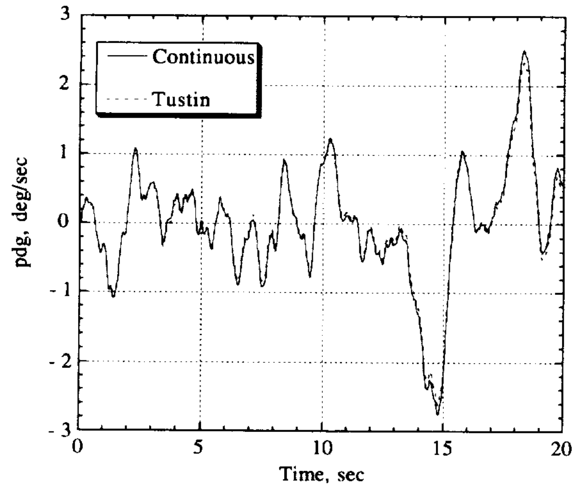


(e).- q-component.

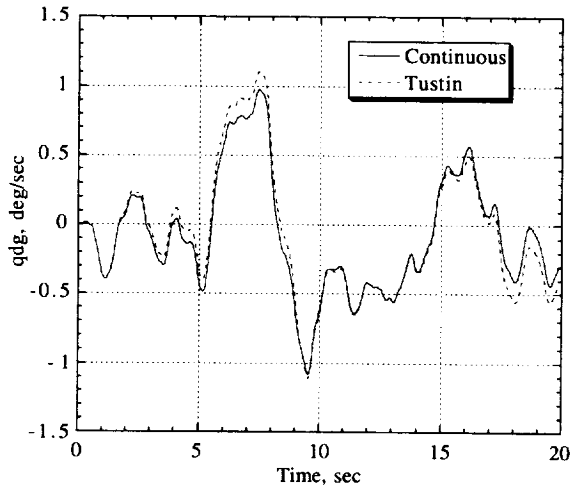


(f).- r-component.

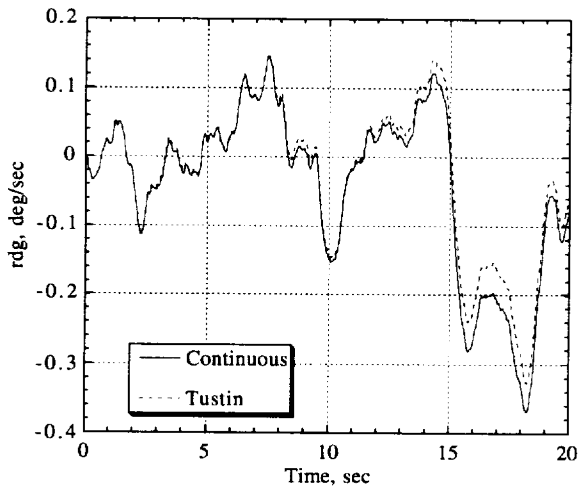
Figure 24. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 25°$, seed no. 1.



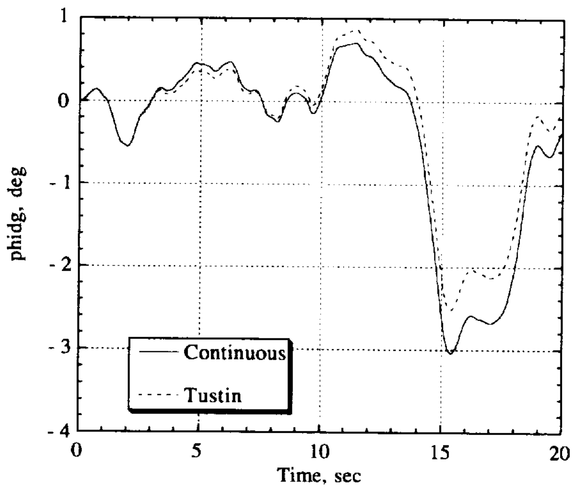
(a).- Roll rate.



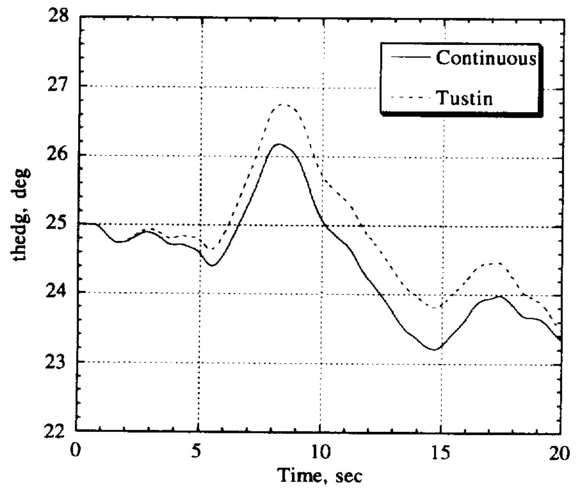
(b).- Pitch rate.



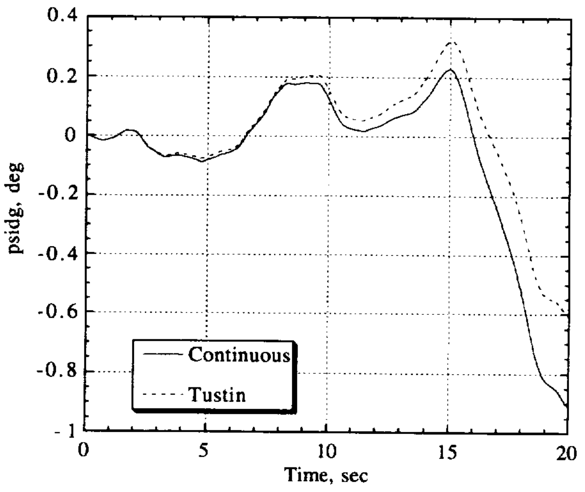
(c).- Yaw rate.



(d).- Bank angle.

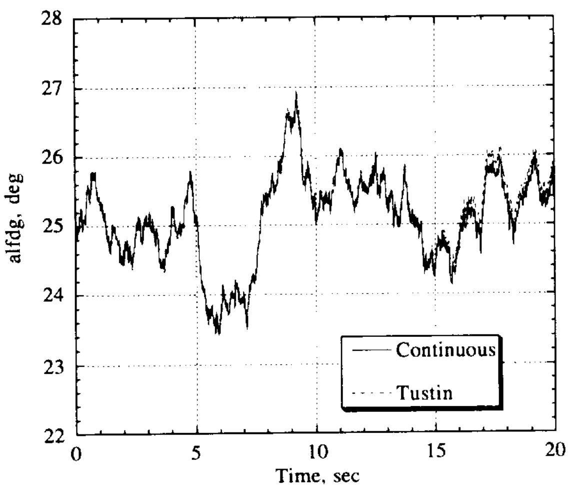


(e).-Pitch angle.

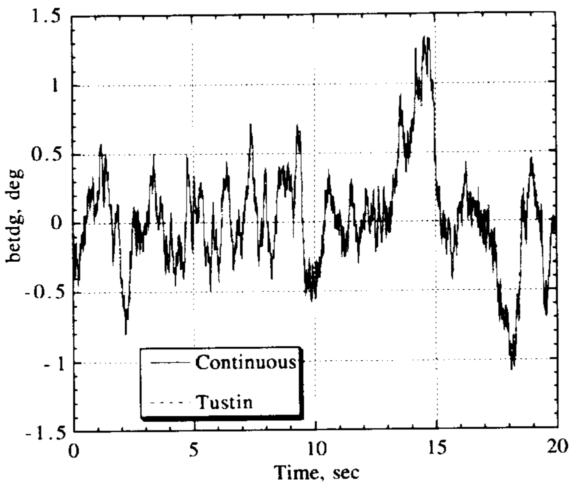


(f).-Heading.

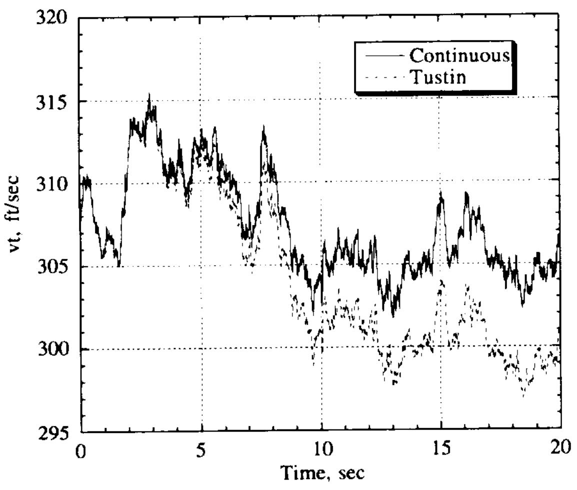
Figure 25. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 25°$, seed no. 1.



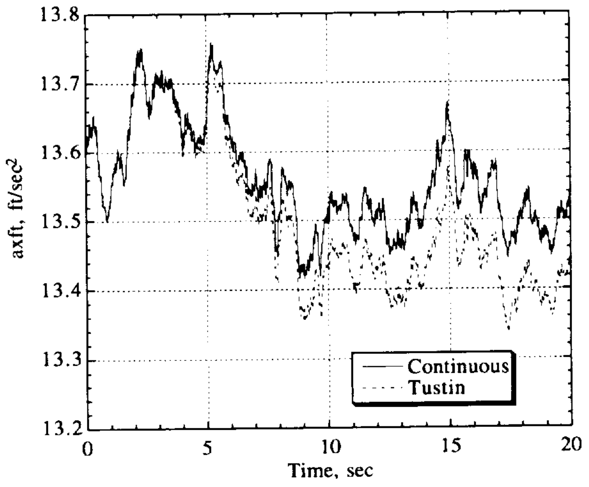
(a).- Angle of attack.



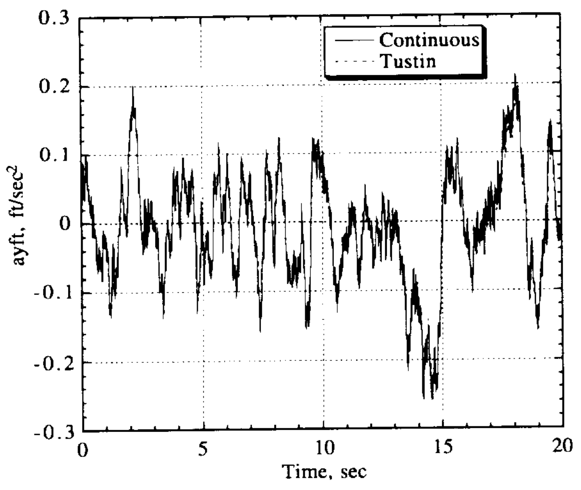
(b).- Sideslip angle.



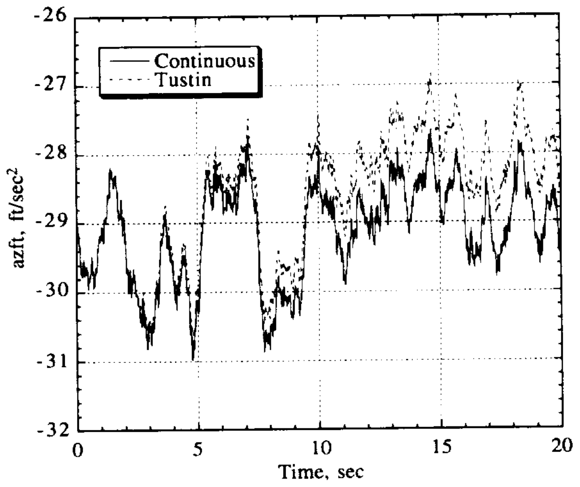
(c).- True airspeed.



(d).- X-axis acceleration.

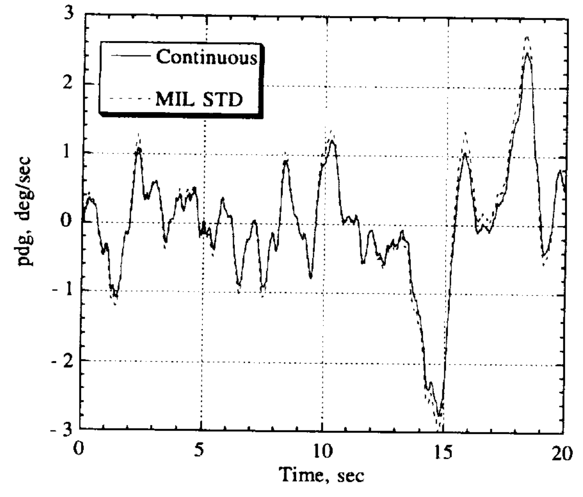


(e).- Y-axis acceleration.

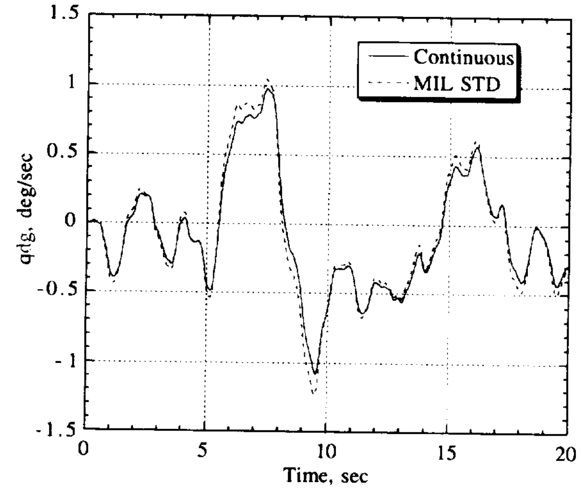


(f).- Z-axis acceleration.

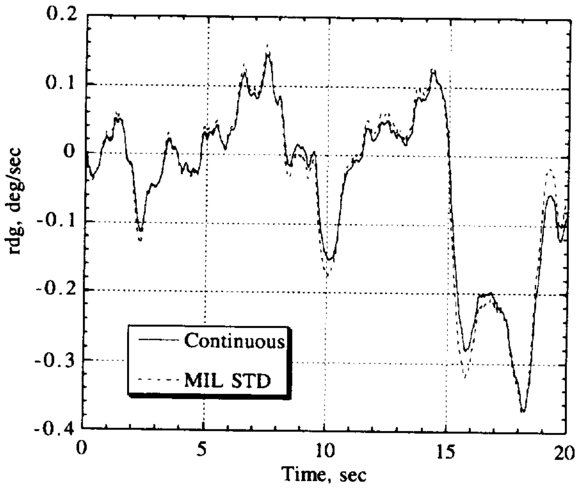
Figure 26. Comparison of continuous and Tustin air data and accelerations for $\alpha = 25°$, seed no. 1.



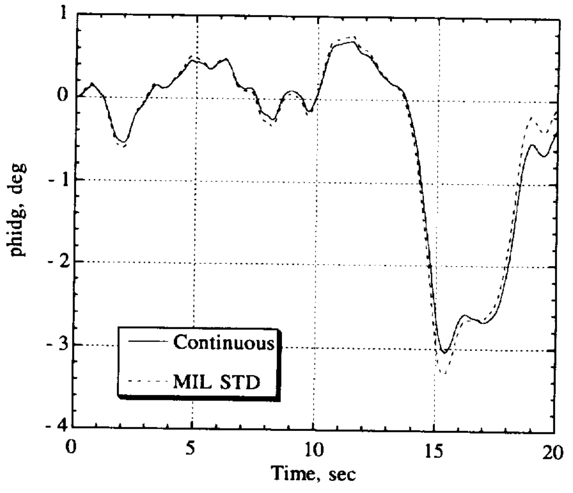
(a).- Roll rate.



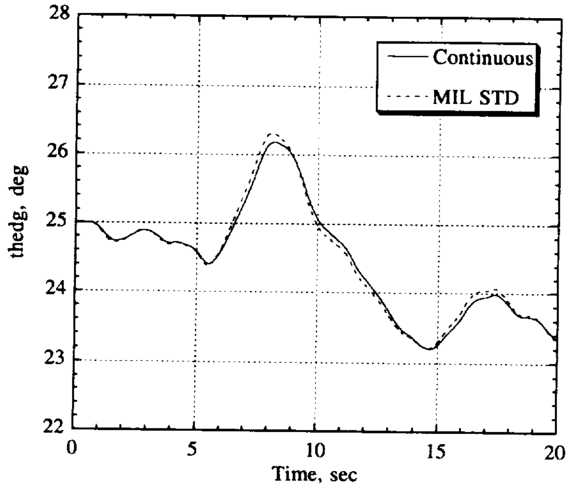
(b).- Pitch rate.



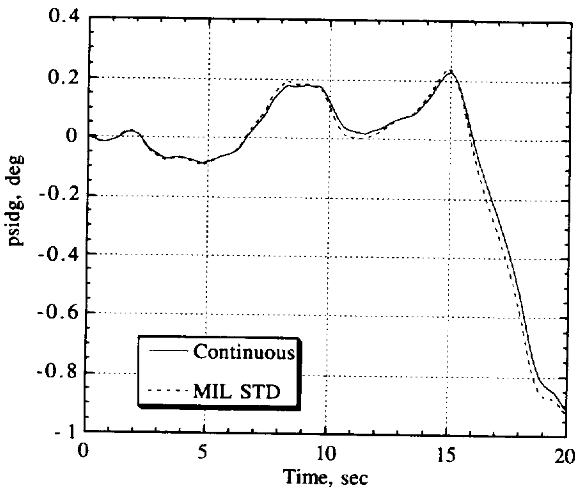
(c).- Yaw rate.



(d).- Bank angle.

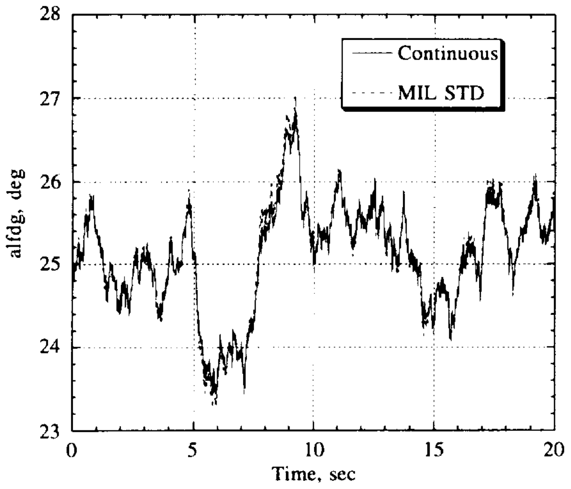


(e).- Pitch angle.

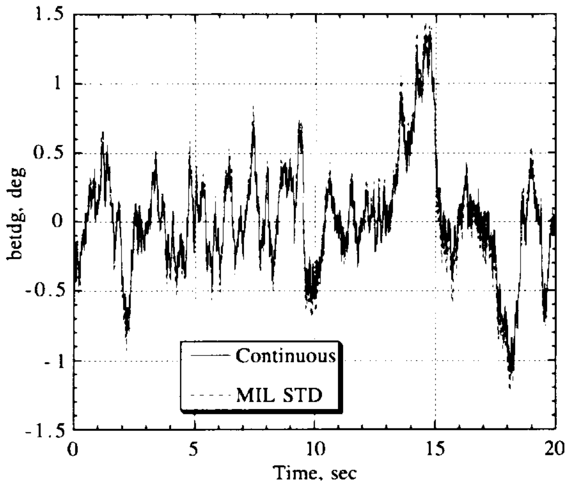


(f).- Heading.

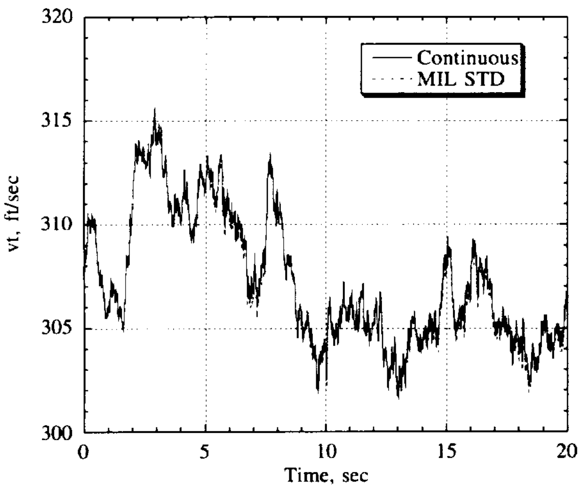
Figure 27. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 25°$, seed no. 1.



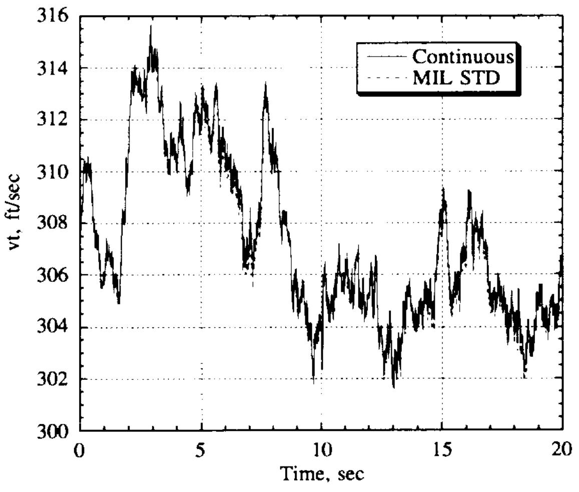
(a).- Angle of attack.



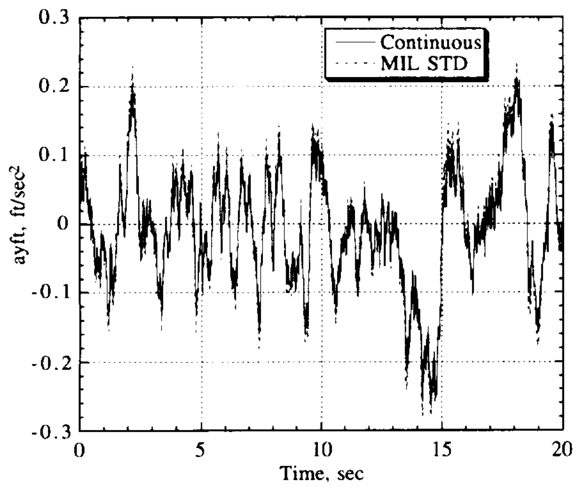
(b).- Sideslip angle.



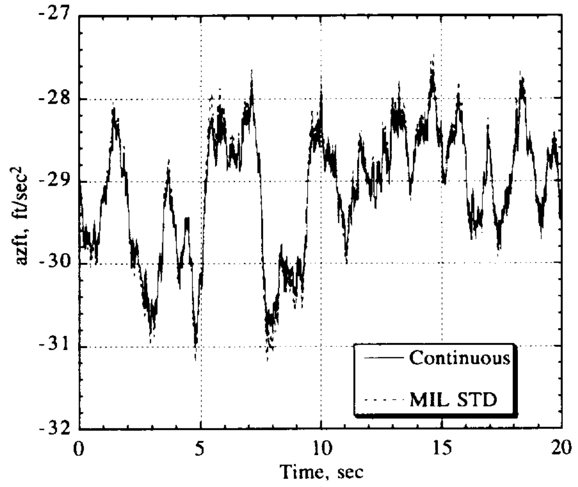
(c).- True airspeed.



(d).- X-axis acceleration.

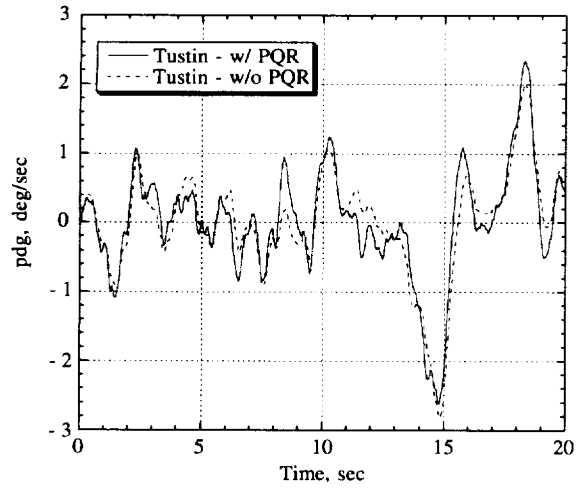


(e).- Y-axis acceleration.

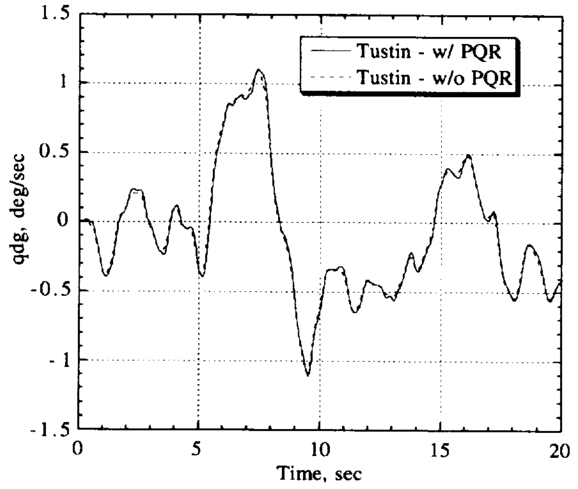


(f).- Z-axis acceleration.

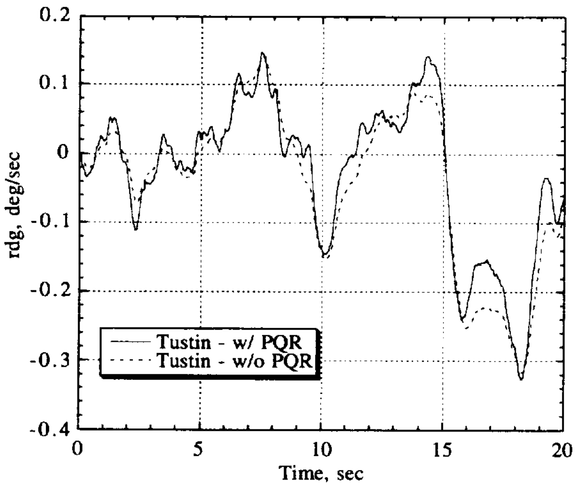
Figure 28. Comparison of continuous and Tustin air data and accelerations for $\alpha = 25°$, seed no. 1.



