NASA/TM-2014-218513

Data Reduction Functions for the Langley 14- by 22-Foot Subsonic Tunnel

Andy D. Boney
Langley Research Center, Hampton, Virginia

August 2014

NASA STI Program . . . in Profile

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2014-218513

Data Reduction Functions for the Langley 14- by 22-Foot Subsonic Tunnel

Andy D. Boney
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

August 2014

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Table of Contents

Abstract

The Langley 14- by 22-Foot Subsonic Tunnel's data reduction software utilizes six major functions to compute the acquired data. These functions calculate engineering units, tunnel parameters, flowmeters, jet exhaust measurements, balance loads/model attitudes, and model /wall pressures. The input(required) variables, the output(computed) variables, and the equations and/or subfunction(s) associated with each major function are discussed.

Introduction

In the Langley 14- by 22-Foot Subsonic Tunnel, data acquisition consists of analog, digital, pressure, and automated cart data. Analog and digital data is acquired with the NEFF 600, pressure data is acquired with the ESP 8400, and automated cart data is acquired using the Channel Access software package. After data acquisition, data reduction functions are utilized to compute engineering units, tunnel parameters, flowmeters, jet exhaust measurements, balance loads/model attitudes, and model /wall pressures. For consistency, the names written to an online output data file are used as much as possible in this document. All calculations are single precision except where noted.

Engineering Units’ Conversions

Analog channel data is acquired and converted to millivolts. Next, the data is converted to engineering units. The engineering units’ conversions commonly used in the 14- by 22-Foot Subsonic Tunnel for analog data are *asin*, *lin*, *poly*, and *type_j*.

ASIN

***** ***Equation Definition***

value = ((chan#### - chan####$_{woz}$) – bias) / sensitivity
if (value > 1.0) then
value = 1.0
if (value < -1.0) then
value = -1.0
eu = (dasin(value) * 57.2957795131) - offset

***** ***Setup Example***

name,FALPHA chno,3 dau,Neff
desc,Model pitch
units,deg
euconv,asin
lce(1.233,260.45,-0.102)
range,160
filter,1
cmpllmt,-10
cmpulmt,10
dztype, no

***** ***Equation Example***

value = (chan0003 – 1.233) / 260.45
if (value > 1.0) then
value = 1.0
f (value < -1.0) then
value = -1.0
FALPHA = (dasin(value) * 57.2957795131) + 0.0102

LIN

*   ***Equation Definition***

eu = ((chan#### - chan####$_{woz}$)
    * (expected power supply voltage /
      actual power supply voltage) * sensitivity) + offset

*   ***Setup Example 1***

```
name,TA     chno,1     dau,Neff
        desc,tunnel ambient temperature
        units,degf
        euconv,lin
        lce(0.036,-40)
        range,5120
        filter,1
        dztype, no
```

*   ***Equation Example 1***

TA = (chan0001 * 0.036) – 40.0

*   ***Setup Example 2***

```
name,NF738     chno,10     dau,Neff
desc,Body Balance Normal Force
units,lbs
euconv,lin
lce(141.15448,0)
filter,1
cmpllmt,-400
cmpulmt,400
dztype, wind
pscorr(V738,5)
```

*   ***Equation Example 2***

NF738 = ((chan0010 * (5.0 / V738)) –
    (chan0010$_{woz}$ * (5.0 / V738$_{woz}$))) * 141.15448

POLY

* ***Equation Definition***

$$
\begin{aligned}
eu = constant1 &+ ((chan\#\#\#\# - chan\#\#\#\#_{woz}) * constant2) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^2 * constant3) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^3 * constant4) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^4 * constant5) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^5 * constant6) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^6 * constant7) \\
&+ ((chan\#\#\#\# - chan\#\#\#\#_{woz})^7 * constant8)
\end{aligned}
$$

* ***Setup Example***

```
name,SROLL     chno,43     dau,Neff
        desc,IRTS Shaft Roll
        units,deg
        euconv,coef
        poly,3
        lce(0.1847,-0.039,-0.2865,0.3239)
        range,160
        filter,1
        cmpllmt,-0.5
        cmpulmt,0.5
        dztype, no
```

* ***Equation Example***

$$
\begin{aligned}
SROLL = 0.1847 &+ (chan0043 * (-0.039)) \\
&+ ((chan0043)^2 * (-0.2865)) \\
&+ ((chan0043)^3 * 0.3239))
\end{aligned}
$$

TYPE_J

****** ***Equation Definition***

****** ***type_j denotes iron-constantan***

value = (chan#### + jcn)
** execute subfunction* ***tcic(value, temp)***
eu = temp

* ***Setup Example***

name,ESPTMP52   chno,65   dau,Neff
desc,Temp of ESP52
units,degf
euconv,type_j
range,10
filter,1
dztype, no
jcn,0

****** ***Equation Example***

value = chan0065
** execute subfunction* ***tcic(value, temp)***
ESPTMP52 = temp

* *note*   type_k denotes chromel-alumel; execute subfunction **tcca**

* *note*   type_t denotes copper-constantan; execute subfunction **tccc**

Tunnel Parameters

Input Variables

ccust – customer dynamic pressure calibration constant (default = 0.0)
dpicut – dpils/dpihs cutoff parameter (psf)
dpihs – differential pressure of sidewall static pressure in the entrance cone
referenced to ptot, used for high dynamic pressure (psf)
dpils – differential pressure of sidewall static pressure in the entrance cone
referenced to ptot, used for low dynamic pressure (psf);
currently, value set to dpihs during engineering units' calculations
kprcoef – dynamic pressure probe coefficient
pa – ambient pressure (psf)
pitot – test section dynamic pressure as measured by pitot tube (psf)
ptot – settling chamber total pressure (psf)
qcode – dynamic pressure computation code; constant
qcstn – dynamic pressure station; constant
rjg - gas constant for air times gravitational constant
ta – entrance cone ambient temperature (degF)
tdew – dew point temperature (degF)
wcode – test section configuration code for classical wall corrections; constant

Output Variables

dpi – measured indicated difference between total and static pressure (psf)
dpinf – dynamic pressure uncorrected for compressibility (psf)
mach – free stream mach number with blockage corrections
machu – uncorrected free stream mach number
mu – absolute viscosity of air corrected for temperature (lbs/sec)
pstat – test section static pressure with wall corrections (psf)
pstatu – uncorrected test section static pressure (psf)
pv – vapor pressure calculated from dew point (psf)
q – tunnel dynamic pressure corrected for compressibility; wall corrections (psf)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
rho – air density corrected with blockage corrections (slugscf)
rhou – uncorrected air density (slugscf)
rnft – test section reynolds number with blockage corrections (1/feet)
rnftu – uncorrected test section reynolds number (1/feet)
tr – ambient temperature (degR)
vel – test section free stream velocity with wall corrections (ft/sec)
velu – uncorrected test section free stream velocity (ft/sec)

** note*   *Reference 14- by 22-Foot Subsonic Tunnel document entitled "Dynamic Pressure*
*   *C alibration Corrections" for the current derivation of cprime.*

** note*   *Cprime(dynamic pressure calibration coefficient) is calculated using equations*
*   *determined by utilizing Check Standard Probe test results.*

*   *Cprime is not written to output data file*

*** default values**

```
dpicut = 20.0
rjg = 1716.2290
kprcoef = 0.998
pitot = 0.0
```

*** tunnel total temperature** (degrees rankine)

```
tr = ta+459.67
```

*** vapor pressure**

```
pv = 12.7654*exp(((9.72334*tdew)-311.147)/
          ((0.555556*tdew)+223.192))
```

*** tunnel air density**

```
rhou = (ptot-pv*0.3789)/(rjg*tr)
```

*** air viscosity**

```
mu = 0.0002672/(tr+198.72)*(tr/518.69)**(1.5)
```

*** QCODE Calculations**

```
if  (qcode > 0) then
  dpi = 0.0
  dpinf = 0.0
  if  (qcode > 1) then
    if  (dpihs > dpicut) then
      dpi = dpihs
    else if  (dpils > 0.0.and.dpils <= dpicut) then
      dpi = dpils
    endif
  endif

  if  (qcode = 1.0) then
```