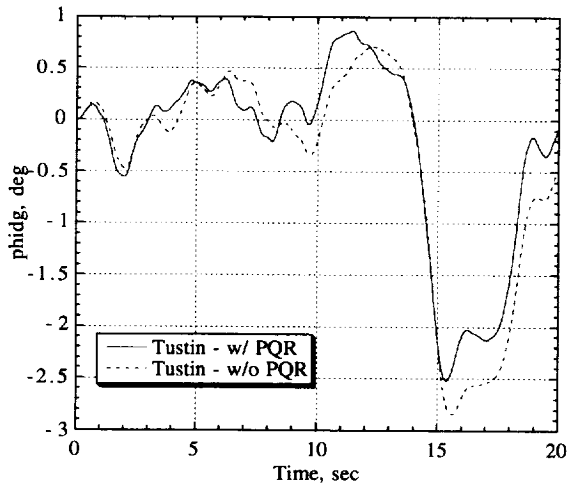
(a).- Roll rate.



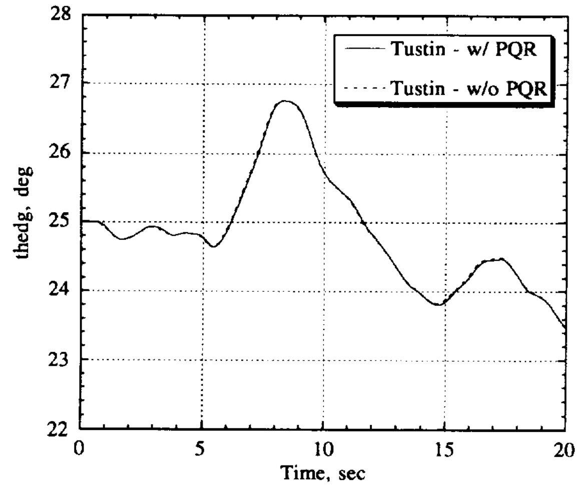
(b).- Pitch rate.



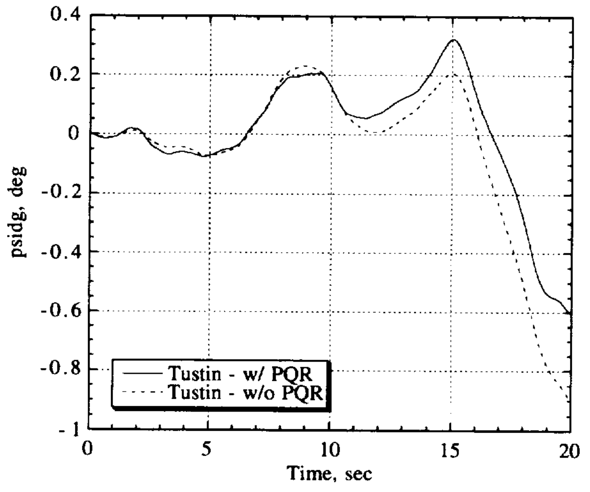
(c).- Yaw rate.



(d).- Bank angle.

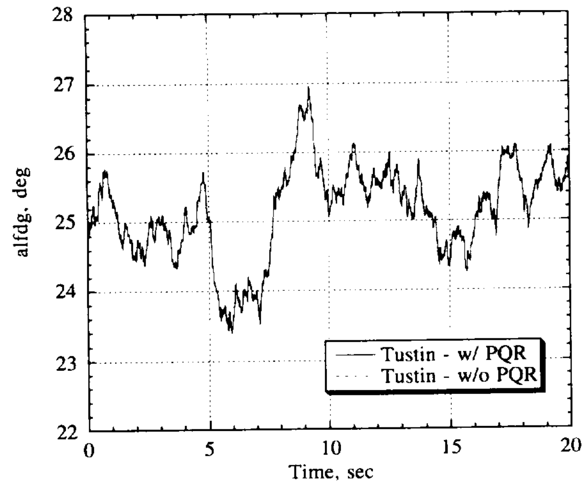


(e).- Pitch angle.

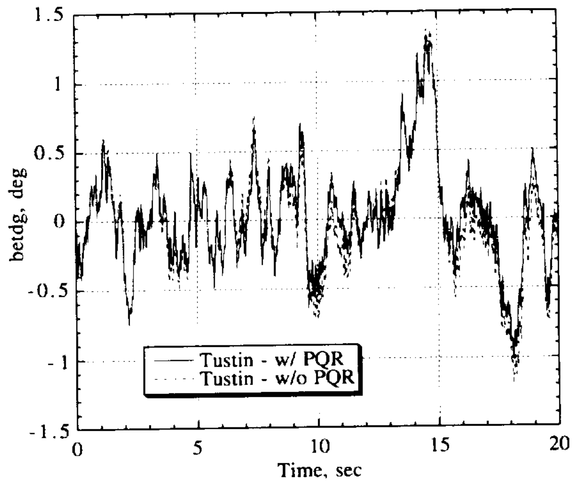


(f).- Heading.

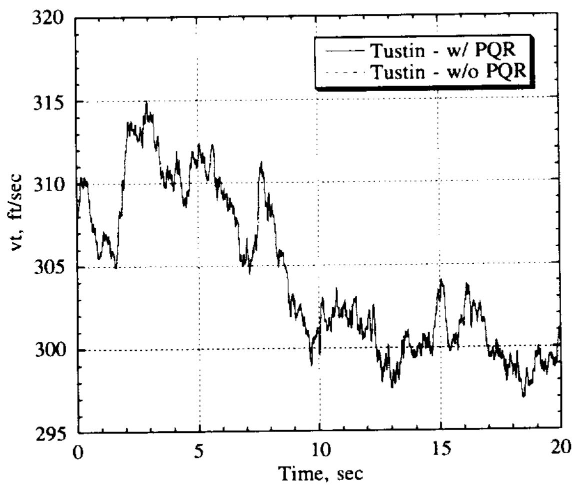
Figure 29. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 25°$, seed no. 1.



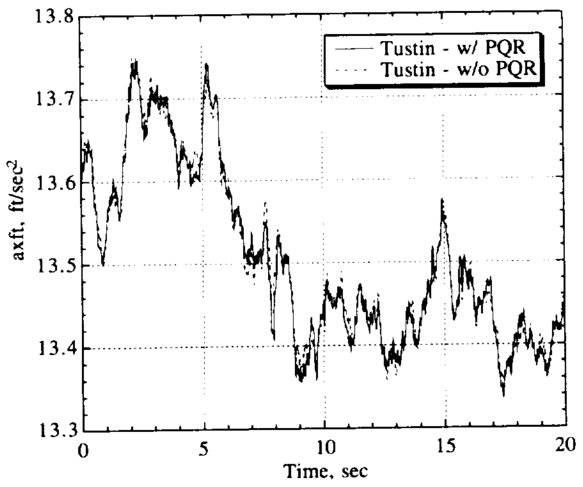
(a).- Angle of attack.



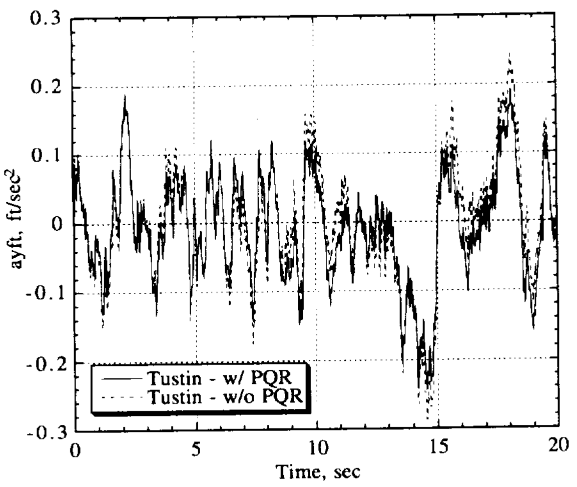
(b).- Sideslip angle.



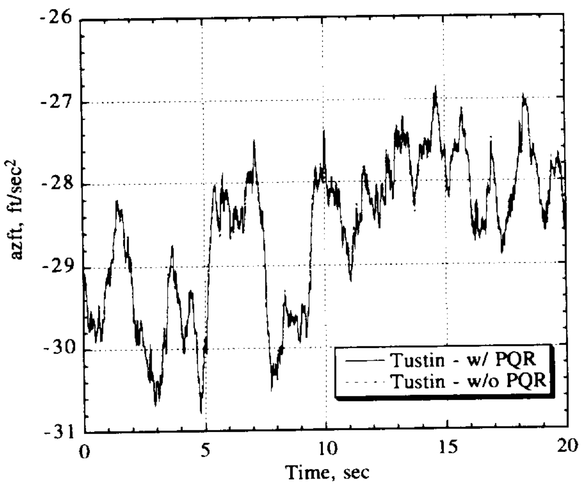
(c).- True airspeed.



(d).- X-axis acceleration.

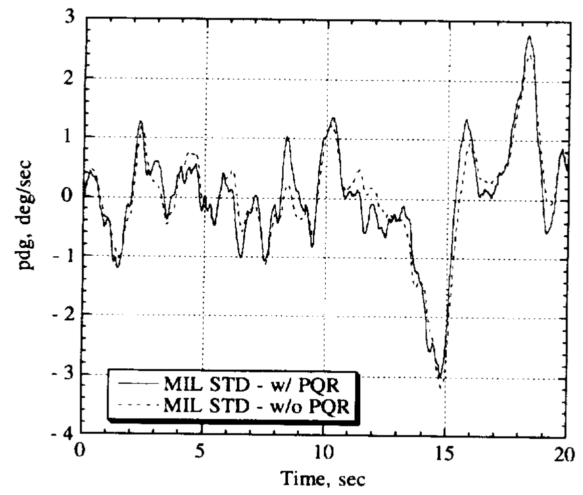


(e).- Y-axis acceleration.

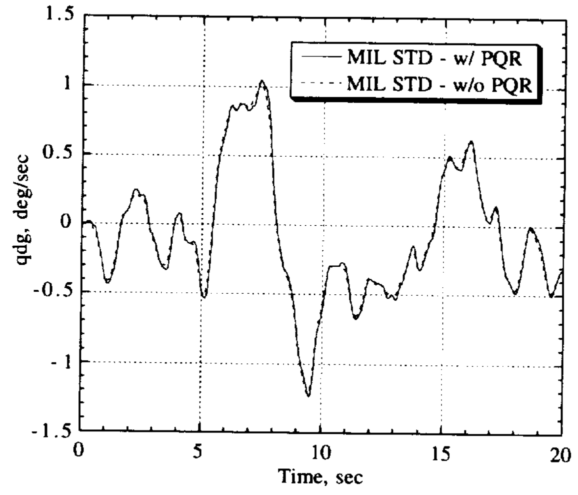


(f).- Z-axis acceleration.

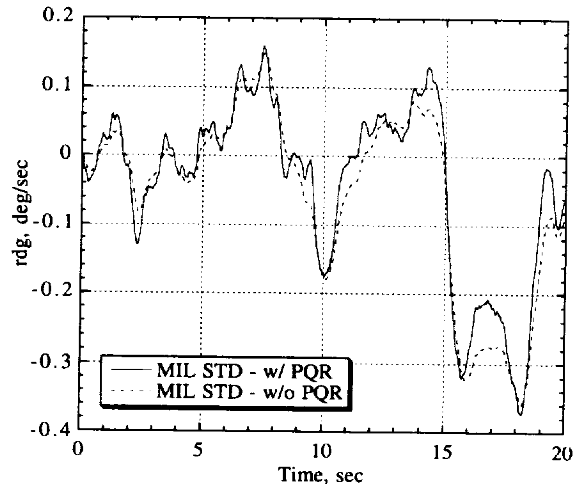
Figure 30. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 25°$, seed no. 1.



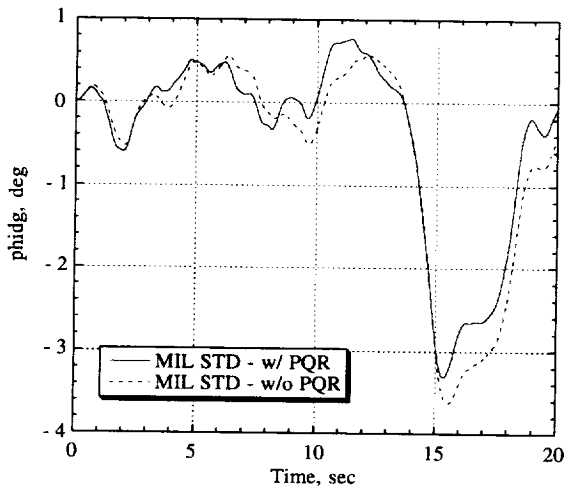
(a).- Roll rate.



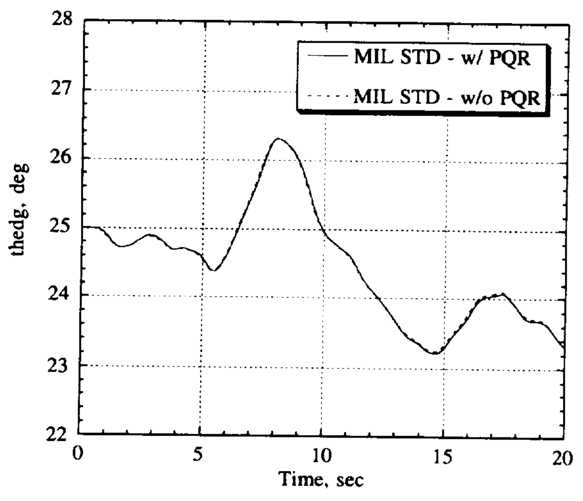
(b).- Pitch rate.



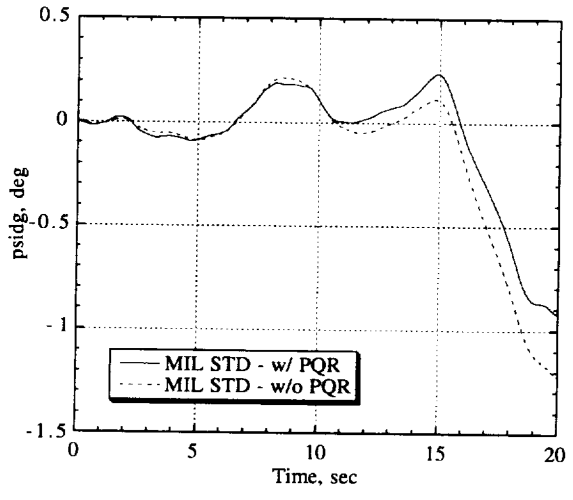
(c).- Yaw rate.



(d).- Bank angle.

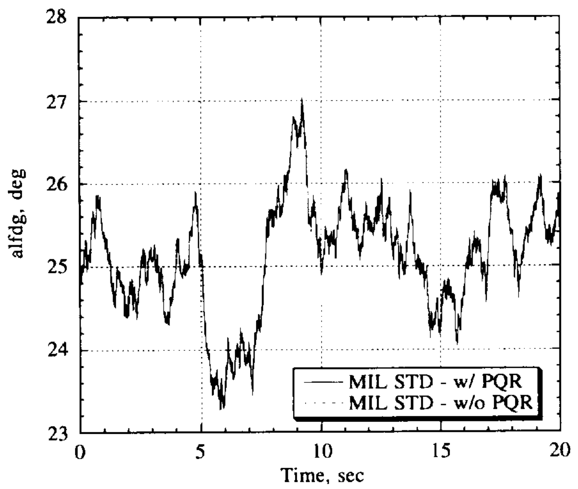


(e).- Pitch angle.

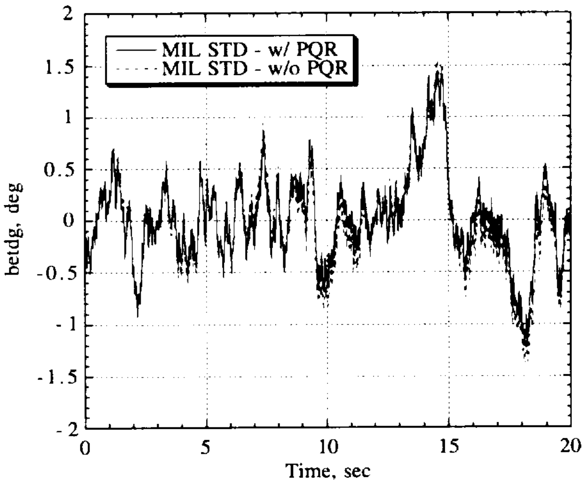


(f).- Heading.

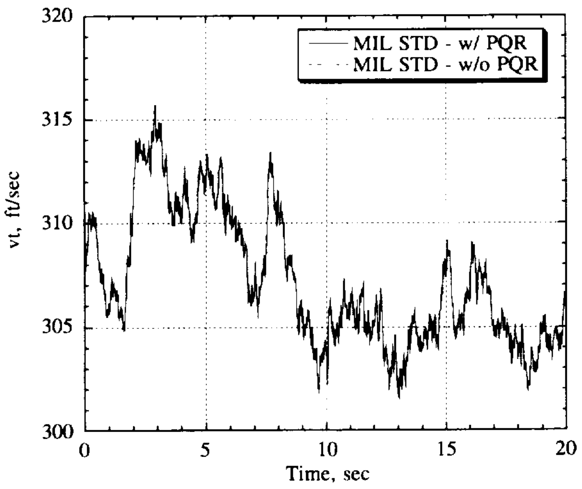
Figure 31. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 25°$, seed no. 1.



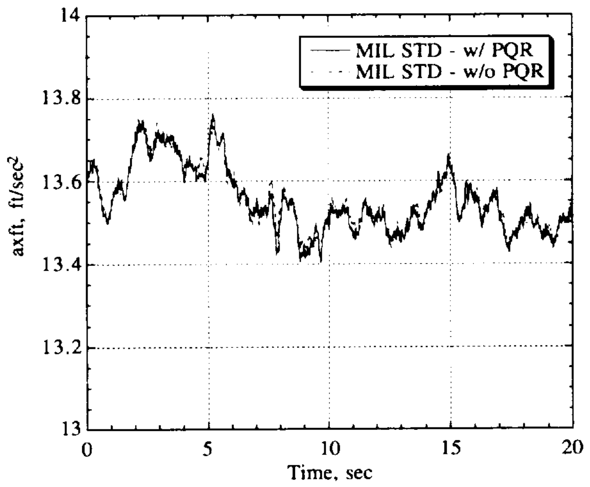
(a).- Angle of attack.



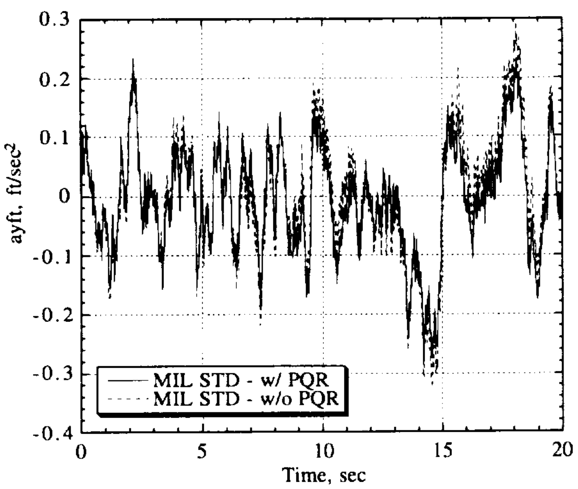
(b).- Sideslip angle.



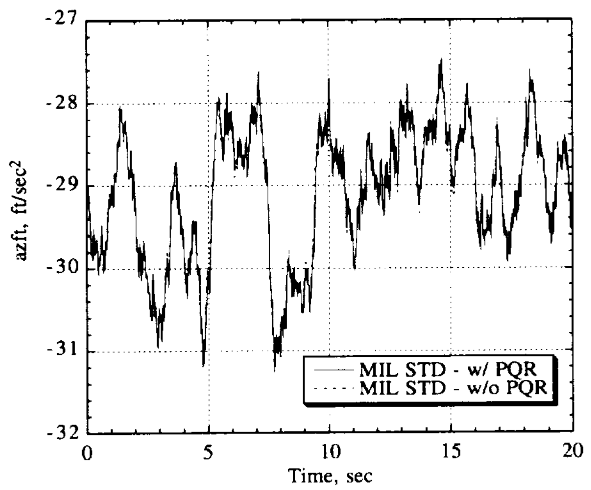
(c).- True airspeed.



(d).- X-axis acceleration.

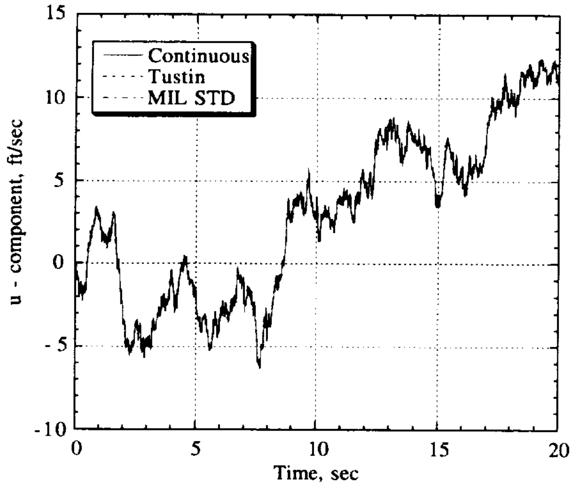


(e).- Y-axis acceleration.

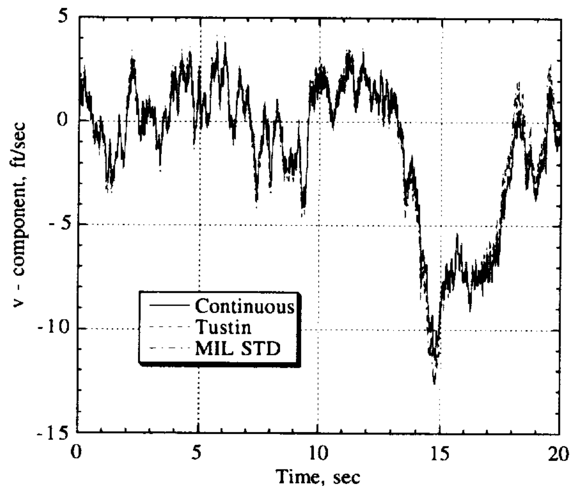


(f).- Z-axis acceleration.

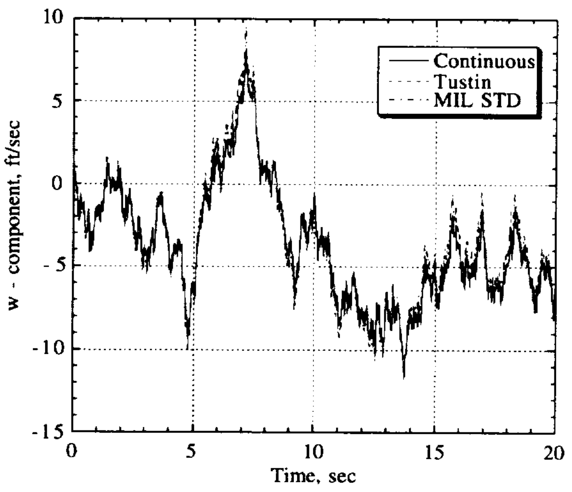
Figure 32. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for α = 25°, seed no. 1.



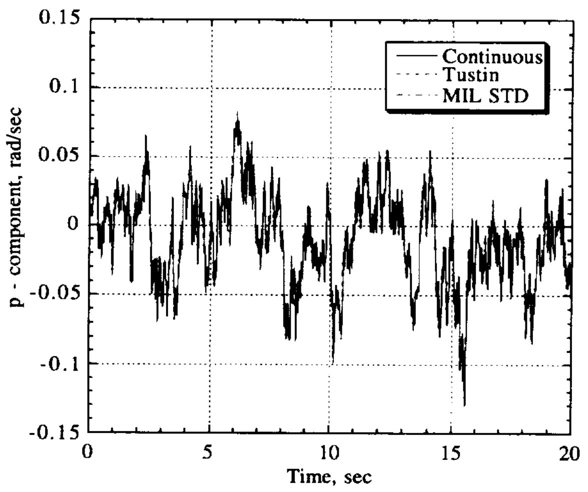
(a).- u-component.



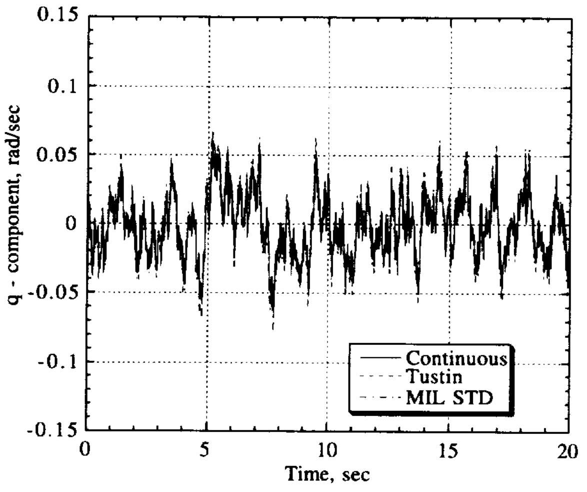
(b).- v-component.



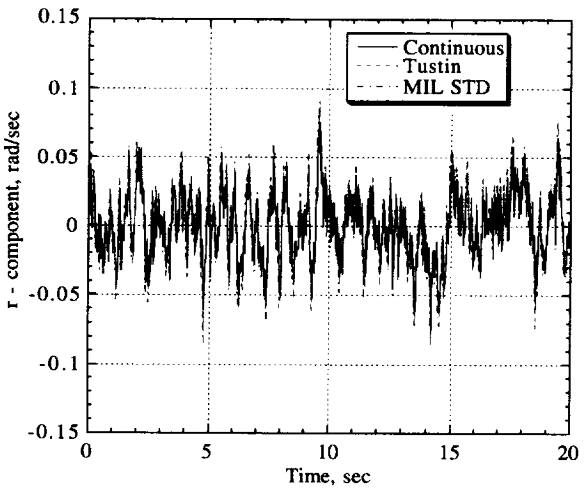
(c).- w-component.



(d).- p-component.

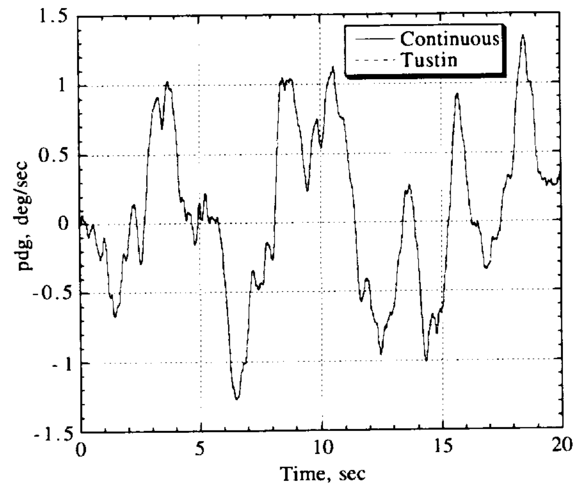


(e).- q-component.

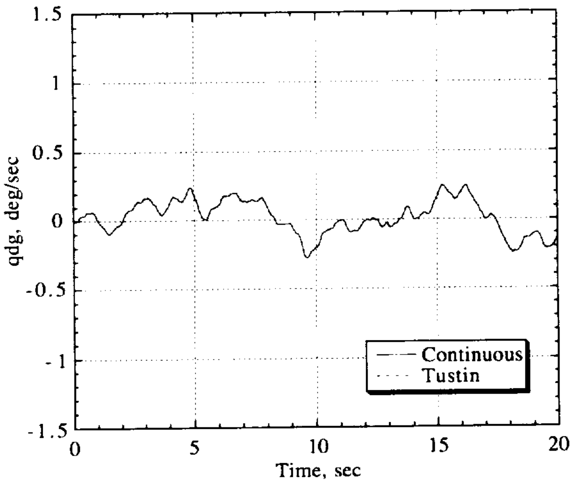


(f).- r-component.

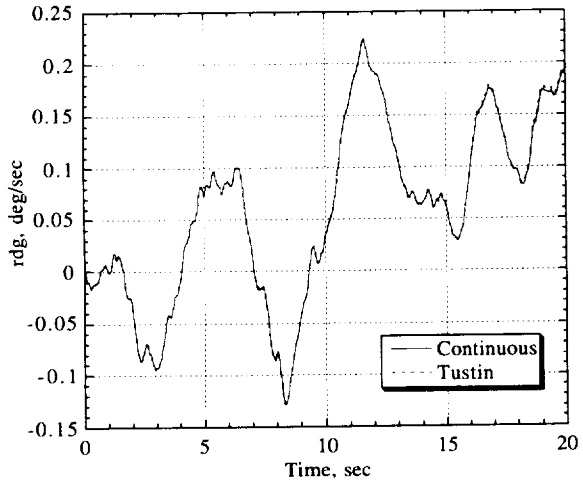
Figure 33. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 35°$, seed no. 1.



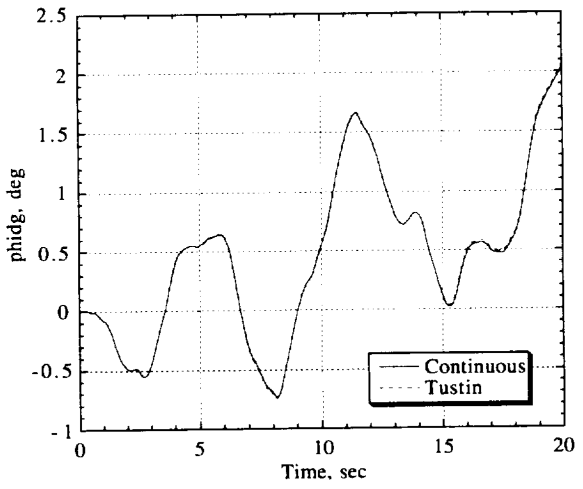
(a).- Roll rate.



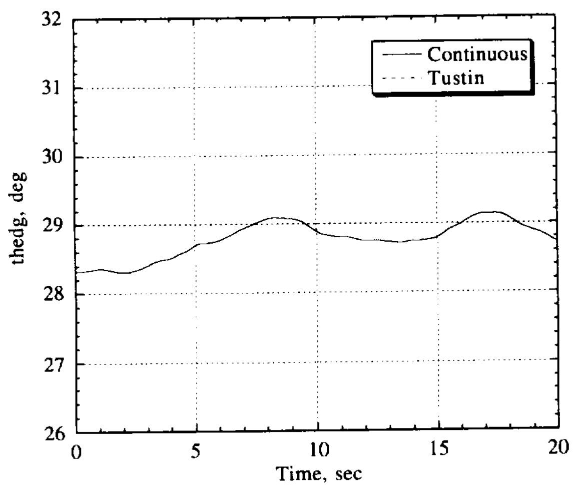
(b).- Pitch rate.



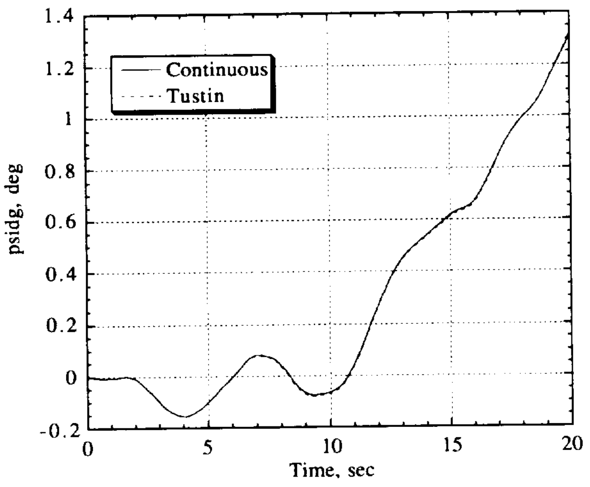
(c).- Yaw rate.



(d).- Bank angle.

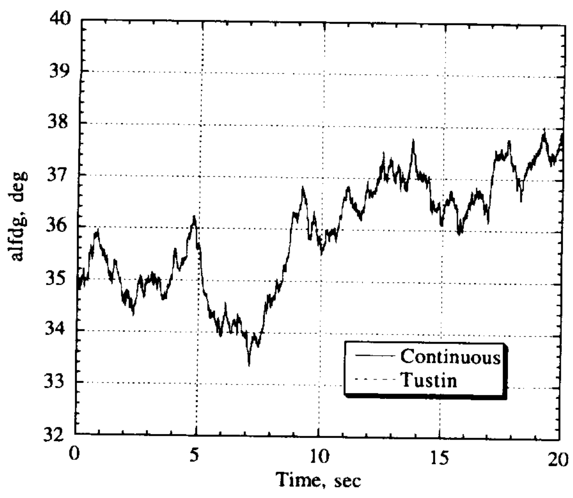


(e).-Pitch angle.

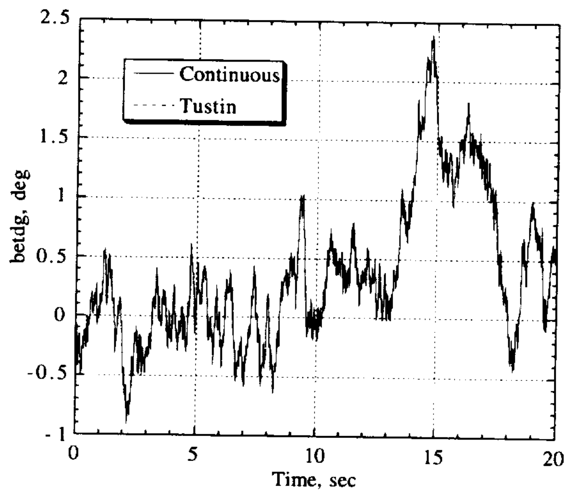


(f).-Heading.

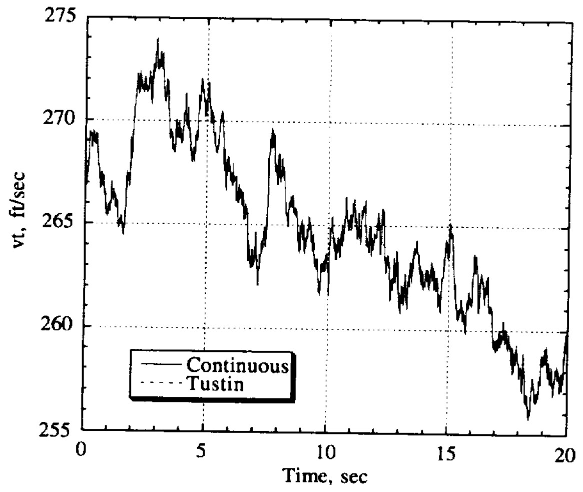
Figure 34. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 35°$, seed no. 1.



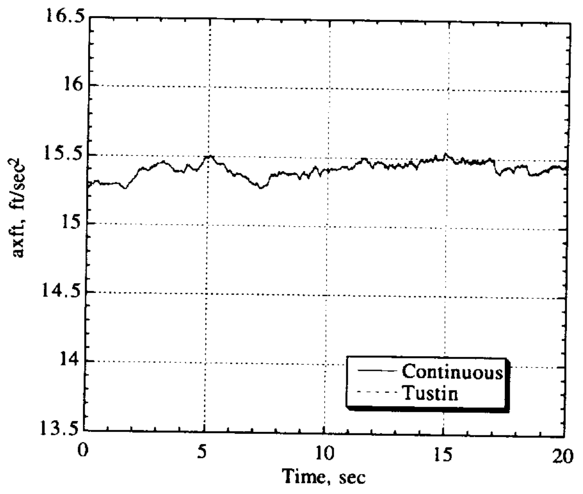
(a).- Angle of attack.



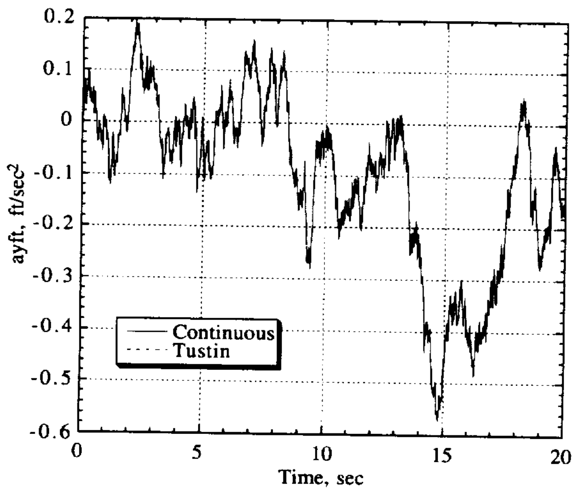
(b).- Sideslip angle.



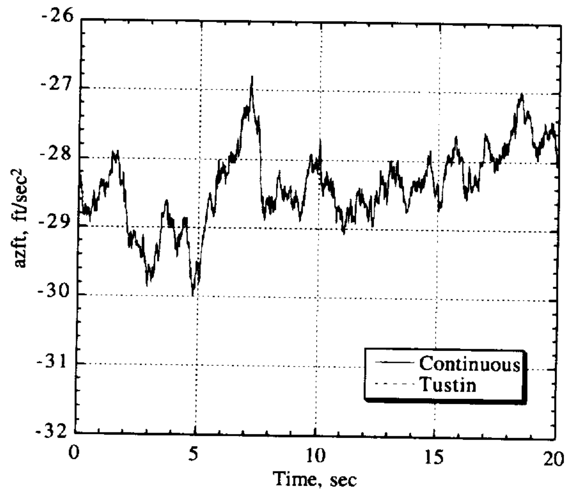
(c).- True airspeed.



(d).- X-axis acceleration.

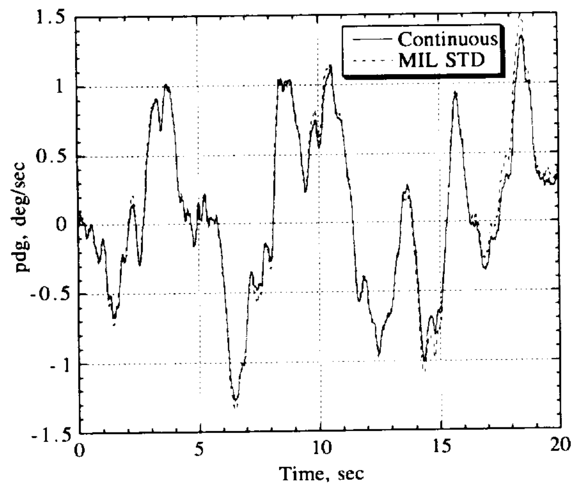


(e).- Y-axis acceleration.

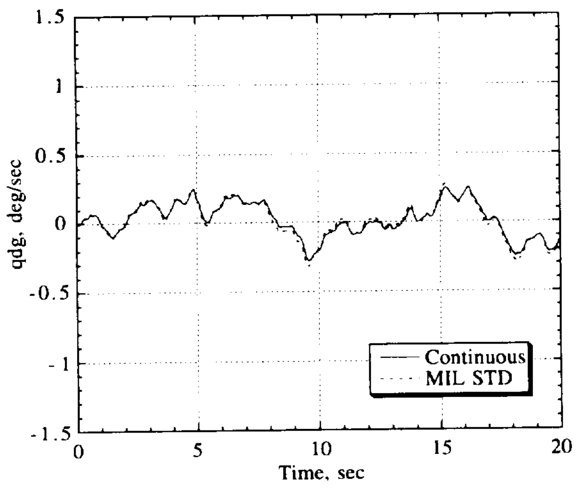


(f).- Z-axis acceleration.

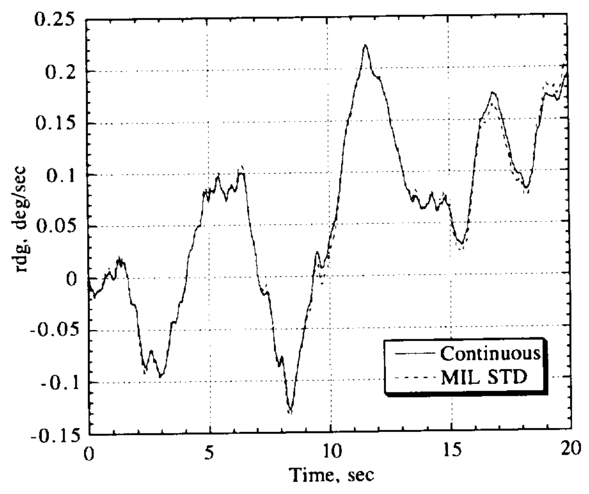
Figure 35. Comparison of continuous and Tustin air data and accelerations for $\alpha = 35°$, seed no. 1.



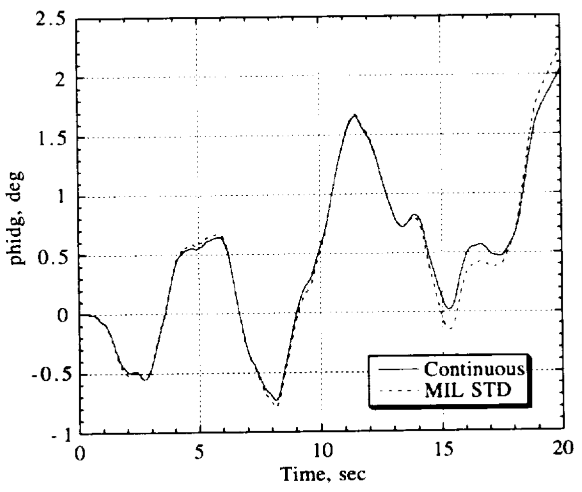
(a).- Roll rate.



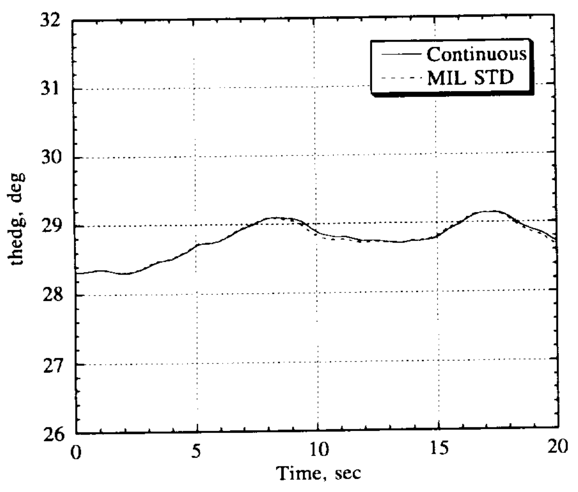
(b).- Pitch rate.



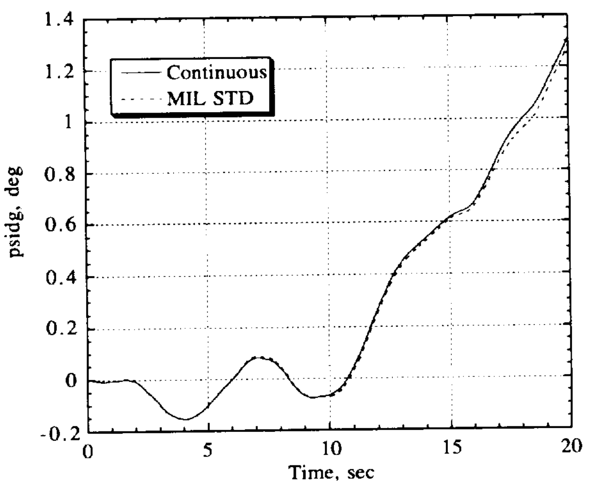
(c).- Yaw rate.



(d).- Bank angle.

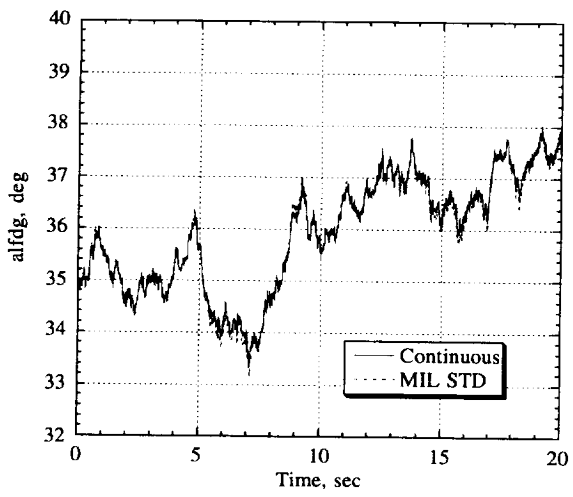


(e).- Pitch angle.

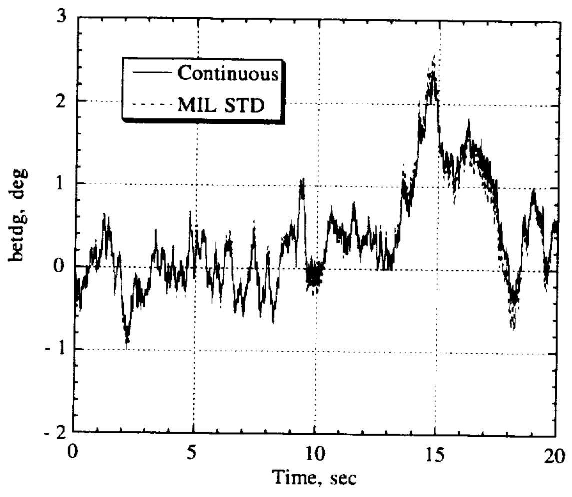


(f).- Heading.

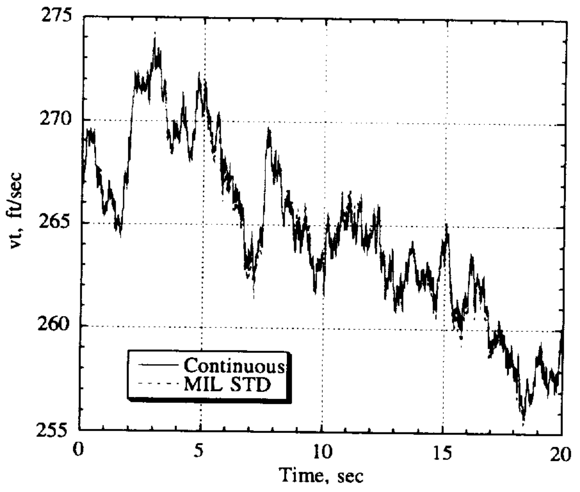
Figure 36. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 35°$, seed no. 1.



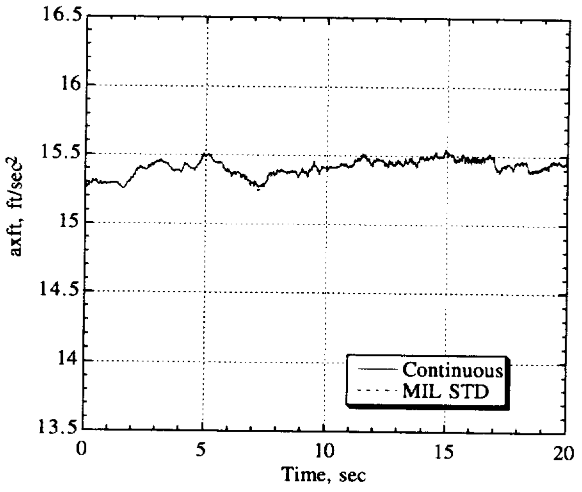
(a).- Angle of attack.



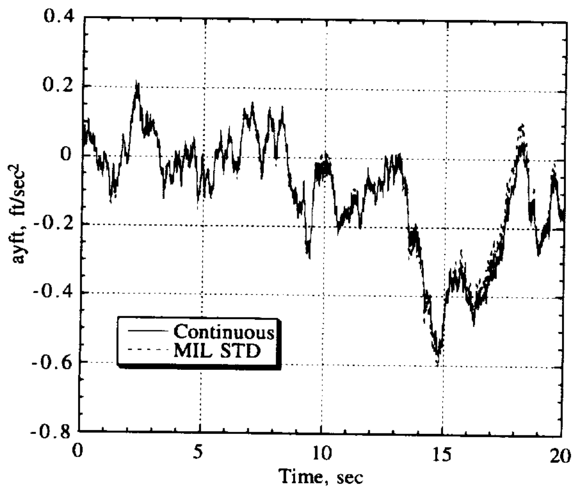
(b).- Sideslip angle.



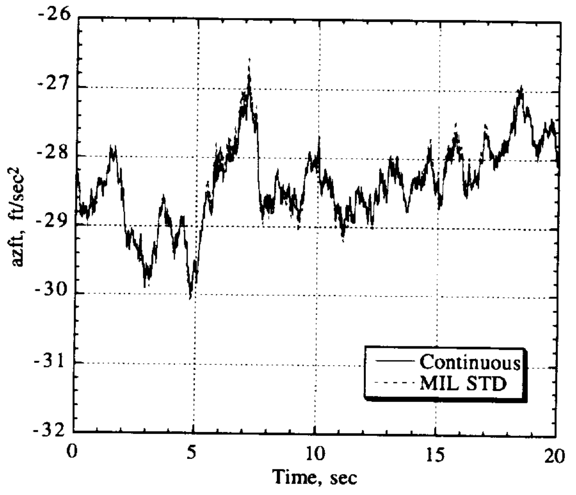
(c).- True airspeed.



(d).- X-axis acceleration.

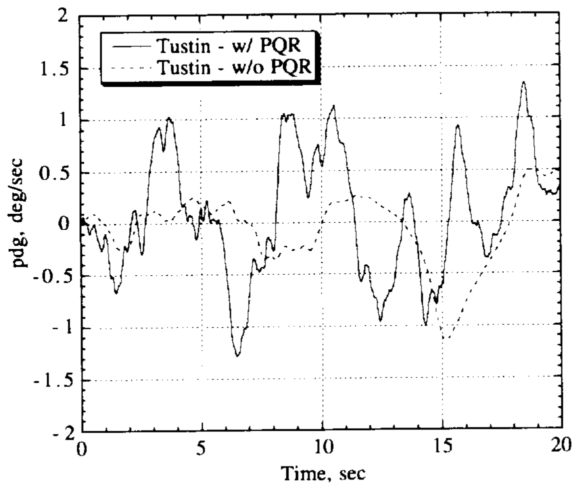


(e).- Y-axis acceleration.

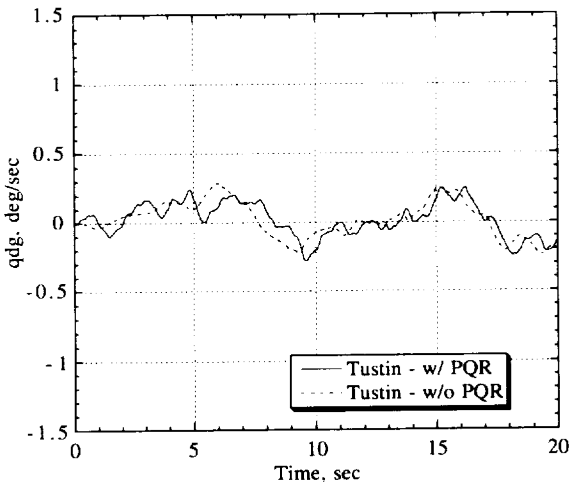


(f).- Z-axis acceleration.