*** Test Section PITOT Probe**

```
    pitot = amax1(pitot,0.0)
    pstatu = ptot-pitot

  else if  ((qcode = 2.0)  and (qcstn = 1.0)) then
```

*** Closed Test Section, Boundary Layer Removal System Suction is Off**

```
        dpinf = cprime*kprcoef*dpi
        pstatu = ptot-dpinf
```

```
        else if  ((qcode = 3.0) and (qcstn = 1.0)) then
```

*** Closed Test Section, Boundary Layer Removal System Suction is On**

```
                dpinf = cprime*kprcoef*dpi
                pstatu = ptot-dpinf

        else if  ((qcode = 4.0) and (qcstn = 1.0)) then
```

*** Open Test Section, Boundary Layer Removal System Suction is Off**

```
                dpinf = cprime*kprcoef*dpi
                pstatu = ptot-dpinf

        else if  (qcode = 5.0) then
```

*** Open Test Section, Boundary Layer Removal System Suction is On**

```
                dpinf = cprime*kprcoef*dpi
                pstatu = ptot-dpinf

        else if  (qcode = 6.0) then
```

*** Customer Assigned Calibration Constant**

```
                dpinf = ccust*dpi
                pstatu = ptot-dpinf

        else if  (qcode = 11.0) then
```

*** Walls Up, Ceiling Down, Boundary Layer Removal System Suction is Off**
****   This calibration is out of date. Need new calibration for future test(s).***

```
                dpinf = cprime*kprcoef*dpi
                pstatu = ptot-dpinf

        else if  (qcode = 12.0) then
```

*** Walls Up, Ceiling Down, Boundary Layer Removal System Suction is On**
****   This calibration is out of date. Need new calibration for future test(s).***

```
                dpinf = cprime*kprcoef*dpi
                pstatu = ptot-dpinf

        else if  (qcode = 13.0) then
```

*** Walls Up, Ceiling Down, Boundary Layer Removal System Suction is Off,**
***   Vortex Vanes*(Generators)* are On**

```
                  dpinf = cprime*kprcoef*dpi
                  pstatu = ptot-dpinf
```

```
        else

          pstatu = ptot

        endif


* note          compressibility correction
*               gamma = 1.4 (gas constant);  -1.0 / gamma = -0.7143

* density

        if  (pstatu > 0.0) then
          rhou = rhou*(ptot/pstatu)**(-0.7143)
        else
          rhou = 0.0
        endif

* mach number

        if  ((pstatu > 0.0) and ((ptot/pstatu) >= 1.0)) then
          machu = sqrt((((ptot/pstatu)**(2.0/7.0)-1.0)*5.0)
        else
          machu = 0.0
        endif

* dynamic pressure

        if  (wcode = 0.0) then
          xlamd = (ptot-pstatu)/ptot
          qu = 3.5*ptot*((1.0-xlamd)**(5.0/7.0)-1.0+xlamd)
        else
          qu = 0.7*pstatu*machu*machu
        endif
        if  (qu < 0.001) then
          qu = 0.0
        endif

* air velocity

        if  (rhou > 0.0) then
          velu= sqrt(abs(2.0*qu/rhou))
        else
          velu = 0.0
        endif

* reynolds number

        rnftu = rhou*velu/mu

    else
```

*** values for qcode = 0.0**

```
    dpi = 0.0
    dpinf = 0.0
    machu = 0.0
    pstatu = ptot
    qu = 0.0
    rhou = 0.0
    rnftu = 0.0
    velu = 0.0

endif
```

*** initial values for corrected tunnel parameters**

```
mach = machu
pstat = pstatu
q = qu
rho = rhou
rnft = rnftu
vel = velu
```

Flowmeters

Input Variables

flodp# – flowmeter differential pressure (psi)
flopi#– flowmeter inlet static pressure (psi)
flotr# – flowmeter temperature (degR)
ptot – settling chamber total pressure (psf)
tr – ambient temperature (degR)

Output Variables

flbeta# – flowmeter diameter ratio
floasq# - cross-sectional area of throat (sqft)
floc# - discharge coefficient
flodi# – flowmeter inlet diameter (in)
flodt# – flowmeter throat diameter (in)
flof# - velocity of approach factor
flomu# – flowmeter fluid viscosity (lbsec/ft)
flopt# - flowmeter throat static pressure (psi)
floptn# - normalized ambient pressure (psi)
flor# - flowmeter pressure ratio
florn# - flowmeter reynolds number
flotf# - flowmeter temperature (degF)
flottn# - normalized ambient temperature (degR)
floy# - flowmeter expansion factor
flsfmc# - flowmeter super compressibility factor
wtflo# - mass flow rate (lbs/sec)
wtflon# - normalized mass flow rate (lbs/sec)

* *note* *# is flowmeter number*
* *maximum number of flowmeters = 15*
* *the input and output variables aboved are per flowmeter*

* *note* *The source code routine, flowmeters.F, is used to perform the flowmeters' calculations.*

Jet Exhaust Measurements

Input Variables

fm1 – primary flow flowmeter frequency; air system 1 (Hz)
fm2 – primary flow flowmeter frequency; air system 2 (Hz)
fms – secondary flow flowmeter frequency (Hz)
machu – uncorrected free stream mach number
pbl1 – tertiary flow static pressure; probe 1 (psi)
pbl2 – tertiary flow static pressure, probe 2 (psi)
pbl3 – tertiary flow static pressure; probe3 (psi)
pbl4 – tertiary flow static pressure; probe4 (psi)
pch1 – plenum chamber total pressure; engine 1 (psi)
pch2 – plenum chamber total pressure; engine 2 (psi)
pch3 – plenum chamber total pressure; engine 3 (psi)
pch4 – plenum chamber total pressure; engine 4 (psi)
pfm1 – primary flow flowmeter pressure; air system 1 (psi)
pfm2 – primary flow flowmeter pressure; air system 2 (psi)
pfms – secondary flow flowmeter pressure; (psi)
pstatu – uncorrected test section static pressure (psf)
psv – venturi static pressure (psi)
ps1 – primary jet static pressure; engine 1 (psi)
ps2 – primary jet static pressure; engine 2 (psi)
ps3 – primary jet static pressure; engine 3 (psi)
ps4 – primary jet static pressure; engine 4 (psi)
ptbl1 – tertiary flow total pressure; probe 1 (psi)
ptbl2 – tertiary flow total pressure; probe 2 (psi)
ptbl3 – tertiary flow total pressure; probe 3 (psi)
ptbl4 – tertiary flow total pressure; probe 4 (psi)
ptj1.1 – primary jet total pressure; engine 1, probe 1 (psi)
ptj1.2 – primary jet total pressure; engine 1, probe 2 (psi)
ptj1.3 – primary jet total pressure; engine 1, probe 3 (psi)
ptj1.4 – primary jet total pressure; engine 1, probe 4 (psi)
ptj1.5 – primary jet total pressure; engine 1, probe 5 (psi)
ptj1.6 – primary jet total pressure; engine 1, probe 6 (psi)
ptj1.7 – primary jet total pressure; engine 1, probe 7 (psi)
ptj1.8 – primary jet total pressure; engine 1, probe 8 (psi)
ptj1.9 – primary jet total pressure; engine 1, probe 9 (psi)
ptj1.10 – primary jet total pressure; engine 1, probe 10 (psi)
ptj1.11 – primary jet total pressure; engine 1, probe 11 (psi)
ptj1.12 – primary jet total pressure; engine 2, probe 12 (psi)
ptj2.1 – primary jet total pressure; engine 2, probe 1 (psi)
ptj2.2 – primary jet total pressure; engine 2, probe 2 (psi)
ptj2.3 – primary jet total pressure; engine 2, probe 3 (psi)
ptj2.4 – primary jet total pressure; engine 2, probe 4 (psi)
ptj2.5 – primary jet total pressure; engine 2, probe 5 (psi)
ptj2.6 – primary jet total pressure; engine 2, probe 6 (psi)
ptj2.7 – primary jet total pressure; engine 2, probe 7 (psi)
ptj2.8 – primary jet total pressure; engine 2, probe 8 (psi)
ptj2.9 – primary jet total pressure; engine 2, probe 9 (psi)