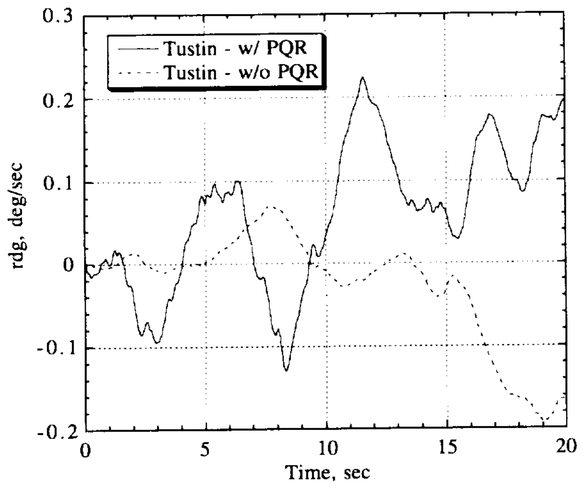
Figure 37. Comparison of continuous and Tustin air data and accelerations for $\alpha = 35°$, seed no. 1.



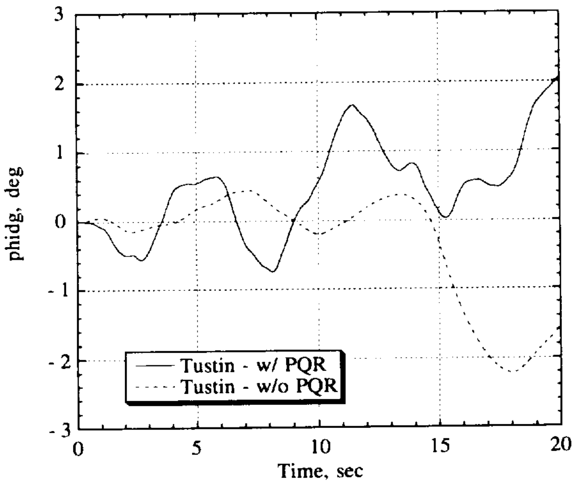
(a).- Roll rate.



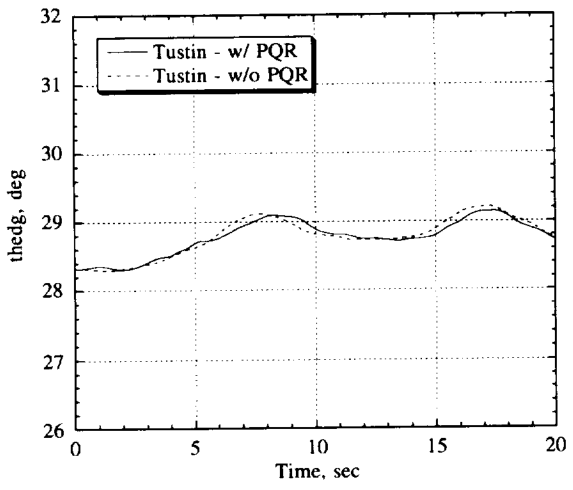
(b).- Pitch rate.



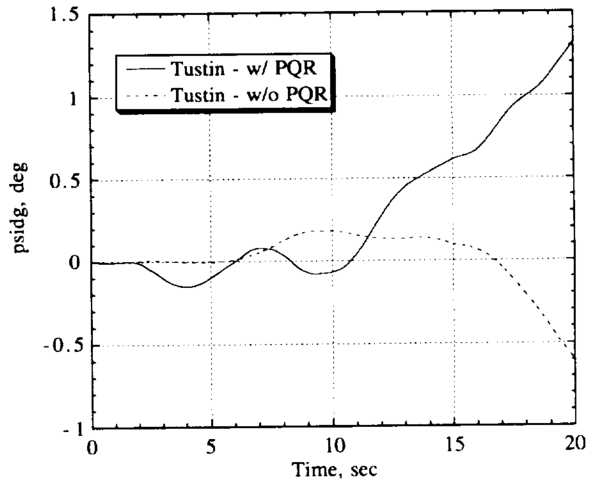
(c).- Yaw rate.



(d).- Bank angle.

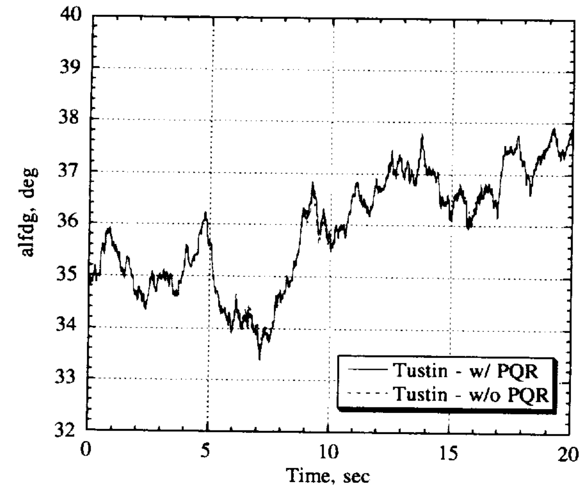


(e).- Pitch angle.

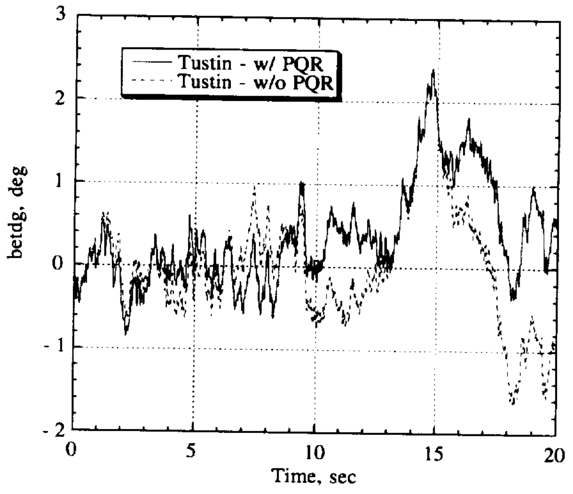


(f).- Heading.

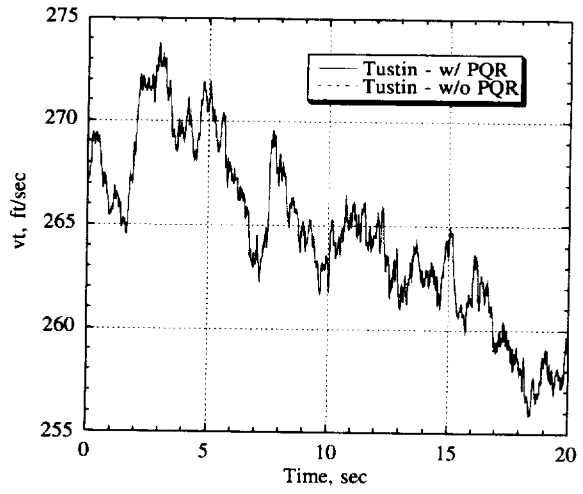
Figure 38. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 35°$, seed no. 1.



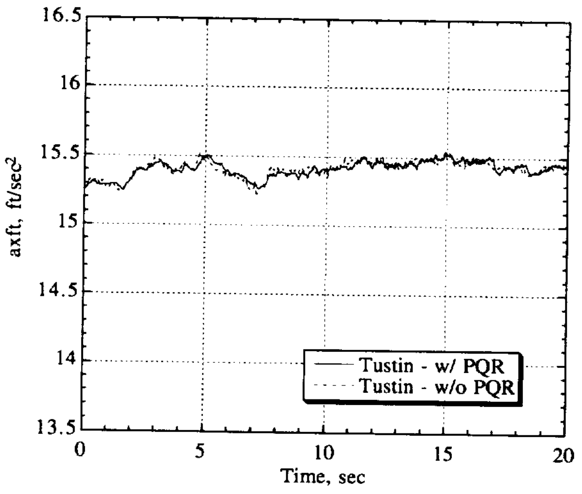
(a).- Angle of attack.



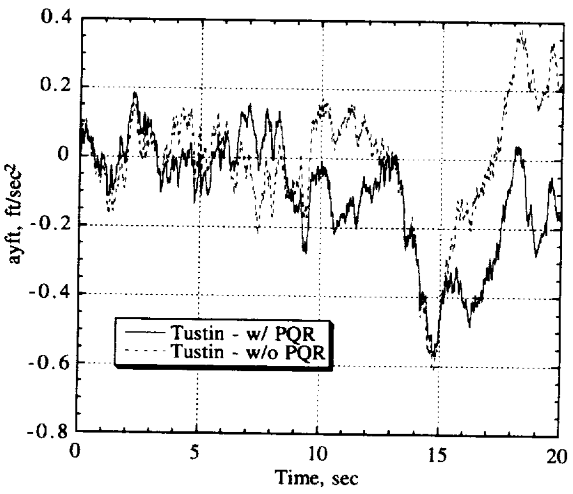
(b).- Sideslip angle.



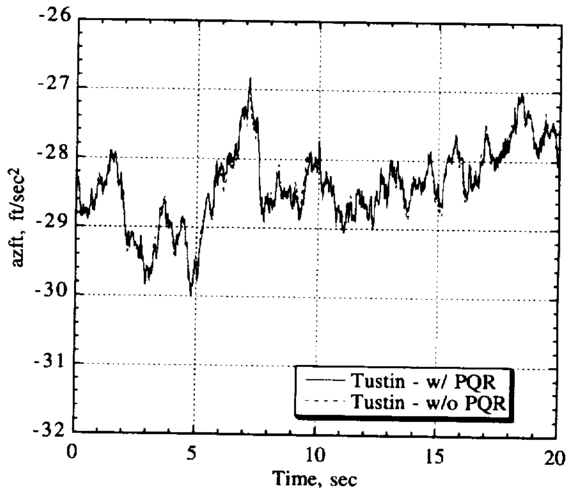
(c).- True airspeed.



(d).- X-axis acceleration.

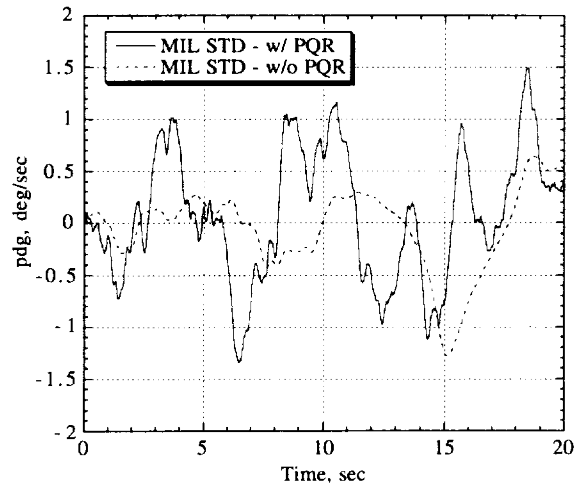


(e).- Y-axis acceleration.

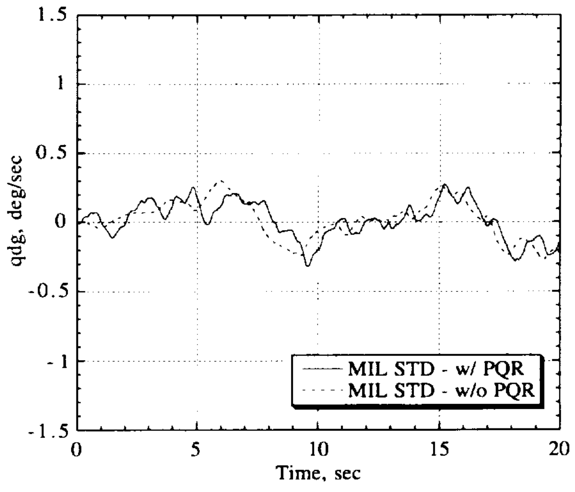


(f).- Z-axis acceleration.

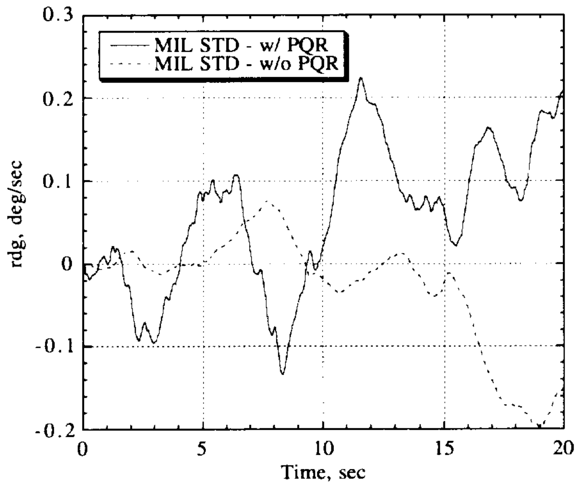
Figure 39. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 35°$, seed no. 1.



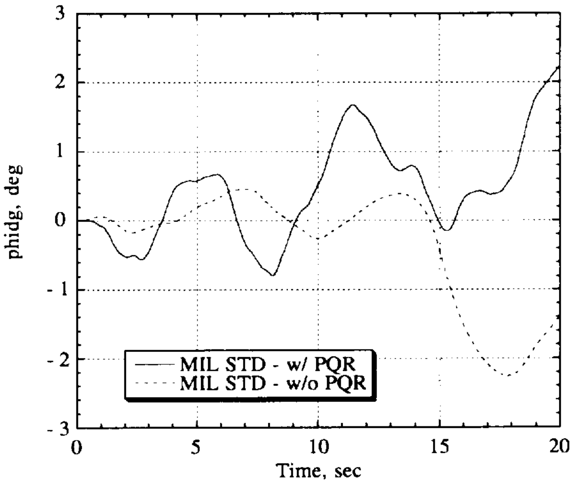
(a).- Roll rate.



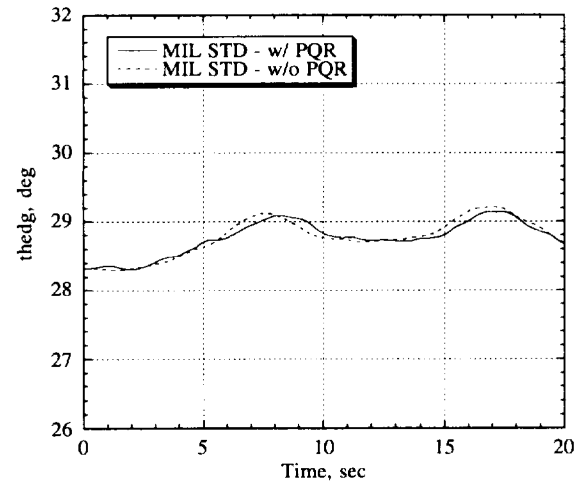
(b).- Pitch rate.



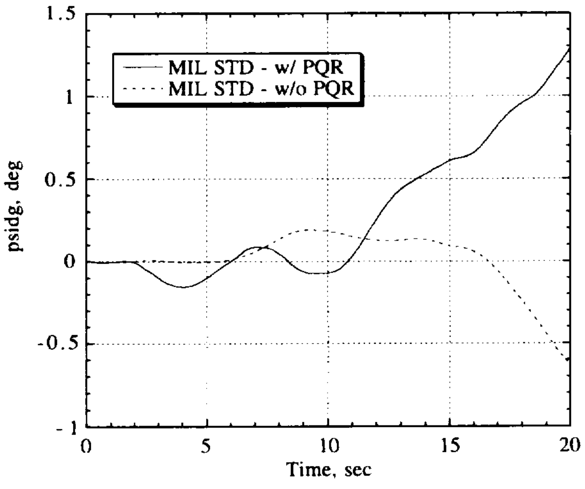
(c).- Yaw rate.



(d).- Bank angle.

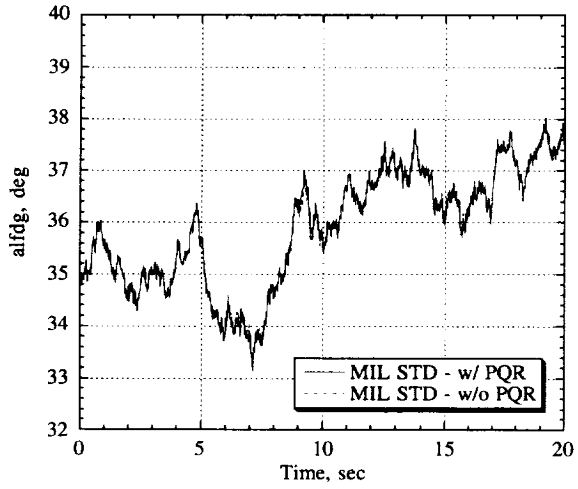


(e).- Pitch angle.

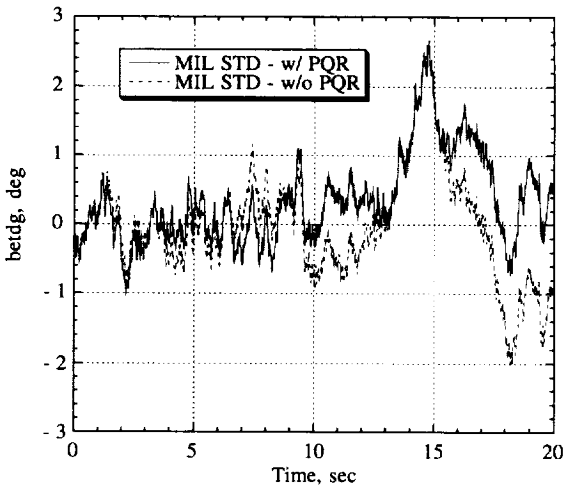


(f).- Heading.

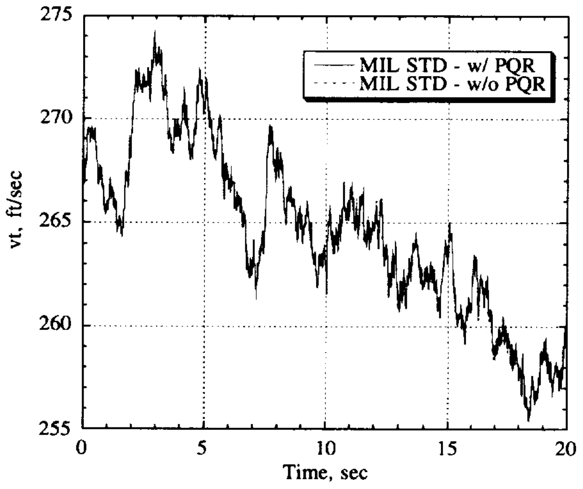
Figure 40. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 35°$, seed no. 1.



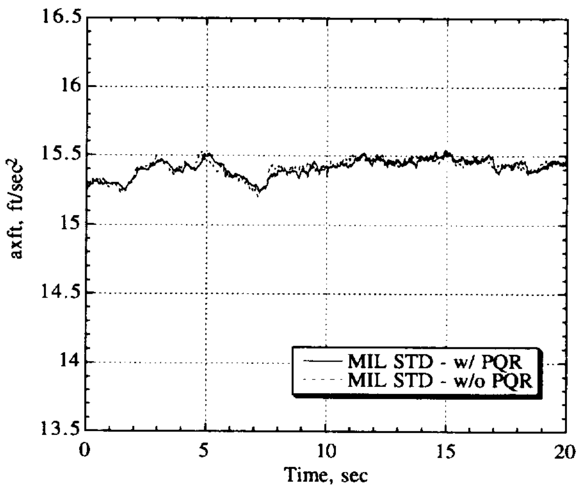
(a).- Angle of attack.



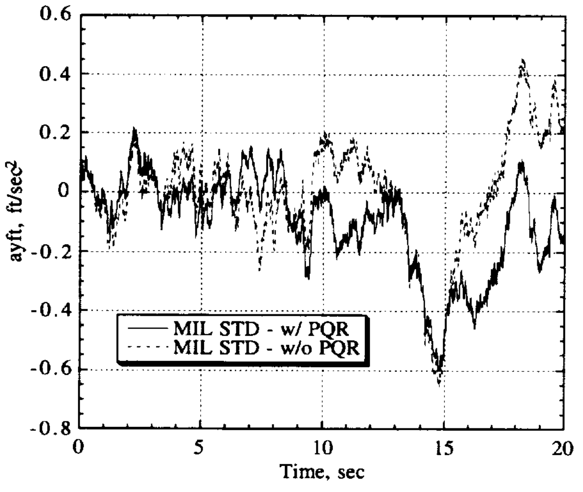
(b).- Sideslip angle.



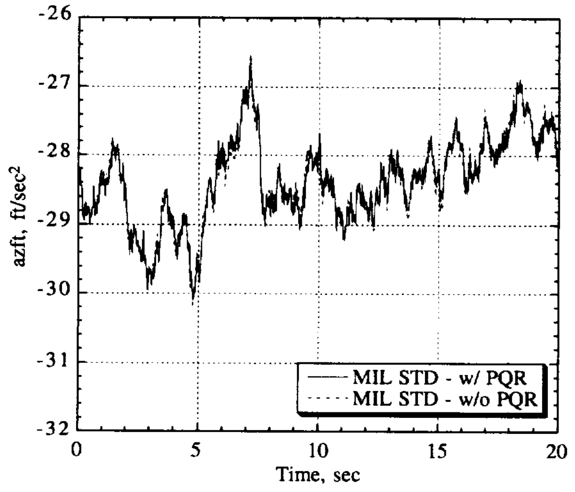
(c).- True airspeed.



(d).- X-axis acceleration.

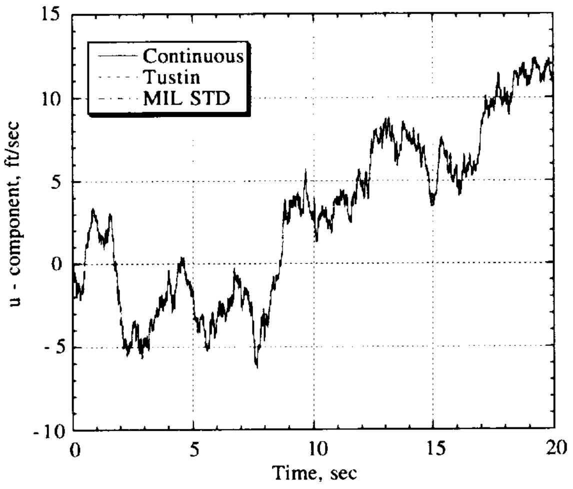


(e).- Y-axis acceleration.

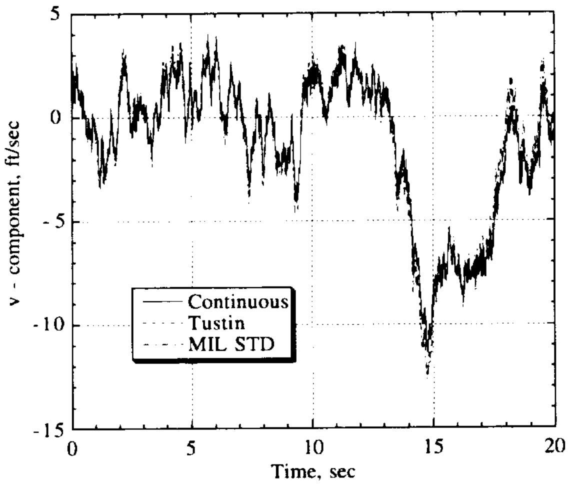


(f).- Z-axis acceleration.

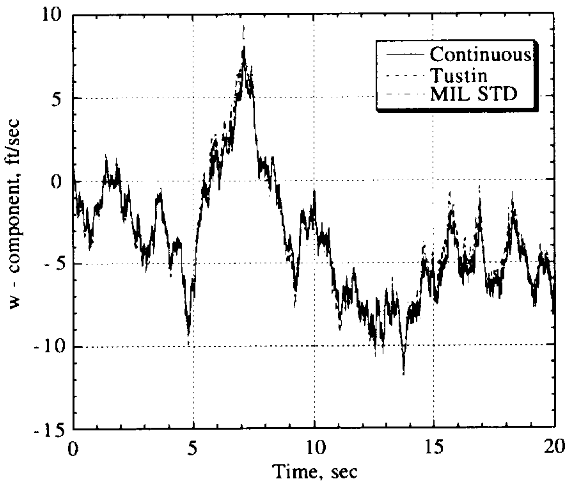
Figure 41. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 35°$, seed no. 1.



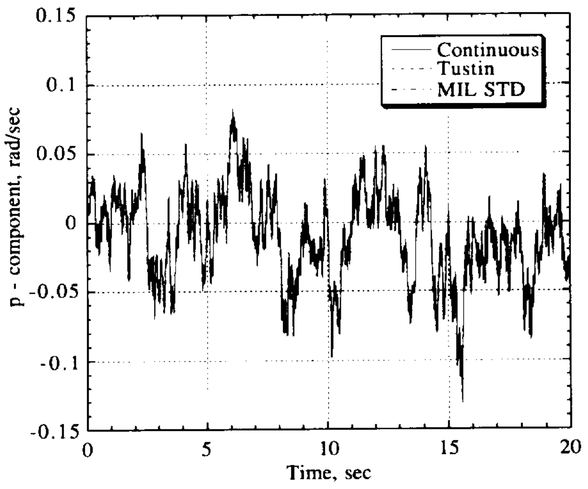
(a).- u-component.