ptj2.10 – primary jet total pressure; engine 2, probe 10 (psi)
ptj2.11 – primary jet total pressure; engine 2, probe 11 (psi)
ptj2.12 – primary jet total pressure; engine 2, probe 12 (psi)
ptj3.1 – primary jet total pressure; engine 3, probe 1 (psi)
ptj3.2 – primary jet total pressure; engine 3, probe 2 (psi)
ptj3.3 – primary jet total pressure; engine 3, probe 3 (psi)
ptj3.4 – primary jet total pressure; engine 3, probe 4 (psi)
ptj3.5 – primary jet total pressure; engine 3, probe 5 (psi)
ptj3.6 – primary jet total pressure; engine 3, probe 6 (psi)
ptj3.7 – primary jet total pressure; engine 3, probe 7 (psi)
ptj3.8 – primary jet total pressure; engine 3, probe 8 (psi)
ptj3.9 – primary jet total pressure; engine 3, probe 9 (psi)
ptj3.10 – primary jet total pressure; engine 3, probe 10 (psi)
ptj3.11 – primary jet total pressure; engine 3, probe 11 (psi)
ptj3.12 – primary jet total pressure; engine 3, probe 12 (psi)
ptj4.1 – primary jet total pressure; engine 4, probe 1 (psi)
ptj4.2 – primary jet total pressure; engine 4, probe 2 (psi)
ptj4.3 – primary jet total pressure; engine 4, probe 3 (psi)
ptj4.4 – primary jet total pressure; engine 4, probe 4 (psi)
ptj4.5 – primary jet total pressure; engine 4, probe 5 (psi)
ptj4.6 – primary jet total pressure; engine 4, probe 6 (psi)
ptj4.7 – primary jet total pressure; engine 4, probe 7 (psi)
ptj4.8 – primary jet total pressure; engine 4, probe 8 (psi)
ptj4.9 – primary jet total pressure; engine 4, probe 9 (psi)
ptj4.10 – primary jet total pressure; engine 4, probe 10 (psi)
ptj4.11 – primary jet total pressure; engine 4, probe 11 (psi)
ptj4.12 – primary jet total pressure; engine 4, probe 12 (psi)
ptot – settling chamber total pressure (psf)
pts1 – secondary flow total pressure; probe 1 (psi)
pts2 – secondary flow total pressure; probe 2 (psi)
pts3 – secondary flow total pressure; probe 3 (psi)
pts4 – secondary flow total pressure; probe 4 (psi)
ptv – venturi total pressure (psi)
pven1.1 – multiple critical static pressure; air system1, upstream of venturi throat (psi)
pven1.2 – multiple critical static pressure; air system1, downstream of venturi throat (psi)
pven1.3 – multiple critical static pressure; air system1, upstream of venturi throat (psi)
pven1.4 – multiple critical static pressure; air system1, downstream of venturi throat (psi)
pven2.1 – multiple critical static pressure; air system 2, upstream of venturi throat (psi)
pven2.2 – multiple critical static pressure; air system 2, downstream of venturi throat (psi)
pven2.3 – multiple critical static pressure; air system 2, upstream of venturi throat (psi)
pven2.4 – multiple critical static pressure; air system 2, downstream of venturi throat (psi)
pvri1.1 – in-line(not mcv) venturi static pressure; air system 1, venturi 1 (psi)
pvri1.2 – in-line(not mcv) venturi static pressure; air system 1, venturi 2 (psi)

pvri1.3 – in-line(not mcv) venturi static pressure; air system 1, venturi 3 (psi)
pvri1.4 – in-line(not mcv) venturi static pressure; air system 1, venturi 4 (psi)
pvri2.1 – in-line(not mcv) venturi static pressure; air system 2, venturi 1 (psi)
pvri2.2 – in-line(not mcv) venturi static pressure; air system 2, venturi 2 (psi)
pvri2.3 – in-line(not mcv) venturi static pressure; air system 2, venturi 3 (psi)
pvri2.4 – in-line(not mcv) venturi static pressure; air system 2, venturi 4 (psi)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
tch1 – plenum chamber total temperature; engine 1 (degF)
tch2 – plenum chamber total temperature; engine 2 (degF)
tch3 – plenum chamber total temperature; engine 3 (degF)
tch4 – plenum chamber total temperature; engine 4 (degF)
tfm1 – primary flow flowmeter temperature; air system 1 (degF)
tfm2 – primary flow flowmeter temperature; air system 2 (degF)
tfms – secondary flow flowmeter temperature (degF)
ttbl – tertiary flow total temperature (degF)
ttj1.1 – primary jet total temperature; engine 1; probe 1 (degF)
ttj1.2 – primary jet total temperature; engine 1; probe 2 (degF)
ttj1.3 – primary jet total temperature; engine 1; probe 3 (degF)
ttj1.4 – primary jet total temperature; engine 1; probe 4 (degF)
ttj1.5 – primary jet total temperature; engine 1, probe 5 (degF)
ttj1.6 – primary jet total temperature; engine 1, probe 6 (degF)
ttj2.1 – primary jet total temperature, engine 2, probe 1 (degF)
ttj2.2 – primary jet total temperature; engine 2, probe 2 (degF)
ttj2.3 – primary jet total temperature; engine 2, probe 3 (degF)
ttj2.4 – primary jet total temperature; engine 2, probe 4 (degF)
ttj2.5 – primary jet total temperature; engine 2, probe 5 (degF)
ttj2.6 – primary jet total temperature; engine 2, probe 6 (degF)
ttj3.1 – primary jet total temperature; engine 3, probe 1 (degF)
ttj3.2 – primary jet total temperature; engine 3, probe 2 (degF)
ttj3.3 – primary jet total temperature; engine 3, probe 3 (degF)
ttj3.4 – primary jet total temperature; engine 3, probe 4 (degF)
ttj3.5 – primary jet total temperature; engine 3, probe 5 (degF)
ttj3.6 – primary jet total temperature; engine 3, probe 6 (degF)
ttj4.1 – primary jet total temperature; engine 4, probe 1 (degF)
ttj4.2 – primary jet total temperature; engine 4, probe 2 (degF)
ttj4.3 – primary jet total temperature; engine 4, probe 3 (degF)
ttj4.4 – primary jet total temperature; engine 4, probe 4 (degF)
ttj4.5 – primary jet total temperature; engine 4, probe 5 (degF)
ttj4.6 – primary jet total temperature; engine 4, probe 6 (degF)
ttsec – secondary flow total temperature; (degF)
ttv – tertiary flow venturi temperature (degF)
tv1.1 – venturi temperature; air system 1, venturi 1 (degF)
tv1.2 – venturi temperature; air system 1, venturi 2 (degF)
tv2.1 – venturi temperature; air system 2, venturi 1 (degF)
tv2.2 – venturi temperature; air system 2, venturi 2 (degF)
tvri1.1 – in-line(not mcv) venturi temperature; air system 1, venturi 1 (degF)
tvri1.2 – in-line(not mcv) venturi temperature; air system 1, venturi 2 (degF)
tvri1.3 – in-line(not mcv) venturi temperature; air system 1, venturi 3 (degF)
tvri1.4 – in-line(not mcv) venturi temperature; air system 1, venturi 4 (degF)
tvri2.1 – in-line(not mcv) venturi temperature; air system 2, venturi 1 (degF)
tvri2.2 – in-line(not mcv) venturi temperature; air system 2, venturi 2 (degF)

tvri2.3 – in-line(not mcv) venturi temperature; air system 2, venturi 3 (degF)
tvri2.4 – in-line(not mcv) venturi temperature; air system 2, venturi 4 (degF)

Output Variables

anlz1 – total throat area; air system 1
anlz2 – total throat area; air system 2
cfi1 – ideal thrust coefficient based on mass flow rate measured by flowmeter; air system 1
cfi2 – ideal thrust coefficient based on mass flow rate measured by flowmeter; air system 2
cfichr1 – ideal thrust coefficient based on mass flow rate computed from plenum chamber measurements; air system 1
cfichr2 – ideal thrust coefficient based on mass flow rate computed from plenum chamber measurements; air system 2
fi1 – ideal thrust of total primary exhaust system based on mass flow rate measured by flowmeter; air system 1 (lbs)
fi2 – ideal thrust of total primary exhaust system based on mass flow rate measured by flowmeter; air system 2 (lbs)
fia – average ideal thrust of total primary exhaust system (lbs)
fichr1 – ideal thrust of total primary exhaust system based on mass flow rate computed from plenum chamber measurements; air system 1 (lbs)
fichr2 – ideal thrust of total primary exhaust system based on mass flow rate computed from plenum chamber measurements; air system 2 (lbs)
fieng1 – ideal thrust based on mass flow rate computed from individual plenum chamber measurements; engine 1 (lbs)
fieng2 – ideal thrust based on mass flow rate computed from individual plenum chamber measurements; engine 2 (lbs)
fieng3 – ideal thrust based on mass flow rate computed from individual plenum chamber measurements; engine 3 (lbs)
fieng4 – ideal thrust based on mass flow rate computed from individual plenum chamber measurements; engine 4 (lbs)
mbldot – tertiary flow mass flow rate (slugs/sec)
mdot1 – primary flow mass flow rate measured by flowmeter; air system 1 (slugs/sec)
mdot2 – primary flow mass flow rate measured by flowmeter; air system 2 (slugs/sec)
mdota – average primary flow mass flow rate (slugs/sec)
mdotch1 – primary flow mass flow rate computed from plenum chamber measurements; air system 1 (slugs/sec)
mdotch 2 – primary flow mass flow rate computed from plenum chamber measurements; air system 2 (slugs/sec)
mduct1 – mach number; engine 1
mduct2 – mach number; engine 2
mduct3 – mach number; engine 3
mduct4 – mach number; engine 4
msdot – secondary flow mass flow rate (slugs/sec)
pblave – average tertiary flow static pressure (psi)
pchoke – primary choked flow jet total pressure ratio
pjet1 – primary jet pressure; air system 1 (psi)
pjet2 – primary jet pressure; air system 2 (psi)

psec – average secondary flow static pressure (psi)
ptblav – average tertiary flow total pressure (psi)
ptb/pj - ratio of tertiary flow total pressure to primary jet total pressure
ptb/po – ratio of tertiary flow total pressure to free stream total pressure
pteng1 – average primary jet total pressure; engine 1 (psi)
pteng2 – average primary jet total pressure; engine 2 (psi)
pteng3 – average primary jet total pressure; engine 3 (psi)
pteng4 – average primary jet total pressure; engine 4 (psi)
ptengo1 – ratio of average primary jet total pressure in engine 1 to tunnel static pressure
ptengo2 – ratio of average primary jet total pressure in engine 2 to tunnel static pressure
ptengo3 – ratio of average primary jet total pressure in engine 3 to tunnel static pressure
ptengo4 – ratio of average primary jet total pressure in engine 4 to tunnel static pressure
ptj/po1 – primary jet total pressure ratio (all engines); air system 1
ptj/po2 – primary jet total pressure ratio (all engines); air system 2
ptj/poa – average primary jet total pressure ratio (all engines)
ptsec – average secondary flow total pressure (psi)
pts/po – ratio of secondary flow total pressure to free stream total pressure
pts/ptj – ratio of secondary flow total pressure to primary jet total pressure
pva1 – coefficient pv1; air system 1
pva2 – coefficient pv1; air system 2
tas1 – temperature; air system 1
tas2 – temperature; air system 2
thetase – secondary flow corrected mass flow ratio (slugs/sec)
thetbl – tertiary flow corrected mass flow rate (slugs/sec)
tteng1 – average primary jet total temperature; engine 1 (degF)
tteng2 – average primary jet total temperature; engine 2 (degF)
tteng3 – average primary jet total temperature; engine 3 (degF)
tteng4 – average primary jet total temperature; engine 4 (degF)
ttjav – average primary jet total temperature; all engines and air systems (degF)
ttjavg1 – average primary jet total temperature; all engines, air system 1 (degF)
ttjavg2 – average primary jet total temperature; all engines, air system 2 (degF)
tva1 – average venturi temperature; air system 1 (degF)
tva2 – average venturi temperature; air system 2 (degF)
vratio1 – ratio of multiple critical venturi static pressures; air system 1; should be less than 0.93
vratio2 – ratio of multiple critical venturi static pressures; air system 2; should be less than 0.93
wi1 – ideal weight flow rate; air system 1 (lbs/sec)
wi2 – ideal weight flow rate; air system 2 (lbs/sec)
wieng1 – ideal weight flow rate; engine 1 (lbs/sec)
wieng2 – ideal weight flow rate; engine 2 (lbs/sec)
wieng3 – ideal weight flow rate; engine 3 (lbs/sec)
wieng4 – ideal weight flow rate; engine 4 (lbs/sec)
wmcv1 – multiple critical venturi weight flow rate; air system 1 (lbs/sec)
wmcv2 – multiple critical venturi weight flow rate; air system 2 (lbs/sec)

wmcv/wi1 – ratio of multiple critical venturi weight flow rate to ideal weight flow rate; air system 1
wmcv/wi2 – ratio of multiple critical venturi weight flow rate to ideal weight flow rate; air system 2
wp1 – primary flow flowmeter weight flow rate; air system 1 (lbs/sec)
wp2 – primary flow flowmeter weight flow rate; air system 2 (lbs/sec)
wpbl – tertiary flow venturi weight flow rate (lbs/sec)
wpchr1 – total primary flow weight flow rate calculated from plenum chamber measurements; air system 1 (lbs/sec)
wpchr2 – total primary flow weight flow rate calculated from plenum chamber measurements; air system 2 (lbs/sec)
wpch/wi1 – total primary flow discharge coefficient computed from plenum chamber measurements; air system 1
wpch/wi2 – total primary flow discharge coefficient computed from plenum chamber measurements; air system 2
wpeng1 – primary flow weight flow rate computed from plenum chamber measurements; engine 1 (lbs/sec)
wpeng2 – primary flow weight flow rate computed from plenum chamber measurements; engine 2 (lbs/sec)
wpeng3 – primary flow weight flow rate computed from plenum chamber measurements; engine 3 (lbs/sec)
wpeng4 – primary flow weight flow rate computed from plenum chamber measurements; engine 4 (lbs/sec)
wpsec – secondary flow weight flow rate (lbs/sec)
wp/wi1 – primary flow discharge coefficient using flowmeter weight flow rate; air system 1
wp/wi2 – primary flow discharge coefficient using flowmeter weight flow rate; air system 2
wp/wie1 – discharge coefficient computed from plenum chamber measurements; engine 1
wp/wie2 – discharge coefficient computed from plenum chamber measurements; engine 2
wp/wie3 – discharge coefficient computed from plenum chamber measurements; engine 3
wp/wie4 – discharge coefficient computed from plenum chamber measurements; engine 4
wvri1.1 – in-line venturi weight flow rate; air system 1, venturi 1 (lbs/sec)
wvri1.2 – in-line venturi weight flow rate; air system 1, venturi 2 (lbs/sec)
wvri1.3 – in-line venturi weight flow rate; air system 1, venturi 3 (lbs/sec)
wvri1.4 – in-line venturi weight flow rate; air system 1, venturi 4 (lbs/sec)
wvri2.1 – in-line venturi weight flow rate; air system 2, venturi 1 (lbs/sec)
wvri2.2 – in-line venturi weight flow rate; air system 2, venturi 2 (lbs/sec)
wvri3.3 – in-line venturi weight flow rate; air system 2, venturi 3 (lbs/sec)
wvri4.4 – in-line venturi weight flow rate; air system 2, venturi 4 (lbs/sec)

* *note* *The source code routine, modul2.F, is used to perform the jet exhaust*
* *measurements' calculations.*

Refer to pages 14 thru 43 of reference 5 for a detailed explanation of the jet exhaust calculations. Also, refer to reference 1 for information concerning the Multiple Critical Venturi System.