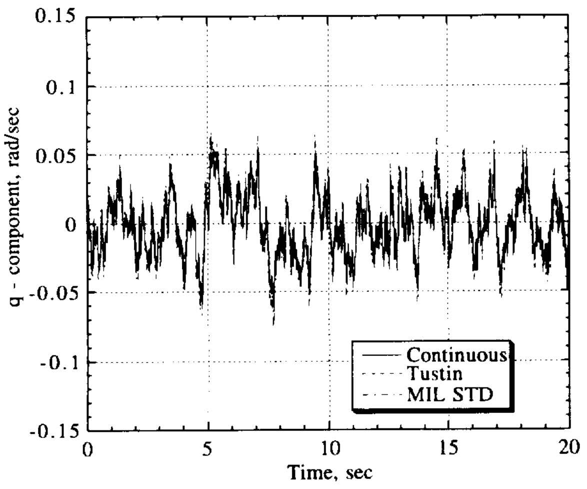
(b).- v-component.



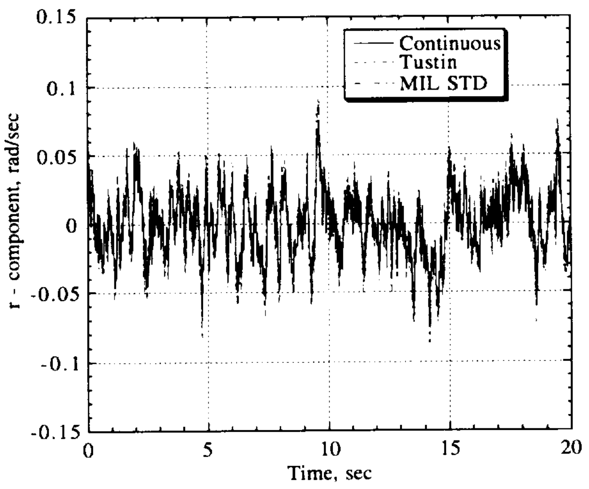
(c).- w-component.



(d).- p-component.

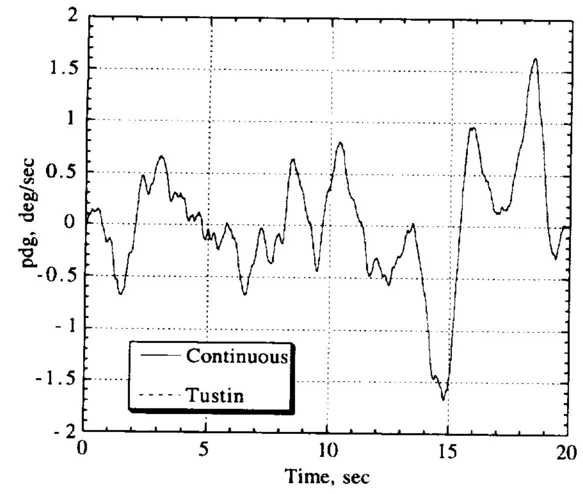


(e).- q-component.

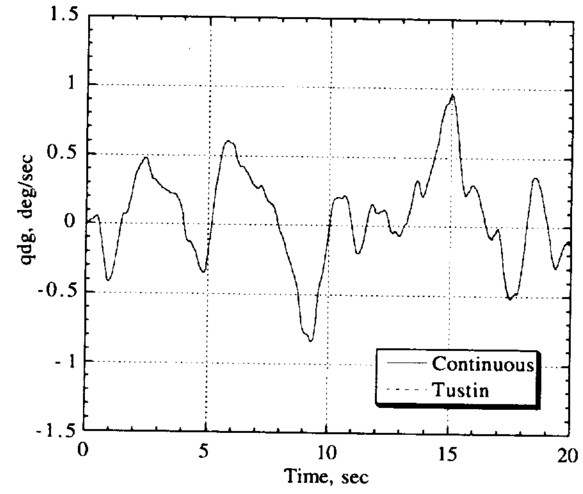


(f).- r-component.

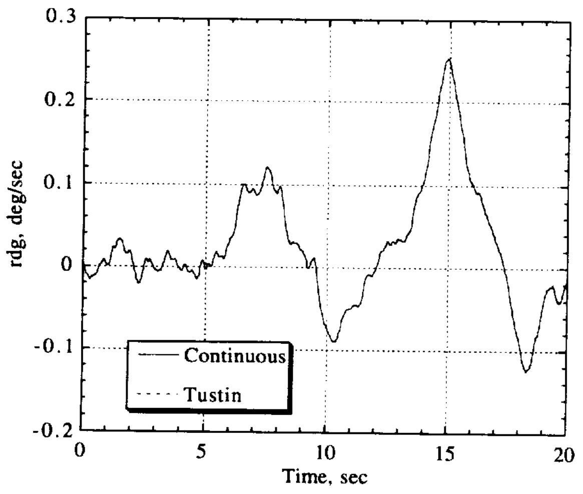
Figure 42. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 55°$, seed no. 1.



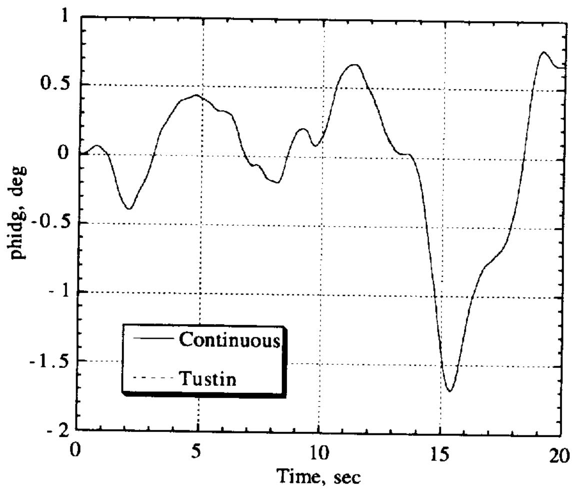
(a).- Roll rate.



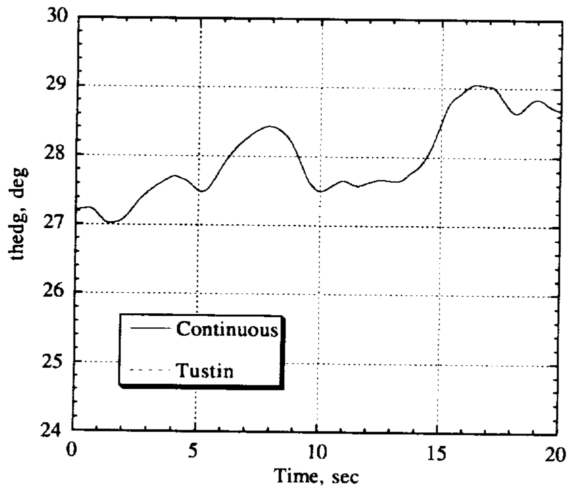
(b).- Pitch rate.



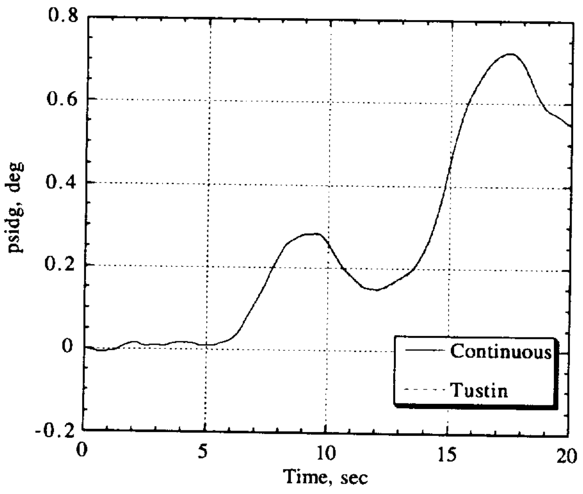
(c).- Yaw rate.



(d).- Bank angle.

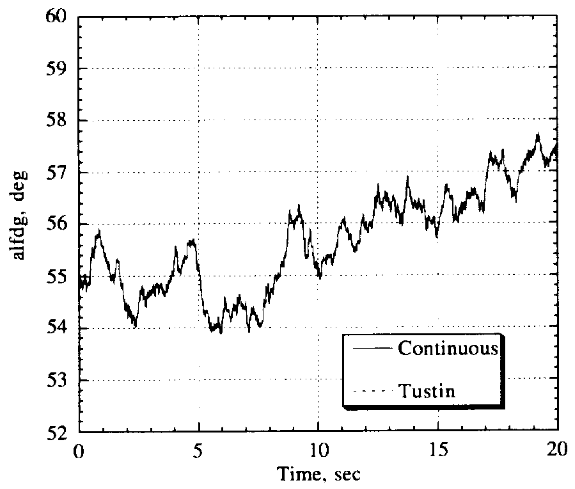


(e).-Pitch angle.

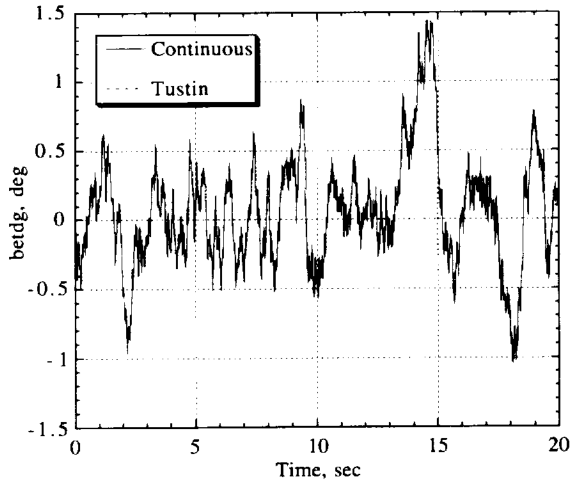


(f).-Heading.

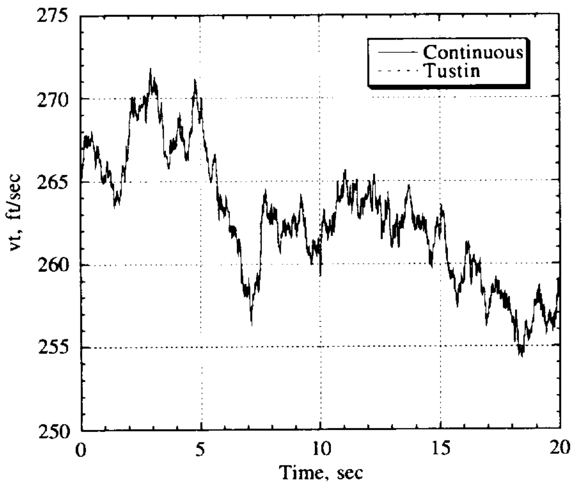
Figure 43. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 55°$, seed no. 1



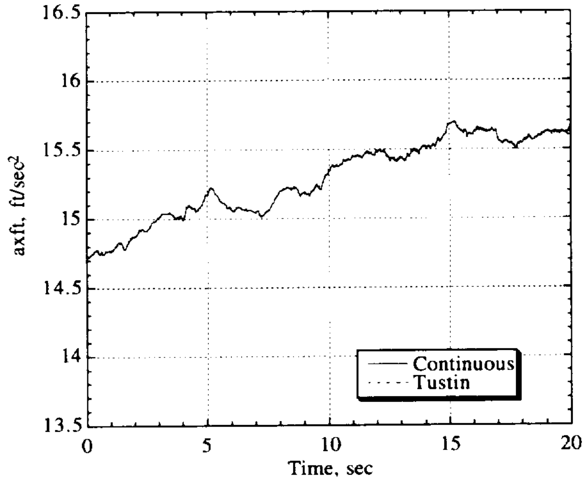
(a).- Angle of attack.



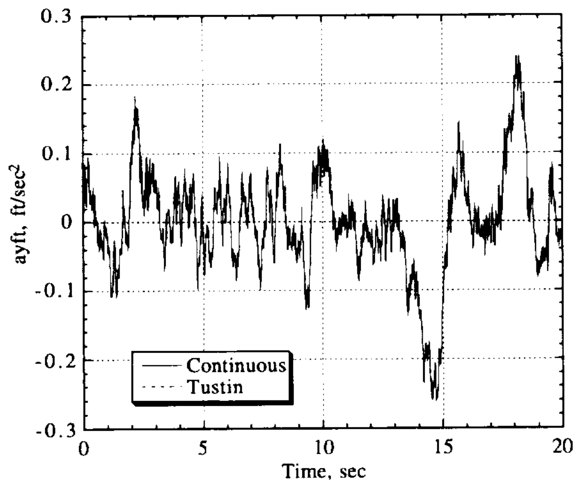
(b).- Sideslip angle.



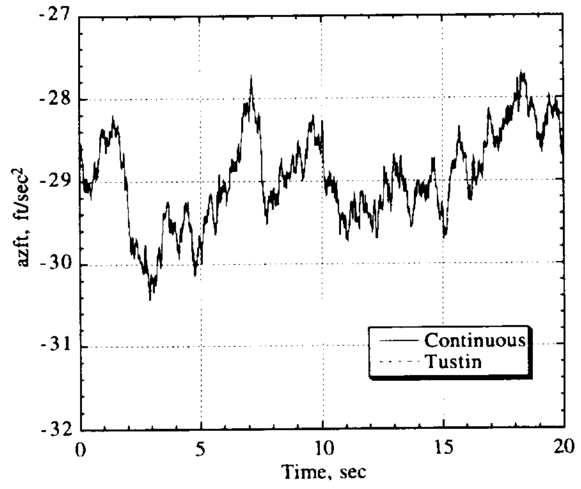
(c).- True airspeed.



(d).- X-axis acceleration.

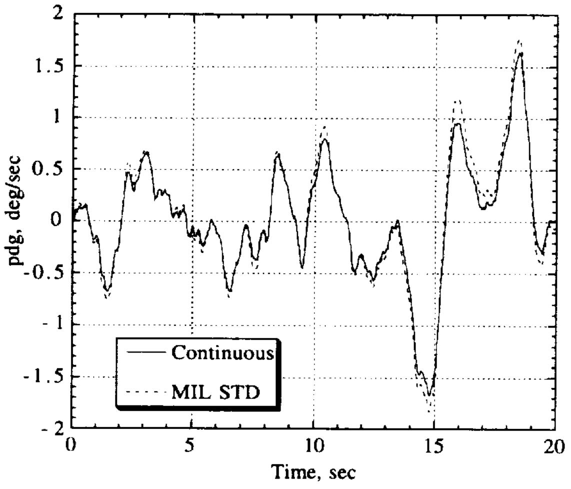


(e).- Y-axis acceleration.

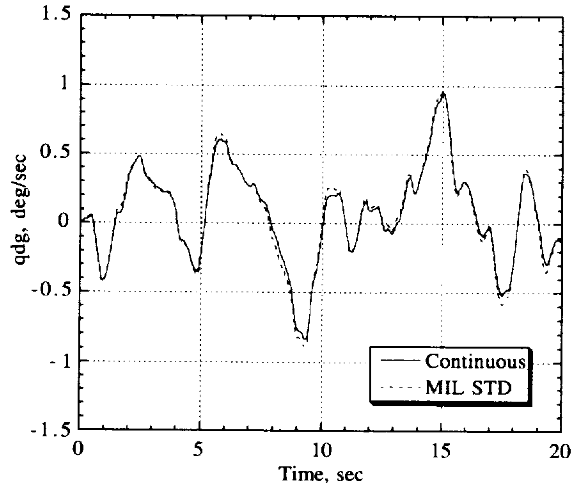


(f).- Z-axis acceleration.

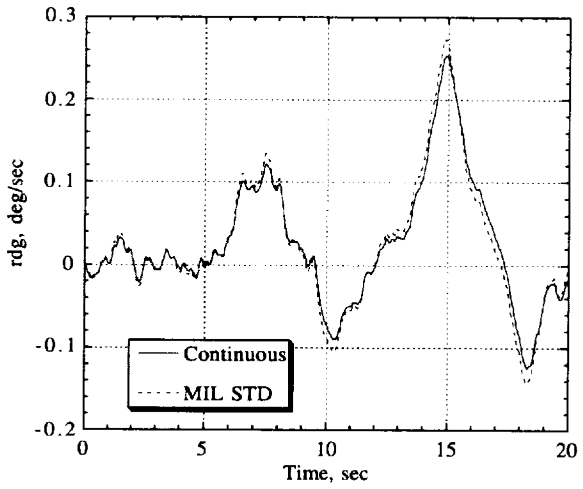
Figure 44. Comparison of continuous and Tustin air data and accelerations for $\alpha = 55°$, seed no. 1.



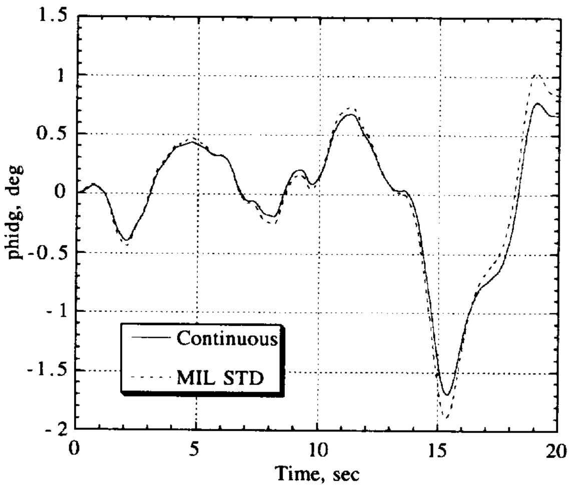
(a).- Roll rate.



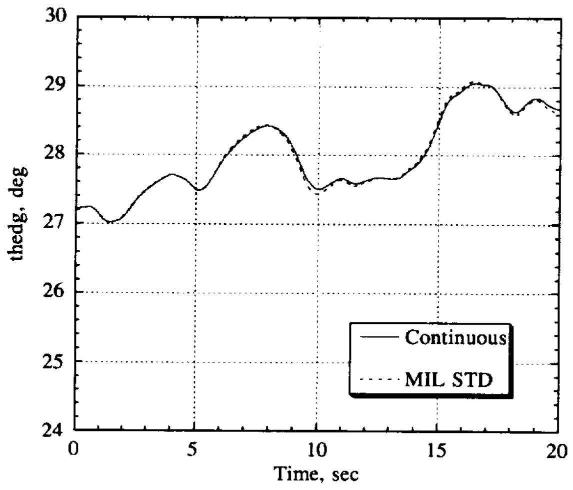
(b).- Pitch rate.



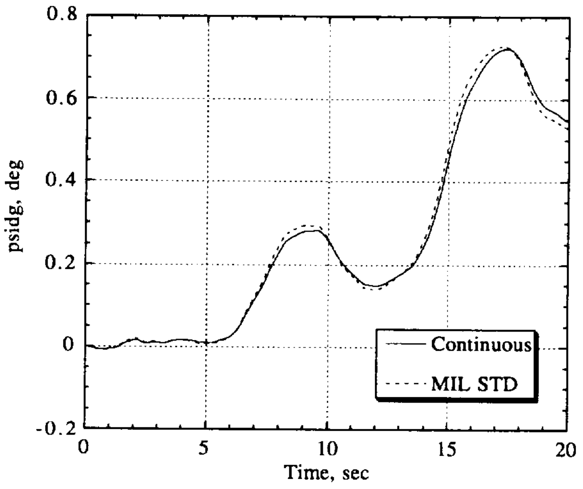
(c).- Yaw rate.



(d).- Bank angle.

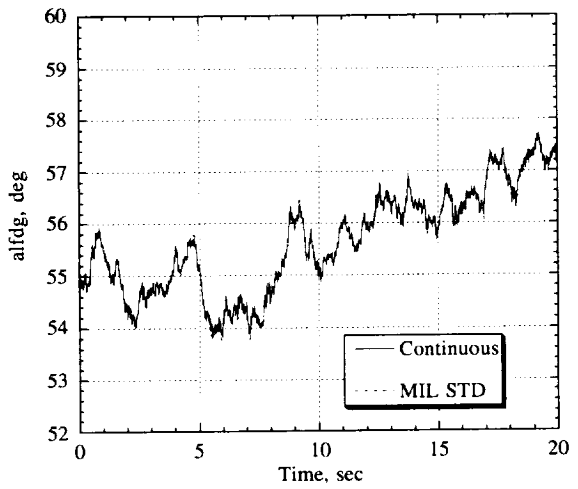


(e).- Pitch angle.

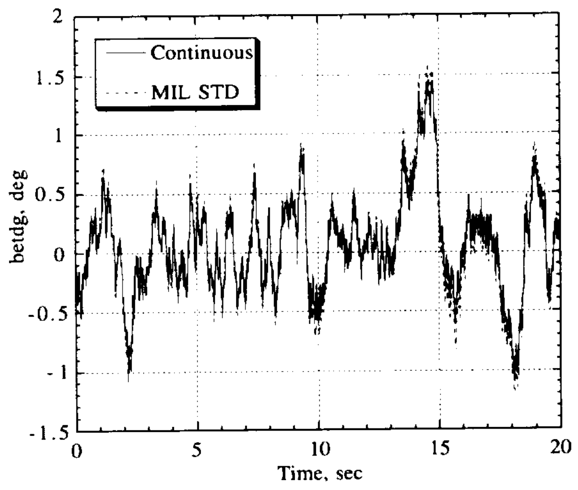


(f).- Heading.

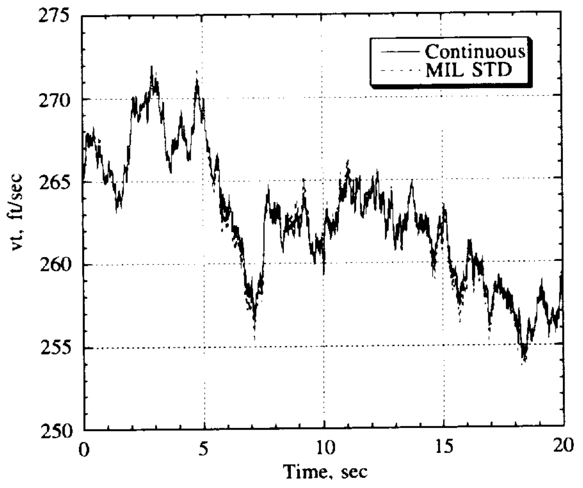
Figure 45. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 55°$, seed no. 1.



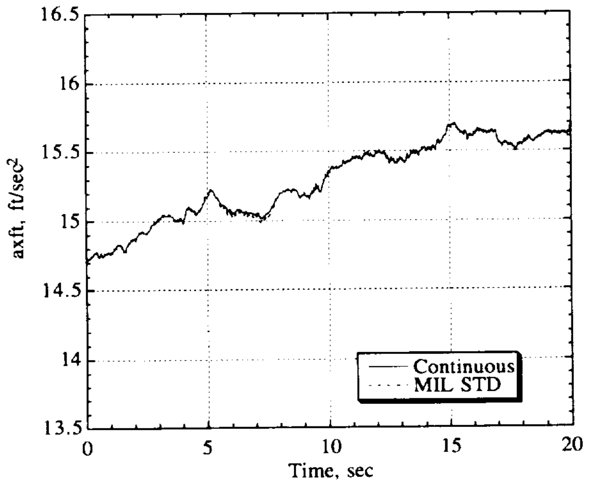
(a).- Angle of attack.



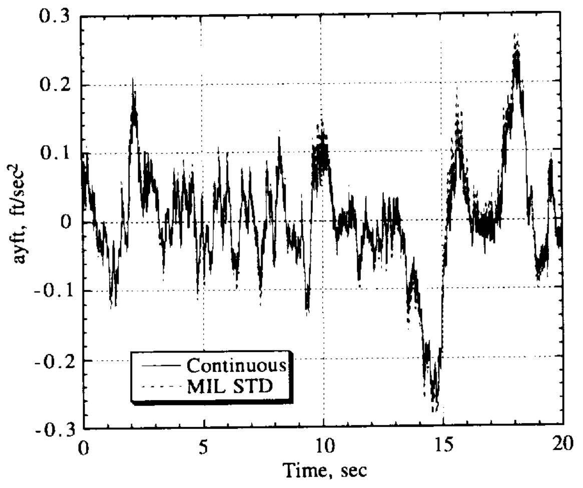
(b).- Sideslip angle.



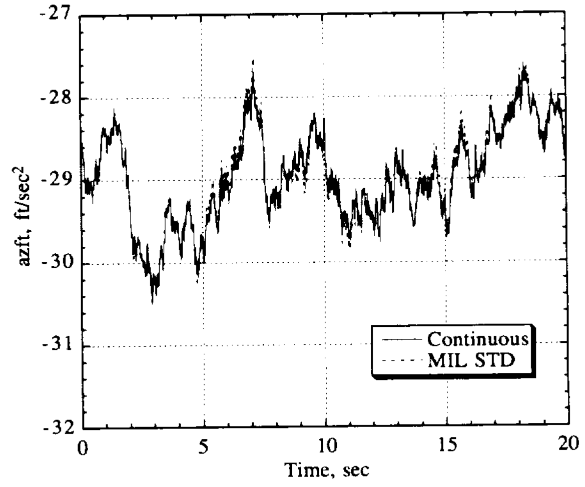
(c).- True airspeed.



(d).- X-axis acceleration.

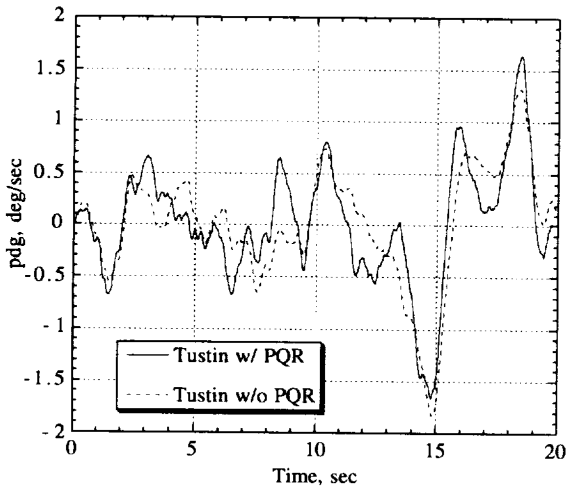


(e).- Y-axis acceleration.