Balance Loads and Model Attitudes

First, calculations are performed for initial loads and second order interactions due to initial loads. Next, corrections for first and second order balance interactions, high interactions/high model restraints, air line pressure, weight tares, method of attachment, and sting deflections are performed. Axis rotations follow these corrections.

Axis rotations include gravity axis to balance axis, gravity axis to model(body) axis, wind axis to gravity axis, and wind axis to model(body) axis. Angle calculations are performed for the balance and model(body) axes.

Next, calculations for the model height, balance components rotated and translated to the model(body) axis, applying estimates of Heyson’s boundary interference, blockage and jet boundary corrections, Heyson’s boundary(wall) interference, and base(cavity) pressures are performed. Finally, balance components and coefficients are calculated for the model, stability, wind, and reference axes.

Although only calculations for the balance one loads are explained, a maximum of four balances may be used. Unless noted, the calculations for balances two thru four are the same as the calculations for balance one. The variable names used in the calculations for each balance are named according to the balance number.

Initial Loads

Input Variables

phi0 – model roll angle at wind-off zero (deg)
theta0 – model pitch angle at wind-off zero (deg)
waf1 – axial force attitude load (lbs)
wnf1 – axial force attitude load (lbs)
wsf1 – side force attitude load (lbs)
wxpm1 – pitching moment attitude load (in-lbs)
wxym1 – yawing moment attitude load (in-lbs)
wyrm1 – rolling moment attitude load (in-lbs)
wyym1 – yawing moment attitude load (in-lbs)
wzpm1 – pitching moment attitude load (in-lbs)
wzrm1 – rolling moment attitude load (in-lbs)

Output Variables

af1.0 – axial force initial load (lbs)
nf1.0 – normal force initial load (lbs)
pm1.0 – pitching moment initial load (in-lbs)
rm1.0 – rolling moment initial load (in-lbs)
sf1.0 – side force initial load (lbs)
ym1.0 – yawing moment initial load (in-lbs)

** note   dtor = 0.0174532925199 {converts degrees to radians}*

```
af1.0 = waf1 * sin(theta0*dtor)
sf1.0 = wsf1 * sin(phi0*dtor) * cos(theta0*dtor)
nf1.0 = -wnf1 * cos(phi0*dtor) * cos(theta0*dtor)
rm1.0 = (wyrm1 * cos(phi0*dtor) * cos(theta0*dtor)) +
        (wzrm1 * sin(phi0*dtor) * cos(theta0*dtor))
pm1.0 = (-wxpm1 * cos(phi0*dtor) * cos(theta0*dtor)) +
        (wzpm1 * sin(theta0*dtor))
ym1.0 = (wxym1 * sin(phi0*dtor) * cos(theta0*dtor)) +
        (wyym1 * sin(theta0*dtor))
```

Second Order Interactions Due To Initial Loads

Input Variables

af1.0 – axial force initial load (lbs)
nf1.0 – normal force initial load (lbs)
pm1.0 – pitching moment initial load (in-lbs)
rm1.0 – rolling moment initial load (in-lbs)
sf1.0 – side force initial load (lbs)
ym1.0 – yawing moment initial load (in-lbs)

Output Variables

afez1 - axial force second order interactions due to initial loads (lbs)
nfez1- normal force second order interactions due to initial loads (lbs)
pmez1- pitching moment second order interactions due to initial loads (in-lbs)
rmez1- rolling moment second order interactions due to initial loads (in-lbs)
sfez1- side force second order interactions due to initial loads (lbs)
ymez1- yawing moment second order interactions due to initial loads (in-lbs)

** note*	*execute subfunction **ctrnl** which computes the second order interactions*
***	*due to initial loads.*

** note*	*afez1, sfez1, nfez1, rmez1, pmez1, ymez1 are not written to output data file*

First & Second Order Balance Interactions

Input Variables

afez1 – axial force second order interactions due to initial loads(lbs)
af1 – uncorrected axial force (lbs)
af1.0 – axial force initial load (lbs)
nfez1– normal force second order interactions due to initial loads(lbs)
nf1 – uncorrected normal force (lbs)
nf1.0 – normal force initial load (lbs)
pmez1– pitching moment second order interactions due to initial loads(in-lbs)
pm1 – uncorrected pitching moment (in-lbs)
pm1.0 – pitching moment initial load (in-lbs)
rmez1– rolling moment second order interactions due to initial loads(in-lbs)
rm1 – uncorrected rolling moment (in-lbs)
rm1.0 – rolling moment initial load (in-lbs)
sfez1– side force second order interactions due to initial loads(lbs)
sf1 – uncorrected side force (lbs)
sf1.0 – side force initial load (lbs)
ymez1– yawing moment second order interactions due to initial loads(in-lbs)
ym1 – uncorrected yawing moment (in-lbs)
ym1.0 – yawing moment initial load (in-lbs)

Output Variables

af1.2 – axial force load corrected for 1^{st} & 2^{nd} order balance interactions (lbs)
nf1.2 – normal force load corrected for 1^{st} & 2^{nd} order balance interactions (lbs)
pm1.2 – pitching moment load corrected for 1^{st} & 2^{nd} order balance interactions (in-lbs)
rm1.2 – rolling moment load corrected for 1^{st} & 2^{nd} order balance interactions (in-lbs)
sf1.2 – side force load corrected for 1^{st} & 2^{nd} order balance interactions (lbs)
ym1.2 – yawing moment load corrected for 1^{st} & 2^{nd} order balance interactions (in-lbs)

** note* *execute subfunction **cintr** which computes corrected delta loads for first*
*** *and second order balance interactions.*

High Interactions and High Model Restraints

The six balance loads are corrected for high interactions and high model restraints. A 6x6 balance interaction matrix is used for the interaction factors.

Input Variables

af1.2 – axial force load corrected for 1st & 2nd order balance interactions (lbs)
b1aa1 – axial force(axial) interaction factor; constant; default value is 1.0
b1na1 – axial force(normal) interaction factor; constant; default value is 0.0
b1pa1 – axial force(pitch) interaction factor; constant; default value is 0.0
b1ra1 – axial force(roll) interaction factor; constant; default value is 0.0
b1sa1 – axial force(side) interaction factor; constant; default value is 0.0
b1ya1 – axial force(yaw) interaction factor; constant; default value is 0.0
b1an1 – normal force(axial) interaction factor; constant; default value is 0.0
b1nn1 – normal force(normal) interaction factor; constant; default value is 1.0
b1pn1 – normal force(pitch) interaction factor; constant; default value is 0.0
b1rn1 – normal force(roll) interaction factor; constant; default value is 0.0
b1sn1 – normal force(side) interaction factor; constant; default value is 0.0
b1yn1 – normal force(yaw) interaction factor; constant; default value is 0.0
b1ap1 – pitching moment(axial) interaction factor; constant; default value is 0.0
b1np1 – pitching moment(normal) interaction factor; constant; default value is 0.0
b1pp1 – pitching moment(pitch) interaction factor; constant; default value is 1.0
b1rp1 – pitching moment(roll) interaction factor; constant; default value is 0.0
b1sp1 – pitching moment(side) interaction factor; constant; default value is 0.0
b1yp1 – pitching moment(yaw) interaction factor; constant; default value is 0.0
b1ar1 – rolling moment(axial) interaction factor; constant; default value is 0.0
b1nr1 – rolling moment(normal) interaction factor; constant; default value is 0.0
b1pr1 – rolling moment(pitch) interaction factor; constant; default value is 0.0
b1rr1 – rolling moment(roll) interaction factor; constant; default value is 1.0
b1sr1 – rolling moment(side) interaction factor; constant; default value is 0.0
b1yr1 – rolling moment(yaw) interaction factor; constant; default value is 0.0
b1as1 – side force(axial) interaction factor; constant; default value is 0.0
b1ns1 – side force(normal) interaction factor; constant; default value is 0.0
b1ps1 – side force(pitch) interaction factor; constant; default value is 0.0
b1rs1 – side force(roll) interaction factor; constant; default value is 0.0
b1ss1 – side force(side) interaction factor; constant; default value is 1.0
b1ys1 – side force(yaw) interaction factor; constant; default value is 0.0
b1ay1 – yawing moment(axial) interaction factor; constant; default value is 0.0
b1ny1 – yawing moment(normal) interaction factor; constant; default value is 0.0
b1py1 – yawing moment(pitch) interaction factor; constant; default value is 0.0
b1ry1 – yawing moment(roll) interaction factor; constant; default value is 0.0
b1sy1 – yawing moment(side) interaction factor; constant; default value is 0.0
b1yy1 – yawing moment(yaw) interaction factor; constant; default value is 1.0
nf1.2 – normal force load corrected for 1st & 2nd order balance interactions (lbs)
pm1.2 – pitching moment load corrected for 1st & 2nd order balance interactions (in-lbs)
rm1.2 – rolling moment load corrected for 1st & 2nd order balance interactions (in-lbs)
sf1.2 – side force load corrected for 1st & 2nd order balance interactions (lbs)
ym1.2 – yawing moment load corrected for 1st & 2nd order balance interactions (in-lbs)

Output Variables

af1.3 – axial force load corrected for high interactions and high model retraints (lbs)
nf1.3 – normal force load corrected for high interactions and high model restraints (lbs)
pm1.3 – pitching moment load corrected for high interactions and high model restraints (in-lbs)
rm1.3 – rolling moment load corrected for high interactions and high model restraints (in-lbs)
sf1.3 – side force load corrected for high interactions and high model restraints (lbs)
ym1.3 – yawing moment load corrected for high interactions and high model restraints (in-lbs)

af1.3 = (af1.2*b1aa1) + (sf1.2*b1sa1) + (nf1.2*b1na1) +(rm1.2*b1ra1) + (pm1.2*b1pa1)
+ (ym1.2*b1ya1)

sf1.3 = (af1.2*b1as1) + (sf1.2*b1ss1) + (nf1.2*b1ns1) + (rm1.2*b1rs1) + (pm1.2*b1ps1)
+ (ym1.2*b1ys1)

nf1.3 = (af1.2*b1an1) + (sf1.2*b1sn1) + (nf1.2*b1nn1) + (rm1.2*b1rn1) + (pm1.2*b1pn1)
+ (ym1.2*b1yn1)

rm1.3 = (af1.2*b1ar1) + (sf1.2*b1sr1) + (nf1.2*b1nr1) + (rm1.2*b1rr1) + (pm1.2*b1pr1)
+ (ym1.2*b1yr1)

pm1.3 = (af1.2*b1ap1) + (sf1.2*b1sp1) + (nf1.2*b1np1) + (rm1.2*b1rp1) + (pm1.2*b1pp1)
+ (ym1.2*b1yp1)

ym1.3 = (af1.2*b1ay1) + (sf1.2*b1sy1) + (nf1.2*b1ny1) + (rm1.2*b1ry1) + (pm1.2*b1py1)
+ (ym1.2*b1yy1)

Air Line Pressure Corrections

The six balance loads are corrected for air line pressure.

Input Variables

af1.3 – axial force load corrected for high interactions and high model retraints (lbs)
nf1.3 – normal force load corrected for high interactions and high model restraints (lbs)
pca1.1 – axial force air line pressure coefficient 1; constant; default value is 0.0
pca2.1 – axial force air line pressure coefficient 2; constant; default value is 0.0
pca3.1 – axial force air line pressure coefficient 3; constant; default value is 0.0
pcn1.1 – normal force air line pressure coefficient 1; constant; default value is 0.0
pcn2.1 – normal force air line pressure coefficient 2; constant; default value is 0.0
pcn3.1 – normal force air line pressure coefficient 3; constant; default value is 0.0
pcp1.1 – pitching moment air line pressure coefficient 1; constant; default value is 0.0
pcp2.1 – pitching moment air line pressure coefficient 2; constant; default value is 0.0
pcp3.1 – pitching moment air line pressure coefficient 3; constant; default value is 0.0
pcr1.1 – rolling moment air line pressure coefficient 1; constant; default value is 0.0
pcr2.1 – rolling moment air line pressure coefficient 2; constant; default value is 0.0
pcr3.1 – rolling moment air line pressure coefficient 3; constant; default value is 0.0
pcs1.1 – side force air line pressure coefficient 1; constant; default value is 0.0
pcs2.1 – side force air line pressure coefficient 2; constant; default value is 0.0
pcs3.1 – side force air line pressure coefficient 3; constant; default value is 0.0
pcy1.1 – yawing moment air line pressure coefficient 1; constant; default value is 0.0
pcy2.1 – yawing moment air line pressure coefficient 2; constant; default value is 0.0
pcy3.1 – yawing moment air line pressure coefficient 3; constant; default value is 0.0
pm1.3 – pitching moment load corrected for high interactions and high model restraints (in-lbs)
pst – air line pressure (psi)
rm1.3 – rolling moment load corrected for high interactions and high model restraints (in-lbs)
sf1.3 – side force load corrected for high interactions and high model restraints (lbs)
ym1.3 – yawing moment load corrected for high interactions and high model restraints (in-lbs)

Output Variables

af1.3 – axial force load corrected for high interactions, high model retraints and air line pressure (lbs)
nf1.3 – normal force load corrected for high interactions, high model restraints and air line pressure (lbs)
pm1.3 – pitching moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
rm1.3 – rolling moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
sf1.3 – side force load corrected for high interactions, high model restraints and air line pressure (lbs)
ym1.3 – yawing moment load corrected for high interactions, high model restraints, and air line presssure (in-lbs)

af1.3 = af1.3 – ((pst*pca1.1) + (pst*pst*pca2.1) + (pst*pst*pst*pca3.1))

sf1.3 = sf1.3 – ((pst*pcs1.1) + (pst*pst*pcs2.1) + (pst*pst*pst*pcs3.1))

nf1.3 = nf1.3 – ((pst*pcn1.1) + (pst*pst*pcn2.1) +(pst*pst*pst*pcn3.1))

rm1.3 = rm1.3 – ((pst*pcr1.1) + (pst*pst*pcr2.1) +(pst*pst*pst*pcr3.1))

pm1.3 = pm1.3 – ((pst*pcp1.1) + (pst*pst*pcp2.1) +(pst*pst*pst*pcp3.1))

ym1.3 = ym1.3 – ((pst*pcy1.1) + (pst*pst*pcy2.1) +(pst*pst*pst*pcy3.1))