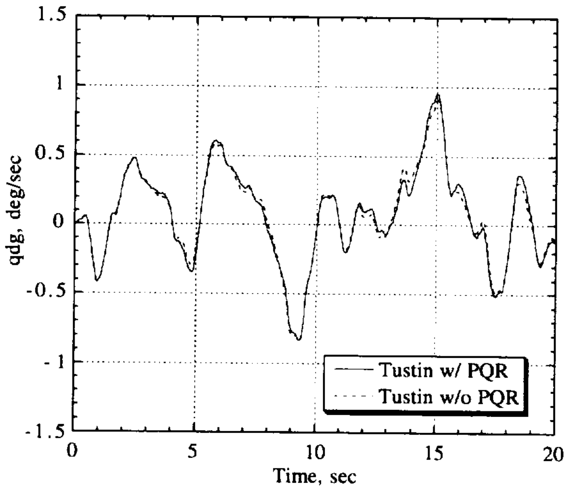


(f).- Z-axis acceleration.

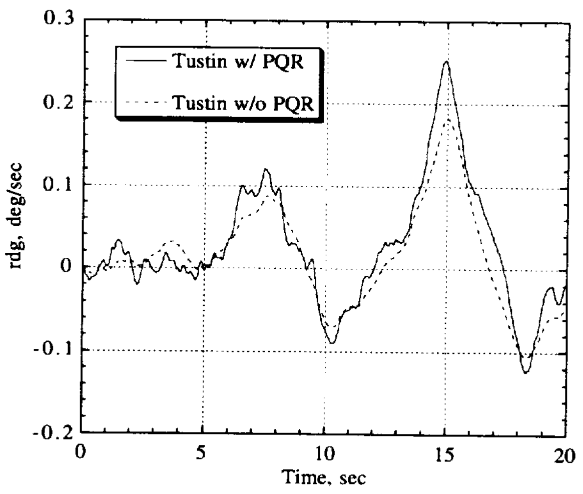
Figure 46. Comparison of continuous and Tustin air data and accelerations for $\alpha = 55°$, seed no. 1.



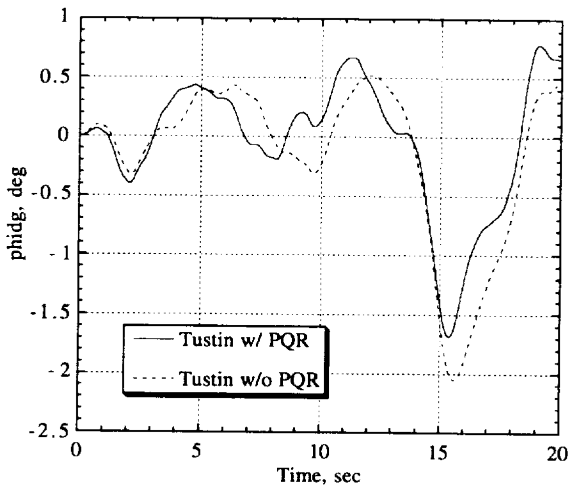
(a).- Roll rate.



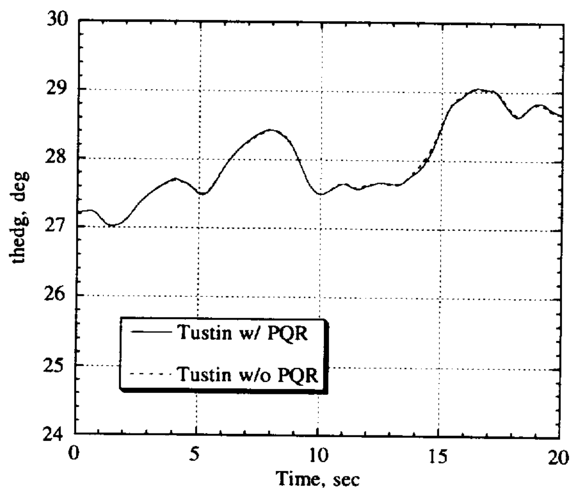
(b).- Pitch rate.



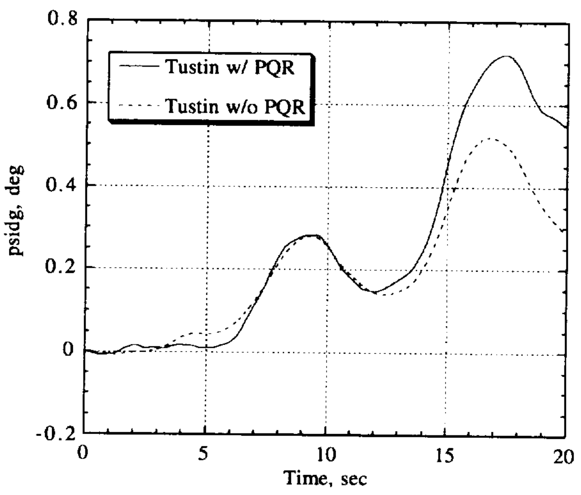
(c).- Yaw rate.



(d).- Bank angle.

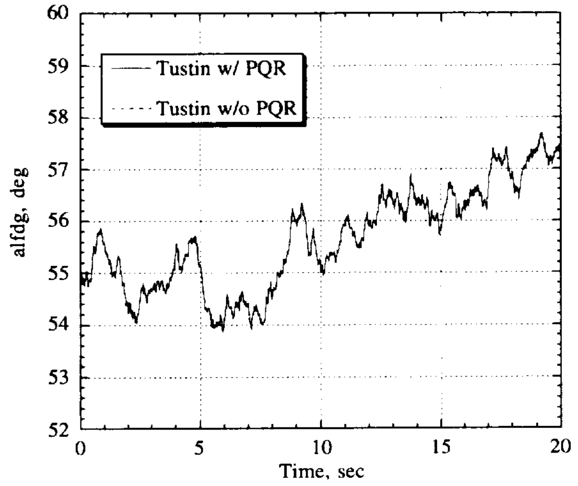


(e).- Pitch angle.

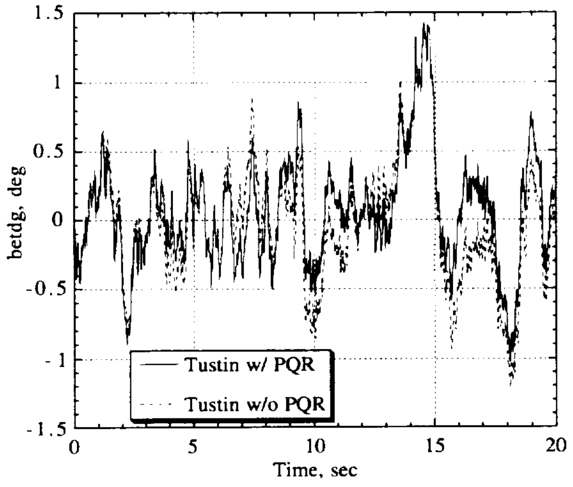


(f).- Heading.

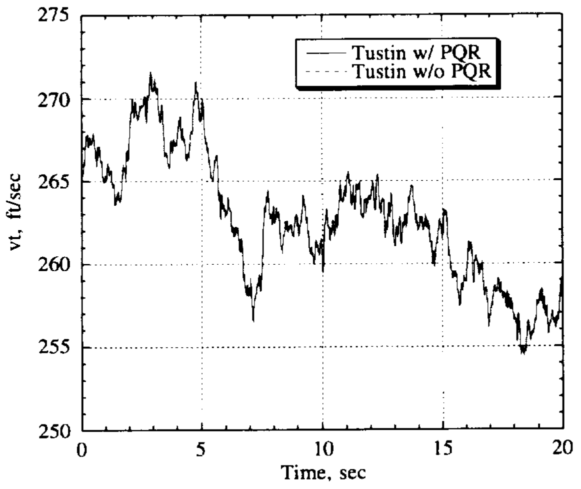
Figure 47. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 55°$, seed no. 1.



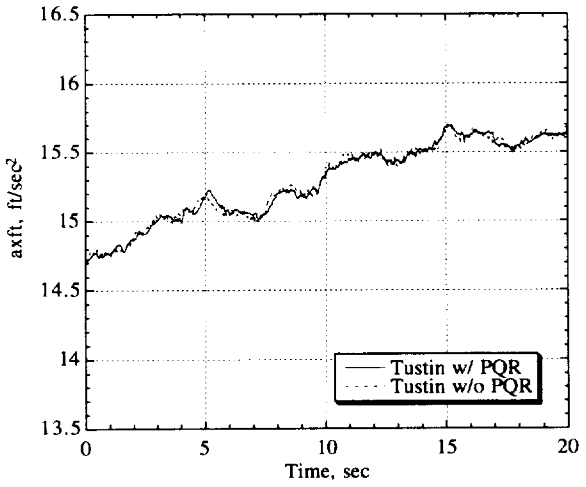
(a).- Angle of attack.



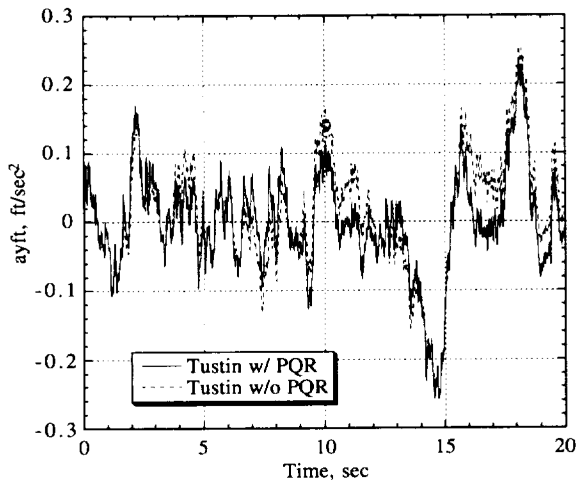
(b).- Sideslip angle.



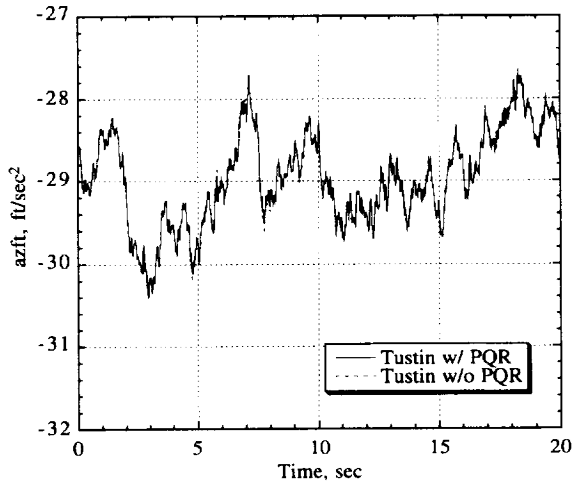
(c).- True airspeed.



(d).- X-axis acceleration.

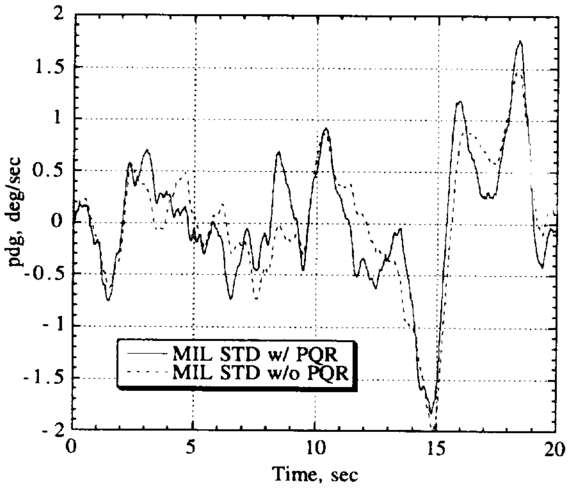


(e).- Y-axis acceleration.

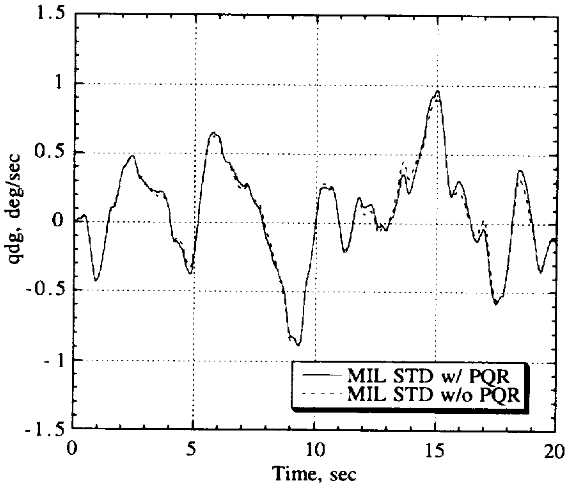


(f).- Z-axis acceleration.

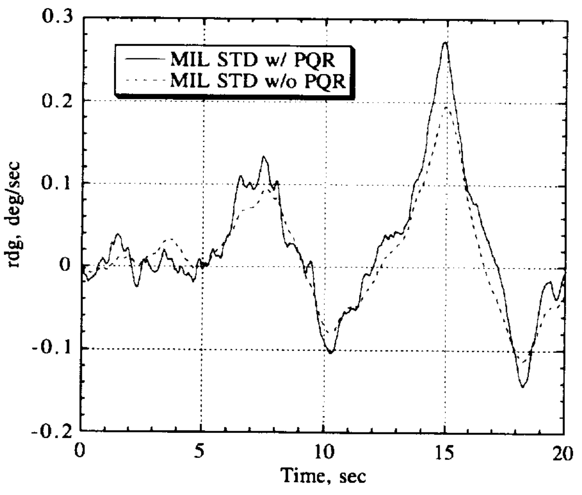
Figure 48. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 55°$, seed no. 1.



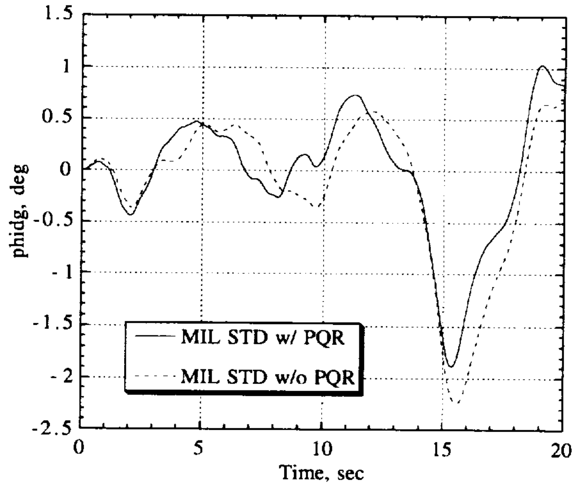
(a).- Roll rate.



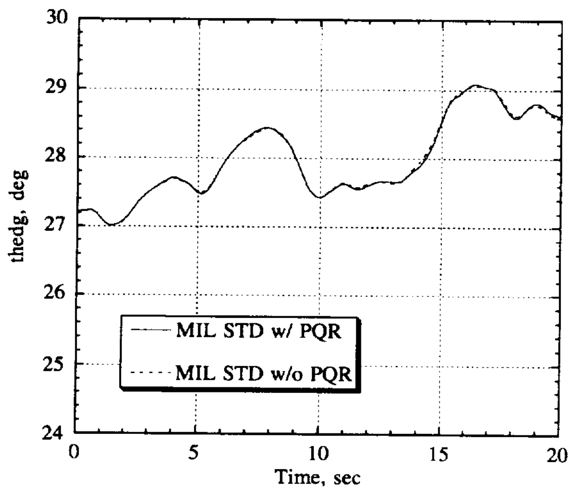
(b).- Pitch rate.



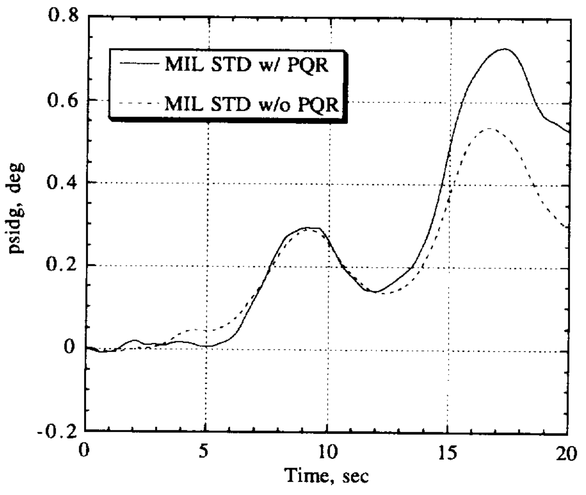
(c).- Yaw rate.



(d).- Bank angle.

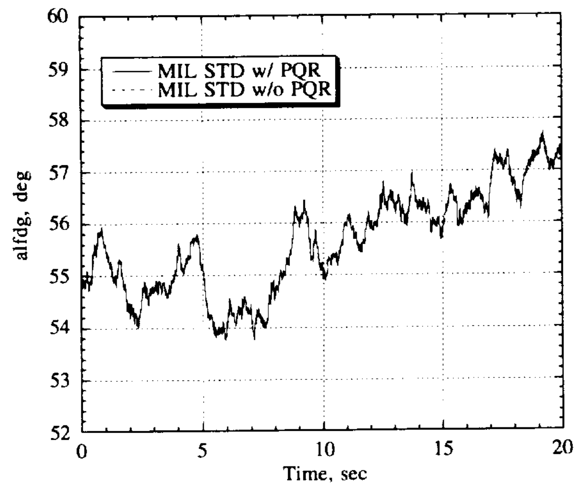


(e).- Pitch angle.

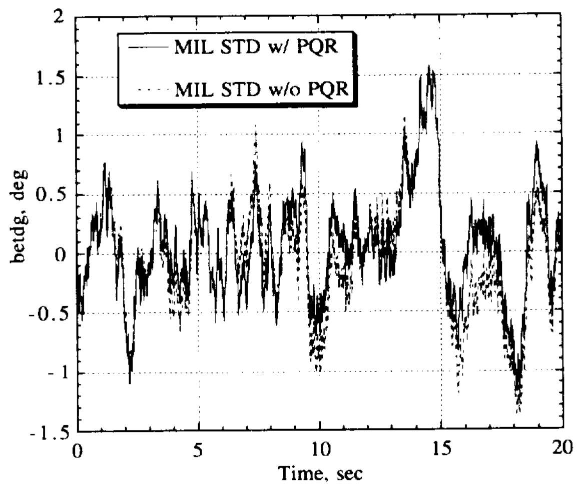


(f).- Heading.

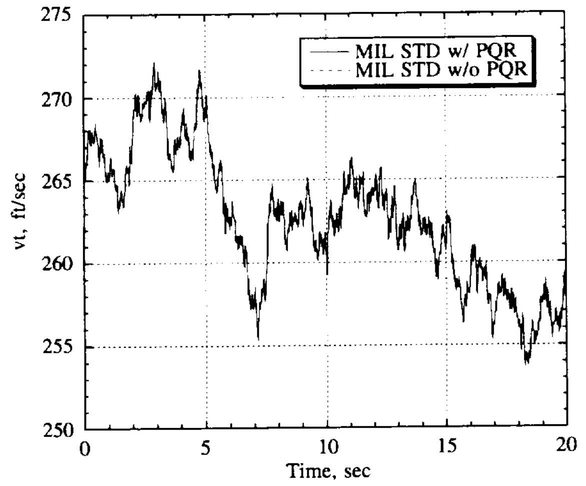
Figure 49. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 55°$, seed no. 1.



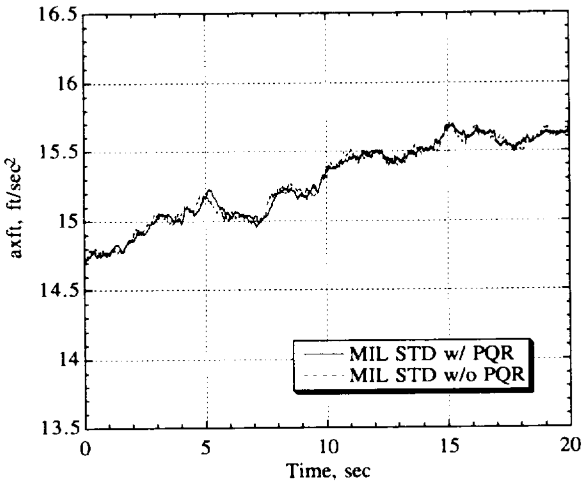
(a).- Angle of attack.



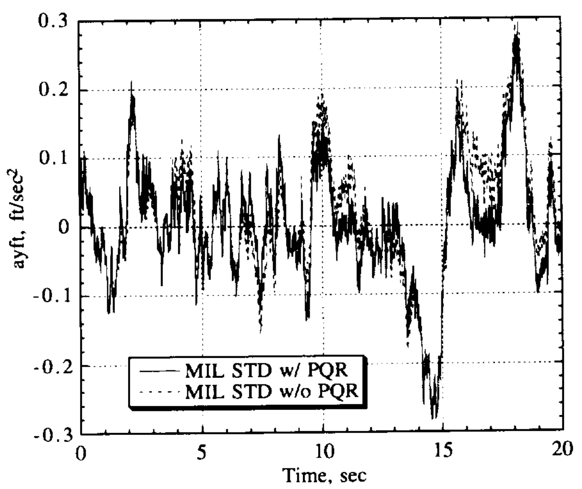
(b).- Sideslip angle.



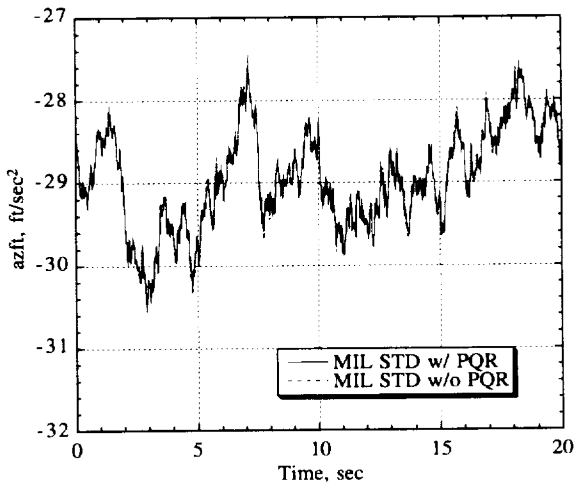
(c).- True airspeed.



(d).- X-axis acceleration.

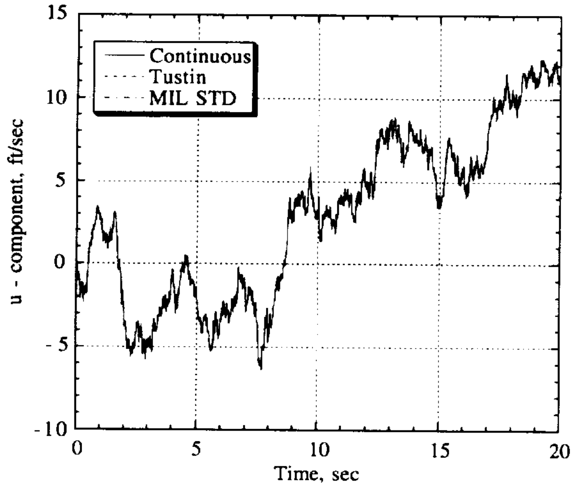


(e).- Y-axis acceleration.

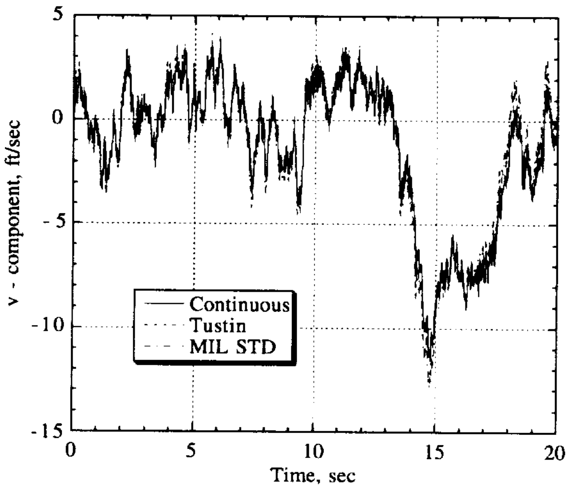


(f).- Z-axis acceleration.

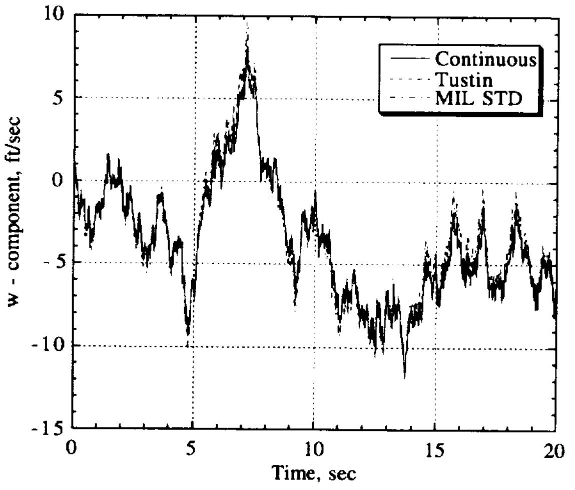
Figure 50. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 55°$, seed no. 1.