Sting Deflections

Input Variables

af1.0 - axial force initial load (lbs)
af1.3 – axial force load corrected for high interactions, high model retraints and air line pressure (lbs)
kdfl – sting deflection corrections flag; constant
nf1.0 - normal force initial load (lbs)
nf1.3 – normal force load corrected for high interactions, high model restraints and air line pressure (lbs)
phida1 – axial force roll deflection; constant
phidn1 – normal force roll deflection; constant
phidp1 – pitching moment roll deflection; constant
phidr1 – rolling moment roll deflection; constant
phids1 – side force roll deflection; constant
phidy1 – yawing moment roll deflection; constant
pm1.0 - pitching moment initial load (in-lbs)
pm1.3 – pitching moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
psida1 –axial force yaw deflection; constant
psidn1 – normal force yaw deflection; constant
psidp1 – pitching moment yaw deflection; constant
psidr1 – rolling moment yaw deflection; constant
psids1 – side force yaw deflection; constant
psidy1 – yawing moment yaw deflection; constant
rm1.0 - rolling moment intial load (in-lbs)
rm1.3 – rolling moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
sf1.0 - side force initial load (lbs)
sf1.3 – side force load corrected for high interactions, high model restraints and air line pressure (lbs)
theda1 – axial force pitch deflection; constant
thedn1 – normal force pitch deflection; constant
thedp1 – pitching moment pitch deflection; constant
thedr1 – rolling moment pitch deflection; constant
theds1 – side force pitch deflection; constant
thedy1 – yawing moment pitch deflection; constant
ym1.0 - yawing moment initial load (in-lbs)
ym1.3 – yawing moment load corrected for high interactions, high model restraints, and air line presssure (in-lbs)

Output Variables

phid – roll deflection angle (deg)
psid – yaw deflection angle (deg)
thetad – pitch deflection angle (deg)

```
* note          The default value is 0.0 for phida1, phids1, phidn1, phidr1, phidp1,
*               phidy1, theda1, theds1, thedn1, thedr1, thedp1, thedy1, psida1, psids1,
*               psidn1, psidr1, psidp1, psidy1

      if  ( kdfl = -1 ) then

*no sting deflection corrections

        phid = 0.0
        thetad = 0.0
        psid = 0.0

      else if  ( kdfl = 0 ) then

*sting deflection corrections using total loads

        phid = (phida1*(af1.0+af1.3)) + (phids1*(sf1.0+sf1.3)) +
               (phidn1*(nf1.0+nf1.3)) + (phidr1*(rm1.0+rm1.3)) +
               (phidp1*(pm1.0+pm1.3)) +  (phidy1*(ym1.0+ym1.3))

        thetad = (theda1*(af1.0+af1.3)) + (theds1*(sf1.0+sf1.3)) +
                 (thedn1*(nf1.0+nf1.3)) + (thedr1*(rm1.0+rm1.3)) +
                 (thedp1*(pm1.0+pm1.3)) +  (thedy1*(ym1.0+ym1.3))

        psid = (psida1*(af1.0+af1.3)) + (psids1*(sf1.0+sf1.3)) +
               (psidn1*(nf1.0+nf1.3)) + (psidr1*(rm1.0+rm1.3)) +
               (psidp1*(pm1.0+pm1.3)) +  (psidy1*(ym1.0+ym1.3))

      else if  ( kdfl = 1 ) then

*sting deflection corrections using delta loads

        phid = (phida1*af1.3) + (phids1*sf1.3) + (phidn1*nf1.3) +
               (phidr1*rm1.3) + (phidp1*pm1.3) + (phidy1*ym1.3)

        thetad = (theda1*af1.3) + (theds1*sf1.3) + thedn1*nf1.3) +
                 (thedr1*rm1.3) + (thedp1*pm1.3) +  (thedy1*ym1.3)

        psid = (psida1*af1.3) + (psids1*sf1.3) + (psidn1*nf1.3) +
               (psidr1*rm1.3) + (psidp1*pm1.3) +  (psidy1*ym1.3)

      endif
```

Weight Tares

Input Variables

af1.3 – axial force load corrected for high interactions, high model restraints and air line pressure (lbs)
nf1.3 – normal force load corrected for high interactions, high model restraints and air line pressure (lbs)
phi – model roll angle (deg)
phi0 – model roll angle at wind-off zero (deg)
pm1.3 – pitching moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
rm1.3 – rolling moment load corrected for high interactions, high model restraints, and air line pressure (in-lbs)
sf1.3 – side force load corrected for high interactions, high model restraints and air line pressure (lbs)
theta – model pitch angle (deg)
theta0 – model pitch angle at wind-off zero (deg)
waf1 – axial force attitude load (lbs)
wnf1 – axial force attitude load (lbs)
wsf1 – side force attitude load (lbs)
wxpm1 – pitching moment attitude load (in-lbs)
wxym1 – yawing moment attitude load (in-lbs)
wyrm1 – rolling moment attitude load (in-lbs)
wyym1 – yawing moment attitude load (in-lbs)
wzpm1 – pitching moment attitude load (in-lbs)
wzrm1 – rolling moment attitude load (in-lbs)
ym1.3 – yawing moment load corrected for high interactions, high model restraints, and air line presssure (in-lbs)

Output Variables

aftare – axial force tare correction (lbs)
af1.4 – axial force load corrected for weight tares (lbs)
nftare – normal force tare correction (lbs)
nf1.4 – normal force load corrected for weight tares (lbs)
pmtare – pitching moment tare correction (in-lbs)
pm1.4 – pitching moment load corrected for weight tares (in-lbs)
rmtare – rolling moment tare correction (in-lbs)
rm1.4 – rolling moment load corrected for weight tares (in-lbs)
sftare – side force tare correction (lbs)
sf1.4 – side force load corrected for weight tares (lbs)
ymtare – yawing moment tare correction (in-lbs)
ym1.4 – yawing moment load corrected for weight tares (in-lbs)

** note   dtor = 0.0174532925199 {converts degrees to radians}*

```
aftare = waf1 * (sin(theta*dtor) - sin(theta0*dtor))
sftare = wsf1 * (cos(theta*dtor) * sin(phi*dtor)) –
                  (cos(theta0*dtor) * sin(phi0*dtor))
nftare = -wnf1* (cos(theta*dtor) * cos(phi*dtor)) –
                  (cos(theta0*dtor) * cos(phi0*dtor))

if (waf1 = 0.0 & wsf1 = 0.0 & wnf1 = 0.0) then
  rmtare = 0.0
  pmtare = 0.0
  ymtare = 0.0
else
  rmtare = ((wzrm1/wsf1)*sftare) – ((wyrm1/wnf1)*nftare)
  pmtare = ((wzpm1/waf1)*aftare) + ((wxpm1/wnf1)*nftare)
  ymtare = ((wxym1/wsf1)*sftare) + ((wyym1/waf1)*aftare)
endif

af1.4 = af1.3 – aftare
sf1.4 = sf1.3 – sftare
nf1.4 = nf1.3 – nftare
rm1.4 = rmf1.3 – rmtare
pm1.4 = pmf1.3 – pmtare
ym1.4 = ym1.3 – ymtare
```

Method of Attachment

Input Variables

af1.4 – axial force load corrected for weight tares (lbs)
ksign1 – balance attachment; constant (1 = normal balance attachment,
-1 = grounding balance by opposite end)
nf1.4 – normal force load corrected for weight tares (lbs)
pm1.4 – pitching moment load corrected for weight tares (in-lbs)
rm1.4 – rolling moment load corrected for weight tares (in-lbs)
sf1.4 – side force load corrected for weight tares (lbs)
ym1.4 – yawing moment load corrected for weight tares (in-lbs)

Output Variables

af1.5 – axial force load corrected for method of attachment (lbs)
nf1.5 – normal force load corrected for method of attachment (lbs)
pm1.5 – pitching moment load corrected for method of attachment (in-lbs)
rm1.5 – rolling moment load corrected for method of attachment (in-lbs)
sf1.5 – side force load corrected for method of attachment (lbs)
ym1.5 – yawing moment load corrected for method of attachment (in-lbs)

af1.5 = af1.4 * ksign1
sf1.5 = sf1.4 * ksign1
nf1.5 = nf1.4 * ksign1
rm1.5 = rm1.4 * ksign1
pm1.5 = pm1.4 * ksign1
ym1.5 = ym1.4 * ksign1

Angles' Calculations

Subfunction ***euler*** makes use of the angles' rotation flags and the 3 by 3 identity, yaw, pitch, and roll matrices to rotate(transform) angles from one axis to another axis. ***Euler*** is used to calculate the rotation matrices for the **gravity axis to balance axis**, **gravity axis to model(body) axis**, and **wind axis to gravity axis rotations**. These matrices are used to generate the angles' calculations.