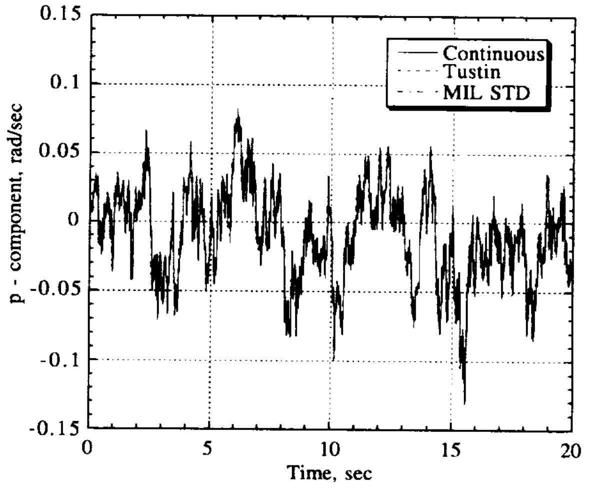
(a).- u-component.



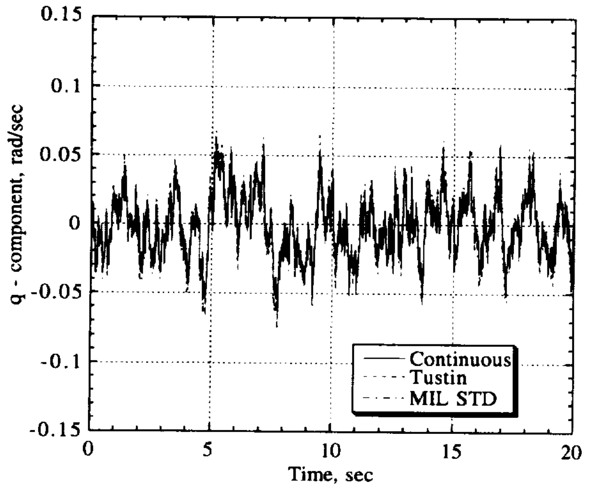
(b).- v-component.



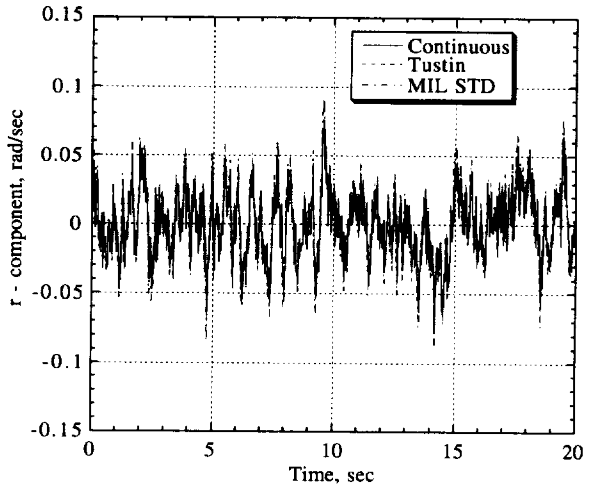
(c).- w-component.



(d).- p-component.

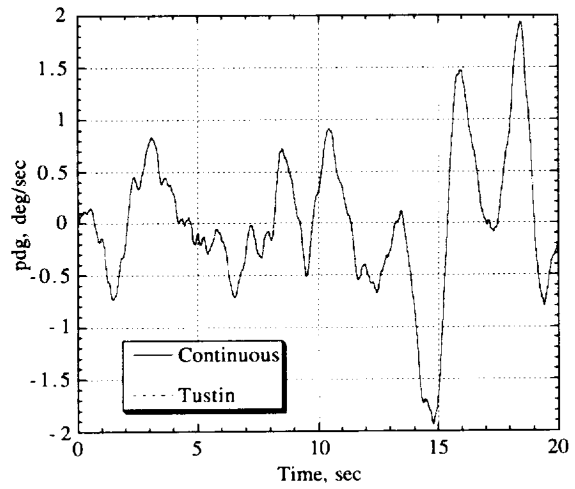


(e).- q-component.

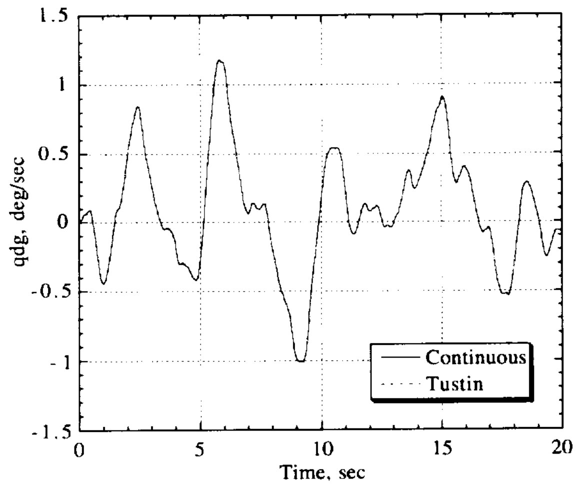


(f).- r-component.

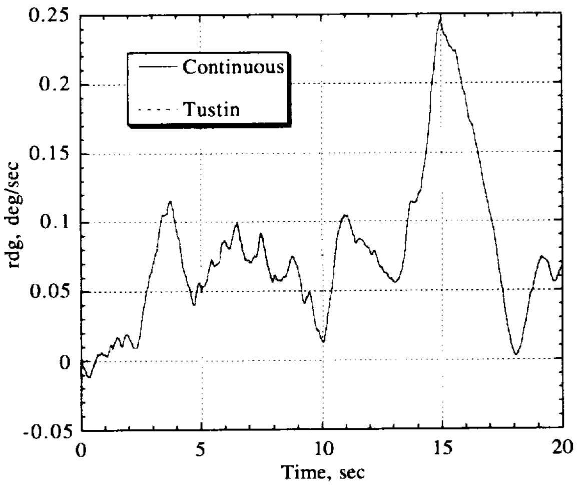
Figure 51. Comparison of continuous, Tustin, and MIL STD model turbulence for $\alpha = 60^\circ$, seed no. 1.



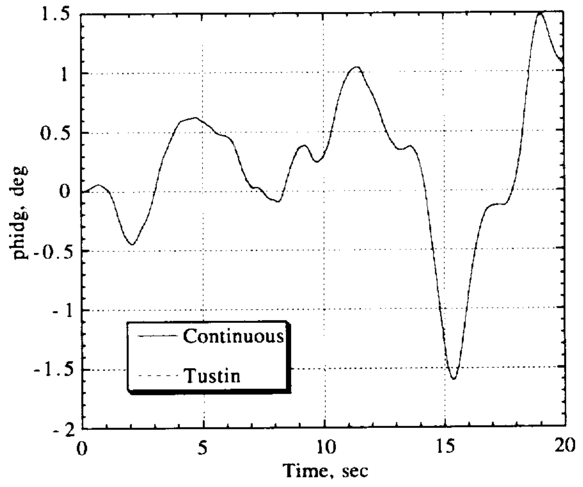
(a).- Roll rate.



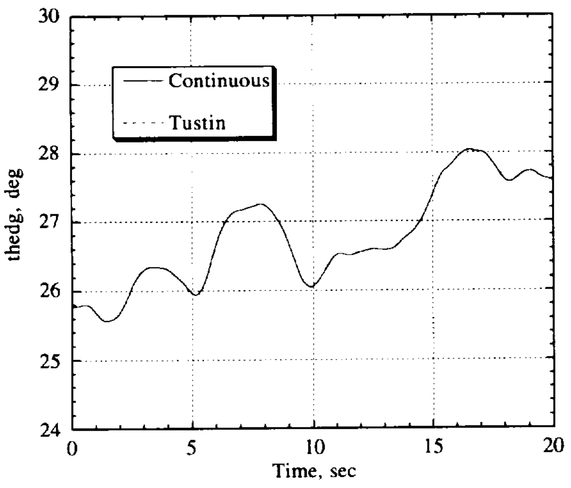
(b).- Pitch rate.



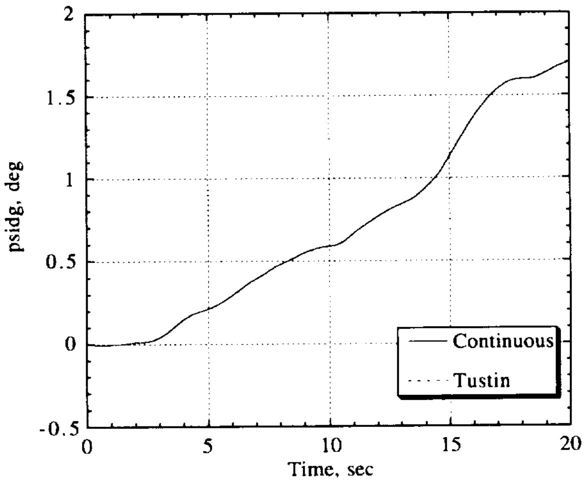
(c).- Yaw rate.



(d).- Bank angle.

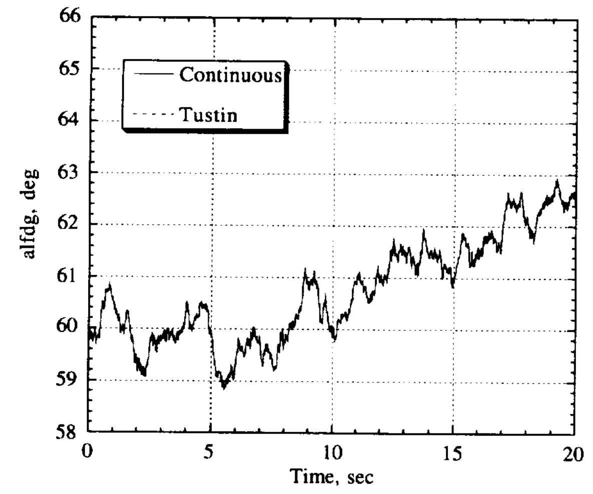


(e).-Pitch angle.

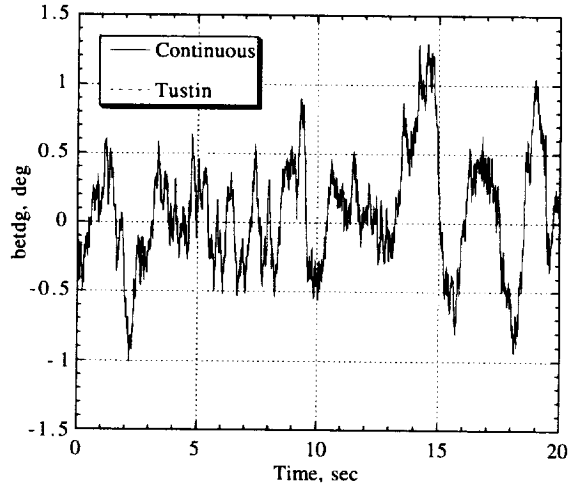


(f).-Heading.

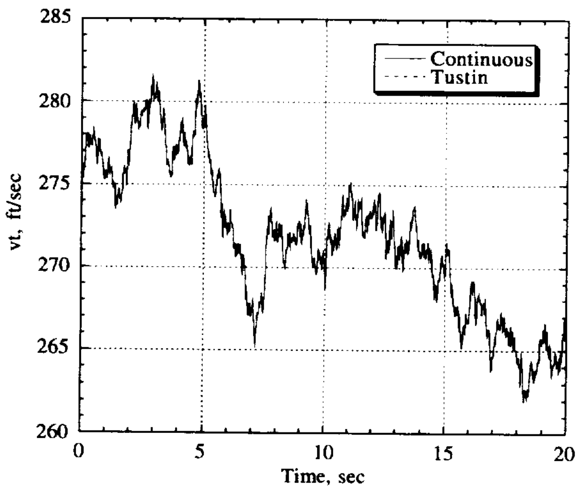
Figure 52. Comparison of continuous and Tustin attitude rates and angles for $\alpha = 60°$, seed no. 1.



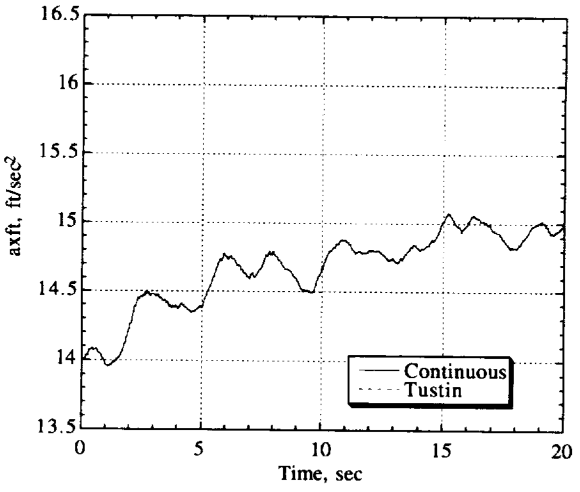
(a).- Angle of attack.



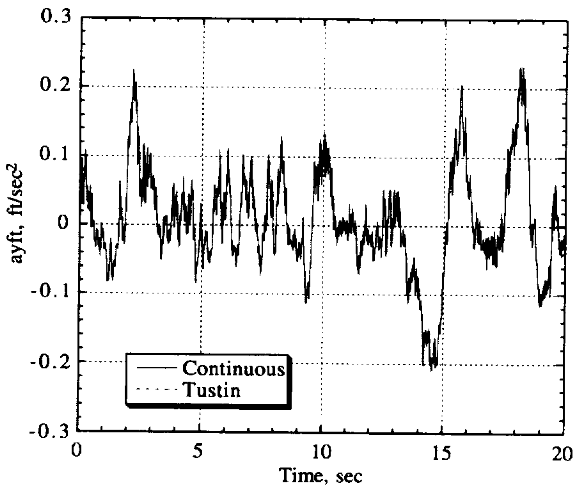
(b).- Sideslip angle.



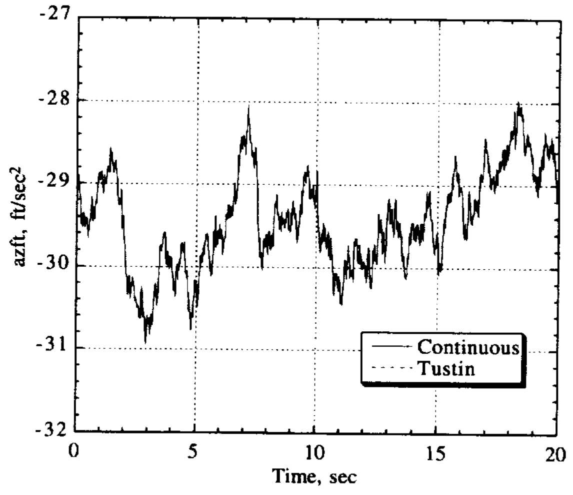
(c).- True airspeed.



(d).- X-axis acceleration.

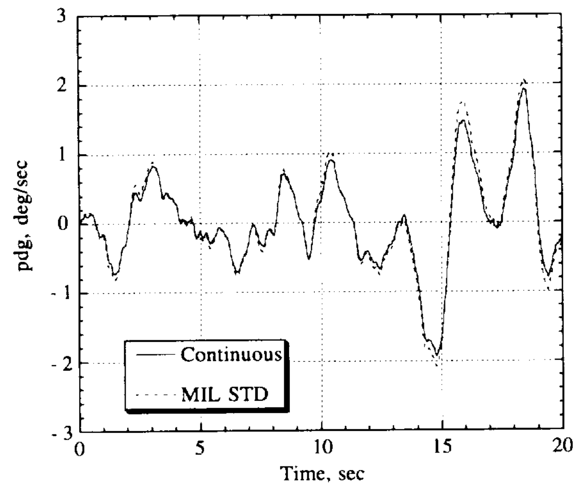


(e).- Y-axis acceleration.

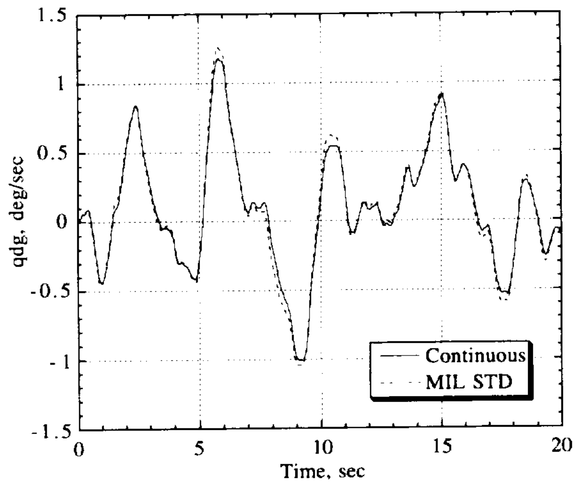


(f).- Z-axis acceleration.

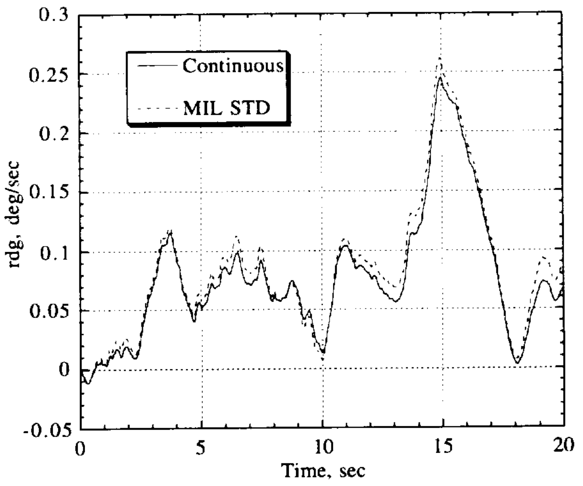
Figure 53. Comparison of continuous and Tustin air data and accelerations for $\alpha = 60°$, seed no. 1.



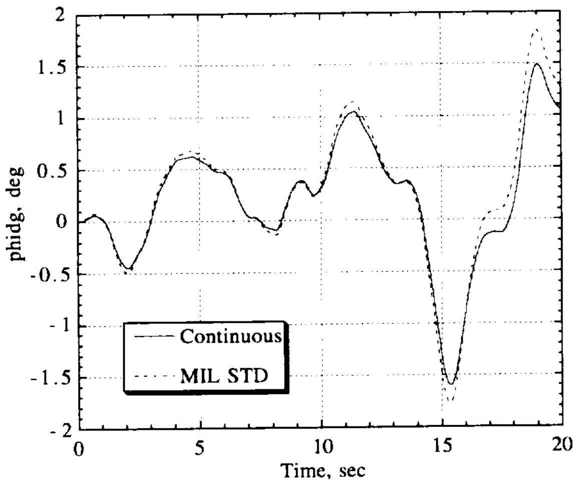
(a).- Roll rate.



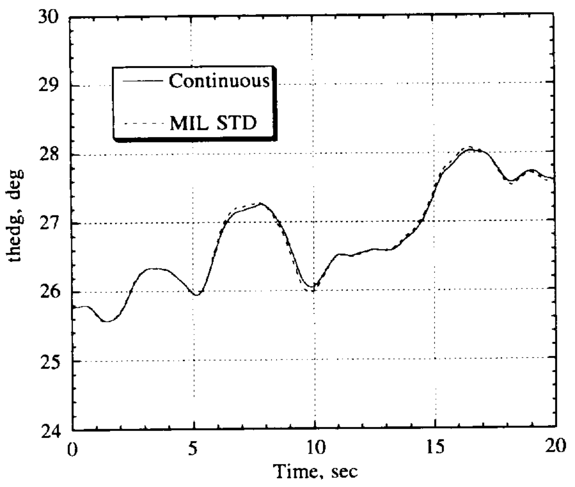
(b).- Pitch rate.



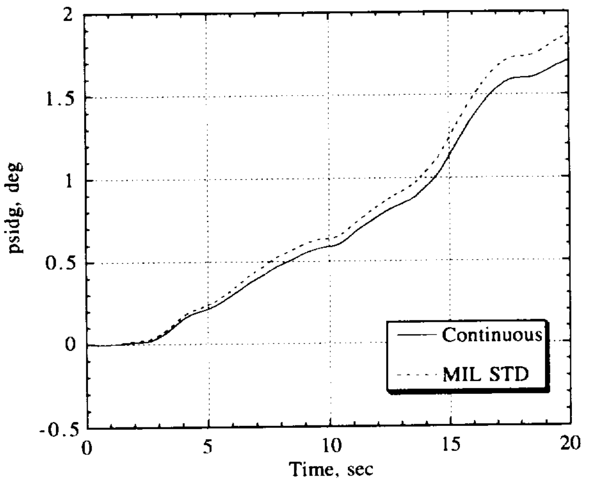
(c).- Yaw rate.



(d).- Bank angle.

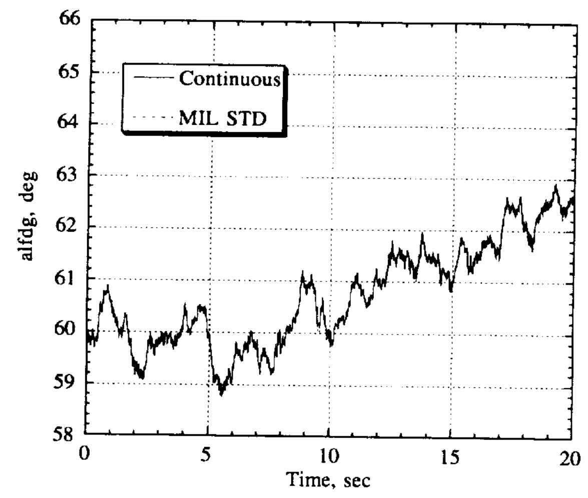


(e).- Pitch angle.

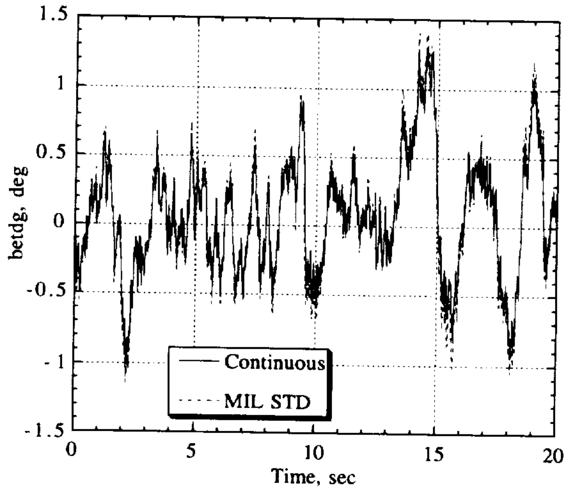


(f).- Heading.

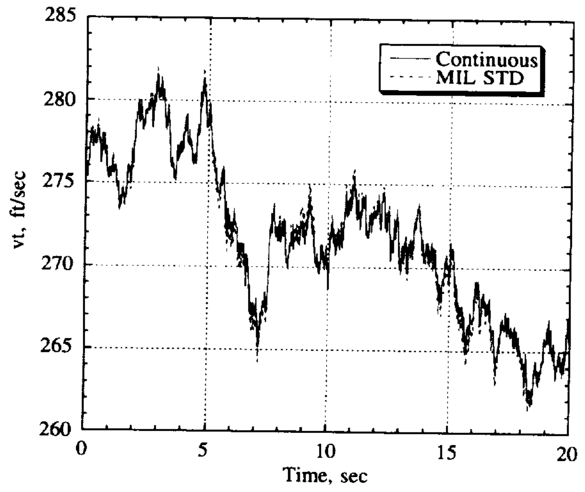
Figure 54. Comparison of continuous and MIL STD attitude rates and angles for $\alpha = 60°$, seed no. 1.



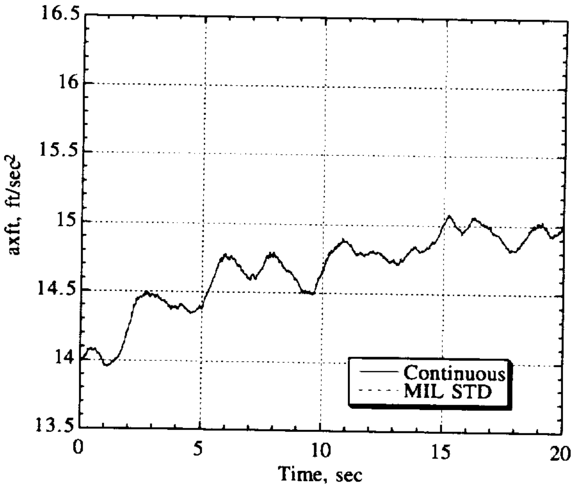
(a).- Angle of attack.



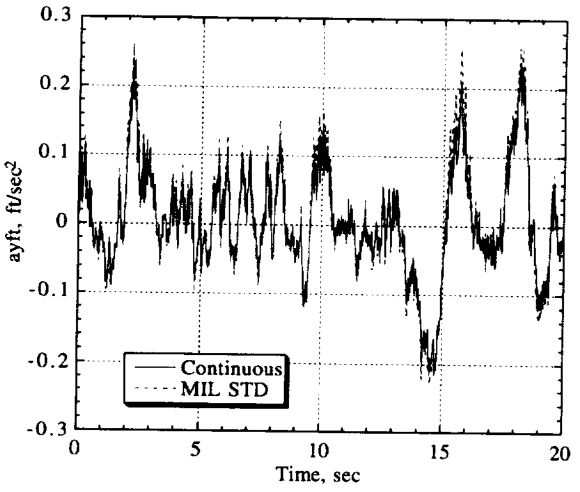
(b).- Sideslip angle.



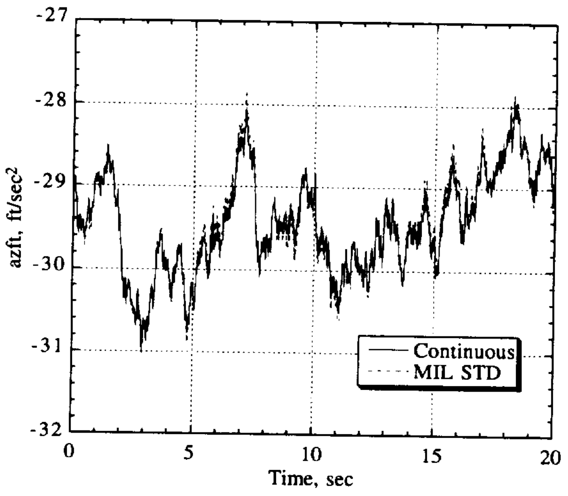
(c).- True airspeed.



(d).- X-axis acceleration.

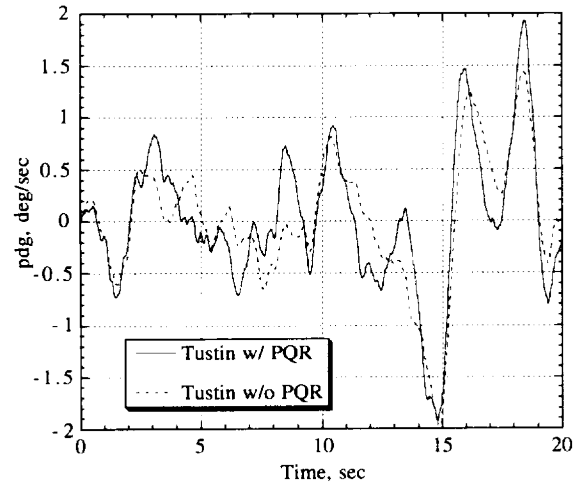


(e).- Y-axis acceleration.

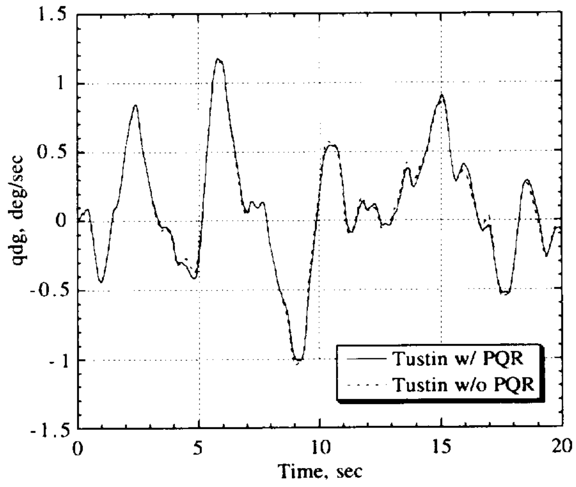


(f).- Z-axis acceleration.

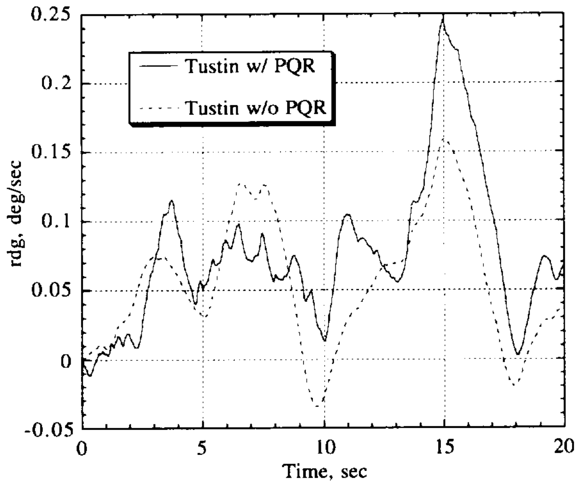
Figure 55. Comparison of continuous and Tustin air data and accelerations for $\alpha = 60°$, seed no. 1.



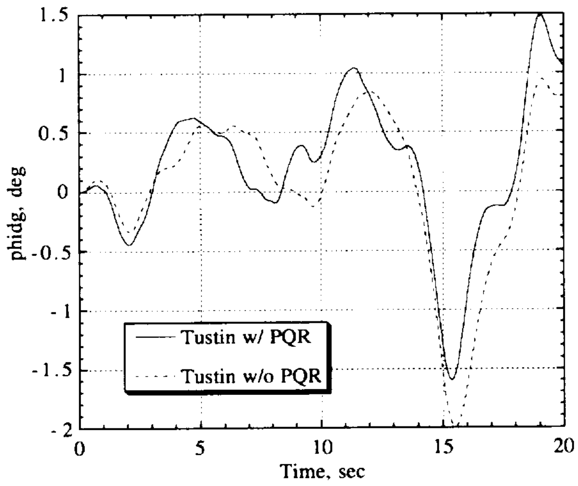
(a).- Roll rate.



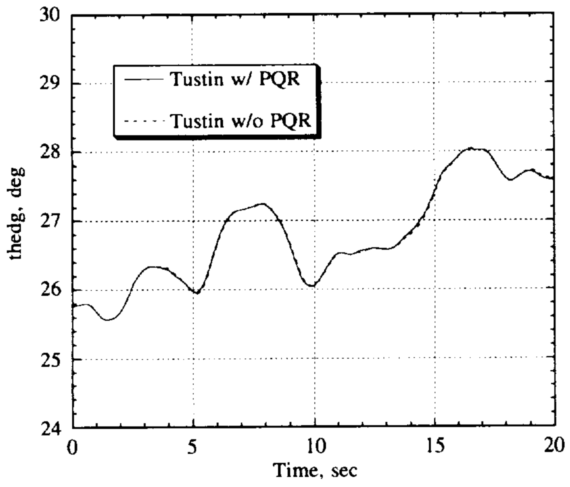
(b).- Pitch rate.



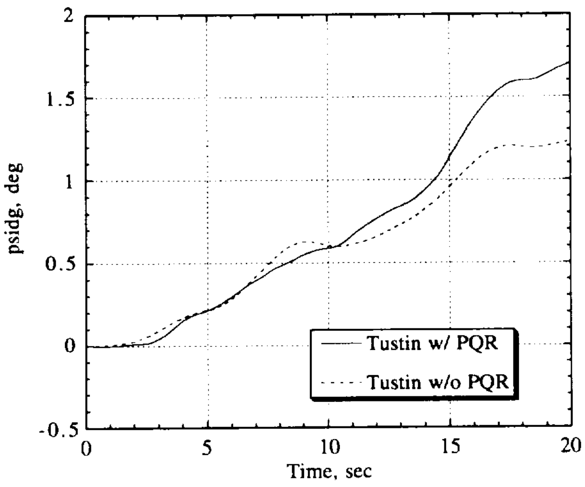
(c).- Yaw rate.



(d).- Bank angle.

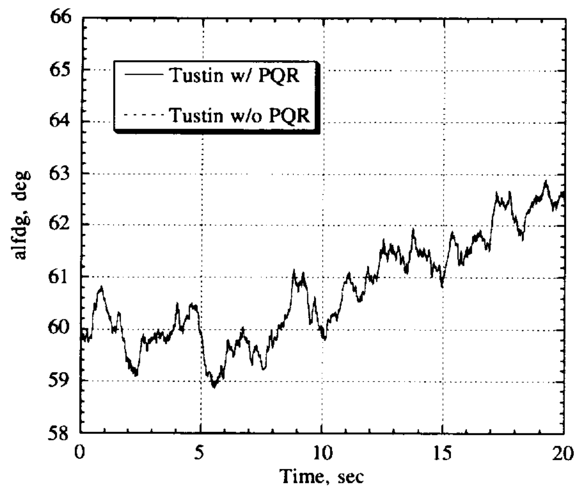


(e).- Pitch angle.

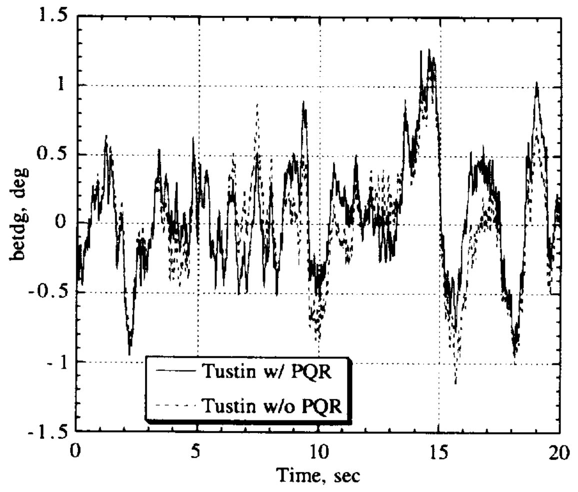


(f).- Heading.

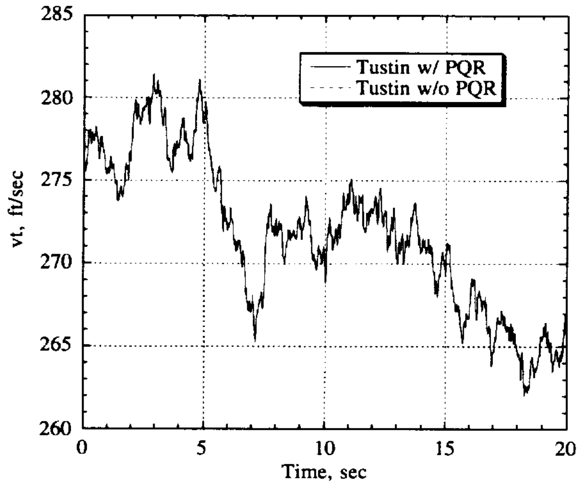
Figure 56. Comparison of Tustin attitude rates and angles w/ and w/o PQR gusts for $\alpha = 60°$, seed no. 1.



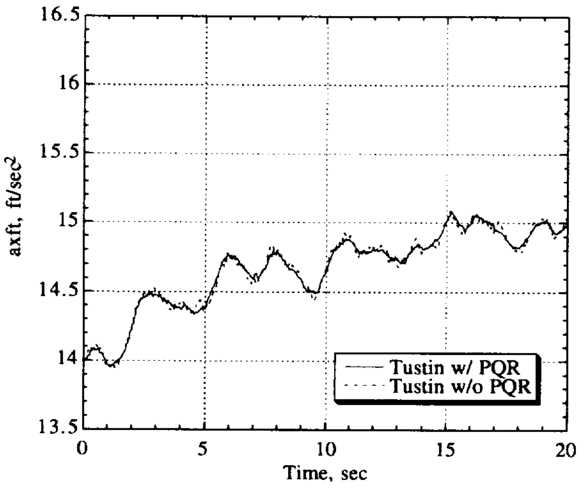
(a).- Angle of attack.



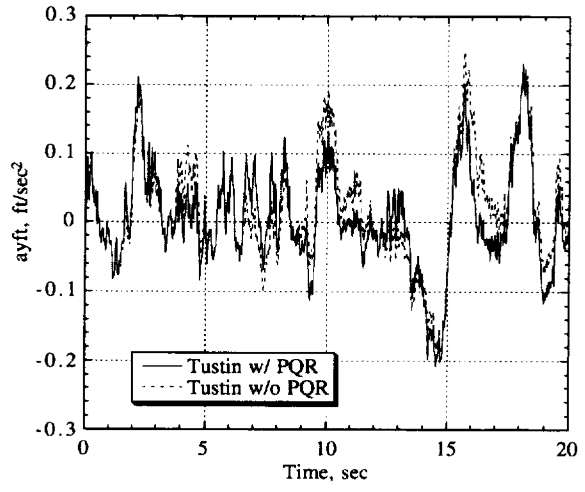
(b).- Sideslip angle.



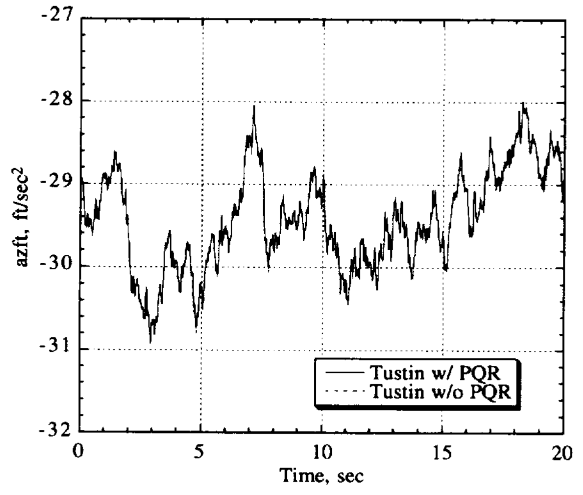
(c).- True airspeed.



(d).- X-axis acceleration.

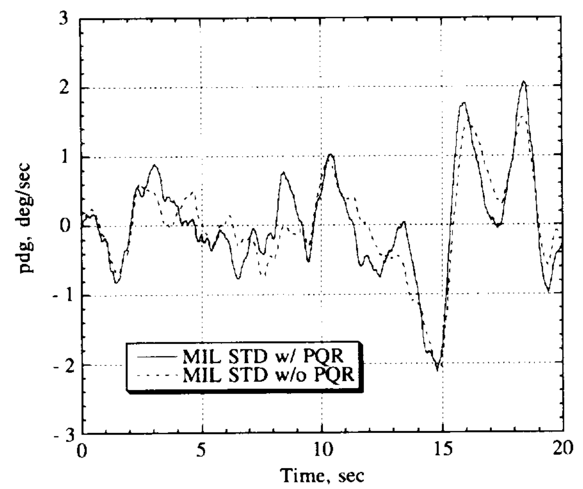


(e).- Y-axis acceleration.

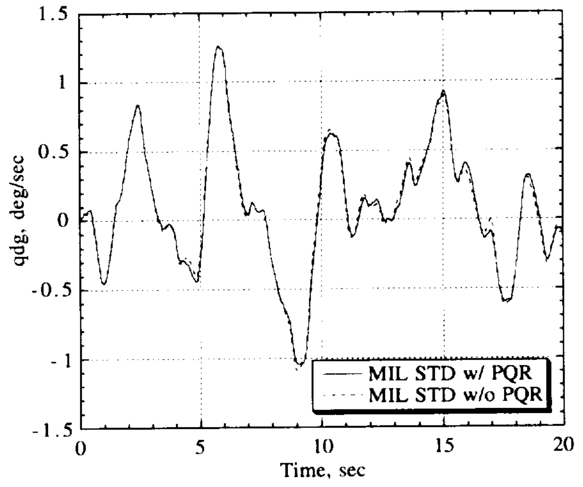


(f).- Z-axis acceleration.

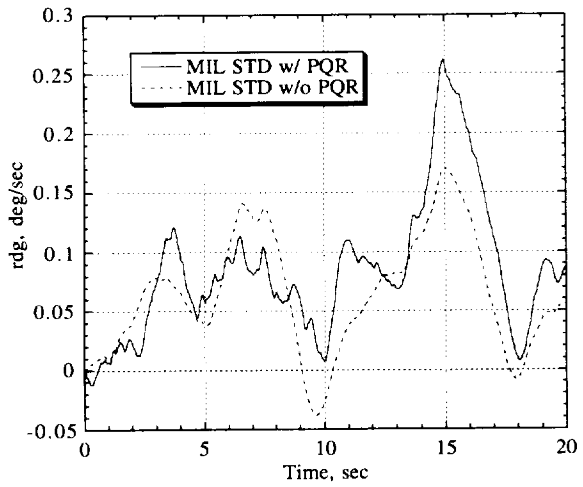
Figure 57. Comparison of Tustin air data and accelerations w/ and w/o PQR gusts for $\alpha = 60°$, seed no. 1.



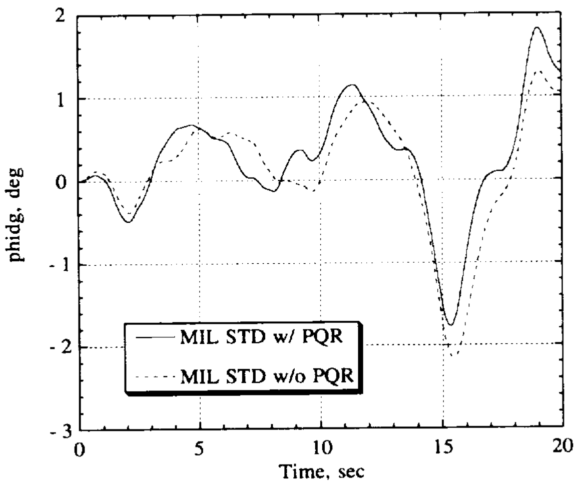
(a).- Roll rate.



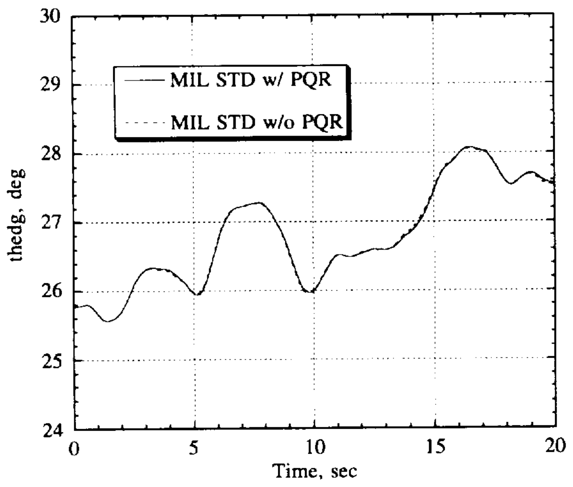
(b).- Pitch rate.



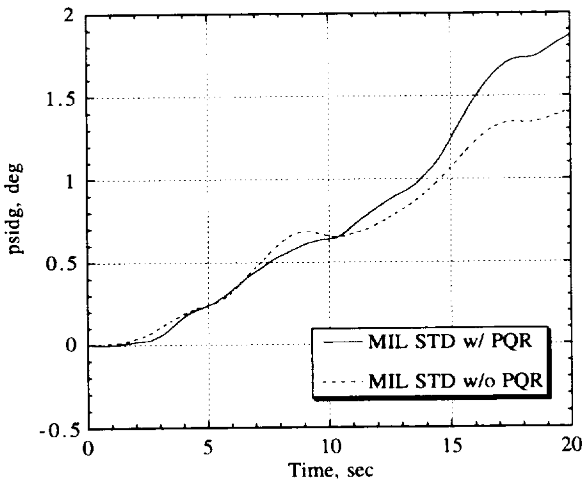
(c).- Yaw rate.



(d).- Bank angle.

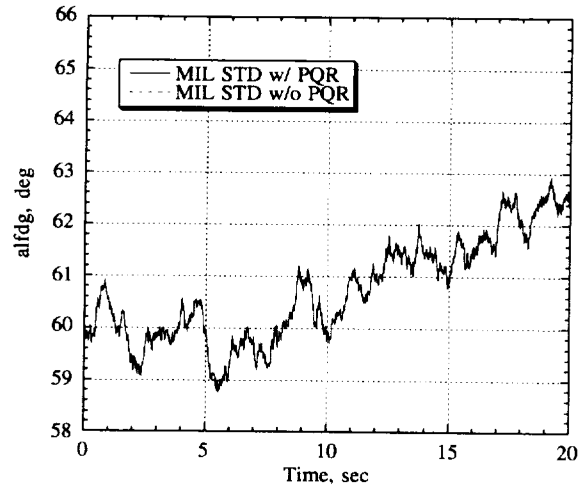


(e).- Pitch angle.

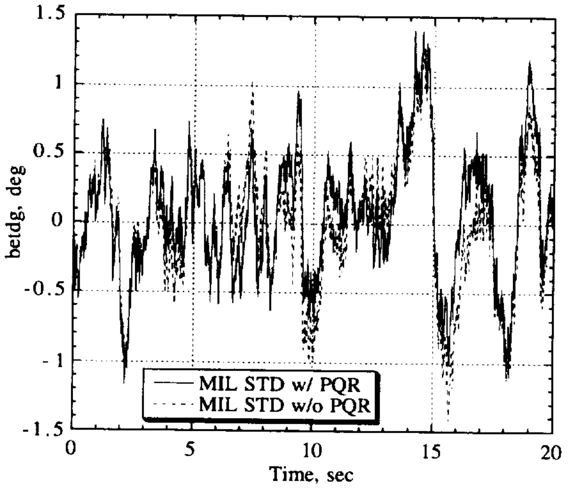


(f).- Heading.

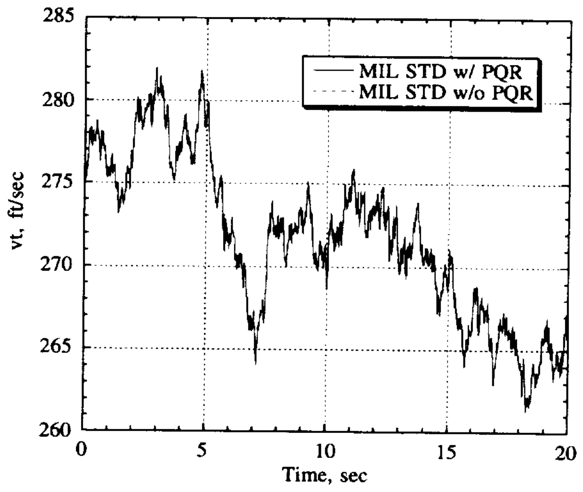
Figure 58. Comparison of MIL STD attitude rates and angles w/ and w/o PQR gusts for $\alpha = 60°$, seed no. 1.



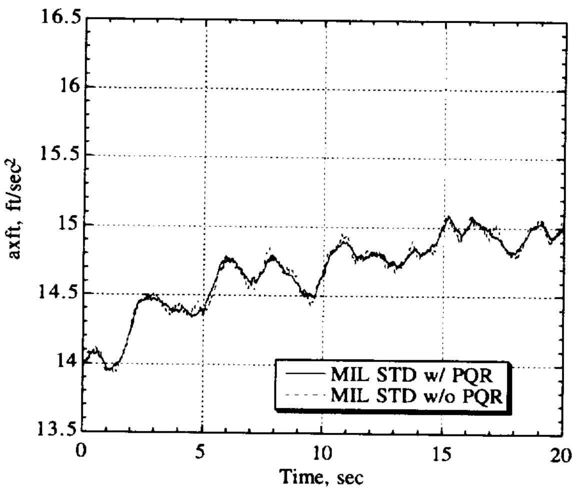
(a).- Angle of attack.



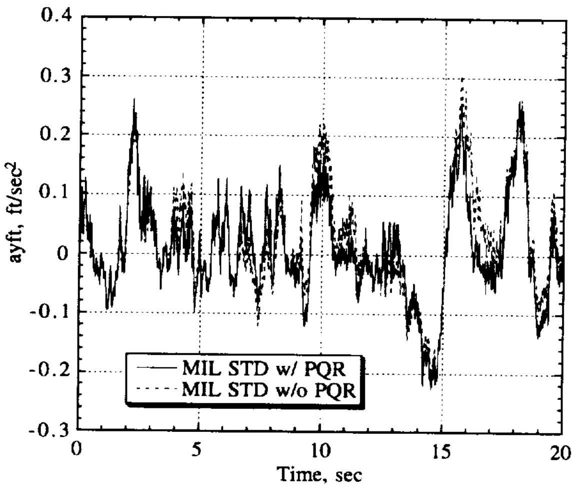
(b).- Sideslip angle.



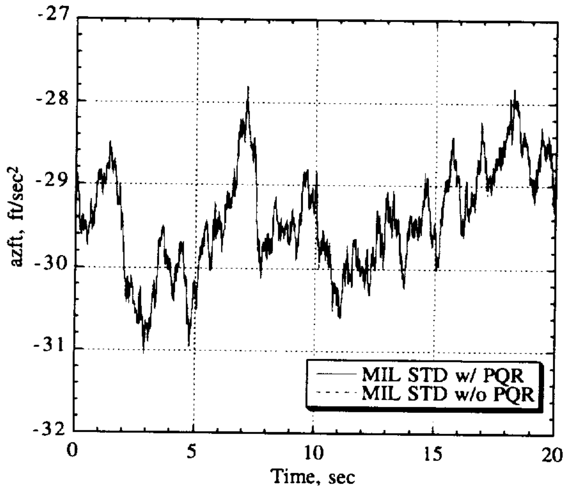
(c).- True airspeed.



(d).- X-axis acceleration.



(e).- Y-axis acceleration.



(f).- Z-axis acceleration.

Figure 59. Comparison of MIL STD air data and accelerations w/ and w/o PQR gusts for $\alpha = 60°$, seed no. 1.

References

1. Anon.: *Military Standard - Flying Qualities for Piloted Aircraft.* MIL-STD-1797A, Jan. 30, 1990.
2. Hoblit, Frederic M.: *Gust Loads on Aircraft: Concepts and Applications.* AIAA, 1988.
3. Buttrill, C. S.; Arbuckle, P. D.; and Hoffler, K. D.: *Simulation Model of a Twin-Tail, High Performance Airplane.* NASA TM-107601, 1992.
4. Messina, Michael D.; Strickland, Mark E.; Hoffler, Keith D.; Carzoo, Susan W.; Bundick, W. Thomas; Yeager, Jessie C.; and Beissner, Jr., Fred L.: *Simulation Model of the F/A-18 High Angle-of-Attack Research Vechicle Utilized for the Design of Advanced Control Laws.* NASA TM-110216, May 1996.
5. *Advanced Continuous Simulation Language (ACSL) Reference manual.* Edition 10.0, Mitchell & Gauthier Associates, 1991.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1998	Contractor Report

4. TITLE AND SUBTITLE
Implementation and Testing of Turbulence Models for the F18-HARV Simulation

5. FUNDING NUMBERS
522-35-11-03
NAS1-96014

6. AUTHOR(S)
Jessie C. Yeager

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Engineering and Sciences
Mail Stop 371
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
NASA Langley Research Center
Hampton, VA 23681-2199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR-1998-206937

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: W. Thomas Bundick

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 08 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
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.

14. SUBJECT TERMS
Dryden spectral model, Dryden turbulence model, aircraft simulation, Tustin Transform

15. NUMBER OF PAGES
147

16. PRICE CODE
A07

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	