Identity matrix(3x3)

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

Yaw matrix(3x3)

cos(angle)	-sin(angle)	0.0
sin(angle)	cos(angle)	0.0
0.0	0.0	1.0

Pitch matrix(3x3)

cos(angle)	0.0	-sin(angle)
0.0	1.0	0.0
sin(angle)	0.0	cos(angle)

Roll matrix(3x3)

1.0	0.0	0.0
0.0	cos(angle)	-sin(angle)
0.0	sin(angle)	cos(angle)

Rotation of Angles from Gravity Axis to Balance Axis

Input Variables

gbangles – nine-element array of angles (Euler input)
gbflg – nine-element array of rotation flags (Euler input); one flag for each angle
    where values are [-1 = yaw rotation, 0 = pitch rotation, 1 = roll rotation]
idmat – 3x3 identity array (Euler input)
phid – deflection roll angle (deg)
phik – knuckle roll angle (deg)
phis – strut roll angle (deg)
psid – deflection yaw angle (deg)
psik – knuckle yaw angle (deg)
psis – strut yaw angle (deg)
thetad – deflection pitch angle (deg)
thetak – knuckle pitch angle (deg)
thetas – strut pitch angle (deg)

Output Variables

gbmat – 3x3 gravity to balance axis array **(**Euler output)

* *note* *idmat and gbmat are double precision*

* *note* *idmat, gbangles, gbflg, gbmat are not written to output data file*

For a yaw-pitch-roll rotation, Let

gbangles(1) = psis
gbangles(2) = thetas
gbangles(3) = phis
gbangles(4) = psik
gbangles(5) = thetak
gbangles(6) = phik
gbangles(7) = psid
gbangles(8) = thetad
gbangles(9) = phid
gbflg(1) = -1
gbflg(2) = 0
gbflg(3) = 1
gbflg(4) = -1
gbflg(5) = 0
gbflg(6) = 1
gbflg(7) = -1
gbflg(8) = 0
gbflg(9) = 1

** note*	*Input angles are in array **gbangles**, input matrix is **idmat**, and*
***	*rotation flags are in array **gbflg***

note* *Execute subfunction **euler which performs angle rotation(s)*

** note* *Output matrix is **gbmat***

** note* *In special cases, up to **twelve knuckle angles** may be used;*
*** *the **other nine knuckle angles** are*
***
*** ***psij, thetaj, phij***
***
*** ***psip, thetap, phip***
***
*** ***psiv, thetav, phiv***

Rotation of Angles from Gravity Axis to Model(Body) Axis

Input Variables

gbmat – 3x3 gravity to balance axis array (Euler input)
gmangles – three-element array of angles (Euler input)
gmflg – three-element array of rotation flags(Euler input); one flag for each angle
    where values are [-1 = yaw rotation , 0 = pitch rotation, 1 = roll rotation]
phib – model(body) roll angle (deg)
psib – model(body) yaw angle (deg)
thetab – model(body) pitch angle (deg)

Output Variables

gmmat – 3x3 gravity to model(body) axis array (Euler output)

** note*   *gbmat and gmmat are double precision*

** note*   *gmangles, gmflg, gbmat, and gmmat are not written to output data file*

For a yaw-pitch-roll rotation, Let

gmangles(1) = psib
gmangles(2) = thetab
gmangles(3) = phib
gmflg(1) = -1
gmflg(2) = 0
gmflg(3) = 1

** note*   *Input angles are in array **gmangles**, input matrix is **gbmat**, and*
***   *rotation flags are in array **gmflg***

note*   *Execute subfunction **euler which performs angle rotation(s)*

** note*   *Output matrix is **gmmat***

Rotation of Angles from Wind Axis to Gravity Axis

Input Variables

idmat - 3x3 identity array (Euler input)
psiu – sideflow angle (deg)
thetau – upflow angle (deg)
wgangles - two-element array of angles (Euler input)
wgflg - two-element array of rotation flags (Euler input); one flag for each angle
where values are [0 = pitch rotation, -1 = yaw rotation]

Output Variables

wgmat - 3x3 wind to gravity axis array (Euler output)

** note idmat and wgmat are double precision*

** note idmat, wgflg, wgmat are not written to output data file*

For a pitch-yaw rotation, Let

wgangles(1) = thetau
wgangles(2) = psiu
wgflg(1) = 0
wgflg(2) = -1

** note Input angles are in array **wgangles**, input matrix is **idmat**, and*
** rotation flags are in array **wgflg***

note Execute subfunction **euler which performs angle rotation(s)*

** note Output matrix is **wgmat***

Rotation of Angles from Wind Axis to Model(Body) Axis

Input Variables

gmmat - 3x3 gravity to model(body) axis array
wgmat - 3x3 wind to gravity axis array

Output Variables

wmmat - a 3x3 wind to model(body) axis array

** note gmmat, wgmat, and wmmat are double precision*

** note gmmat, wgmat, and wmmat are not written to output data file*

Let

	gm11	gm12	gm13	
gmmat =	gm21	gm22	gm23	gravity *to model(body) axis matrix*
	gm31	gm32	gm33	

and

	wg11	wg12	wg13	
wgmat =	wg21	wg22	wg23	*wind to gravity axis matrix*
	wg31	wg32	wg33	

then

	wm11	wm12	wm13	
wmmat =	wm21	wm22	wm23	*wind to model(body) axis matrix*
	wm31	wm32	wm33	

```
do k=1,3
   do j=1,3
      wm(j,k) = 0.0
      do i=1,3
         wm(j,k) = wm(j,k)+(gm(j,i)*wg(i,k))
      enddo
   enddo
enddo
```

Balance Axis Angles

Input Variables

gbmat **-** 3x3 gravity to balance axis array
id – data identification flag

Output Variables

phi – balance roll angle[-360 to +360 degrees] (deg)
phi0 – balance roll angle[-360 to +360 degrees] at wind-off zero (deg)
psi – balance yaw angle (deg)
psi0 – balance yaw angle at wind-off zero **(**deg)
theta – balance pitch angle **(**deg)
theta0 – balance pitch angle at wind-off zero **(**deg)

* *note* *rtod = 57.2957795131 {converts radians to degrees}*

* *note* *gbmat is double precision*

* *note* *gbmat, psi, and psi0 are not written to output data file*

* *note* *datan2(value1, value2) = datan(value1 / value2)*
* *where datan2 is double precision arc tangent of two arguments*

*** gravity to balance axis angles**

Let

$$gbmat = \begin{matrix} gb11 & gb12 & gb13 \\ gb21 & gb22 & gb23 \\ gb31 & gb32 & gb33 \end{matrix} \quad \text{(gravity to balance axis matrix)}$$

then

```
if ( gb12 = 0.0 & gb11 = 0.0 ) then
  psi = 0.0
else
  psi = datan2(-gb12, gb11) * rtod
endif
theta = dasin(-gb13) * rtod
if ( gb23 = 0.0 & gb33 = 0.0 ) then
  phi = 0.0
else
  phi = datan2(-gb23, gb33) * rtod
endif
```

** note woz is wind-off zero flag number*

```
 if  ( id = woz ) then
     psi0 = psi
     theta0 = theta
     phi0 = phi
endif
```

Model(Body) Axis Angles

Input Variables

gmmat – 3x3 gravity to model(body) axis array
wmmat - 3x3 wind to model(body)axis array

Output Variables

alfunc – uncorrected model(body) angle of attack (deg)
beta – model(body) sideslip angle (deg)
modrol – model(body) roll angle (deg) [-180 to +180 degrees;
used for airplane models]
pitchgm1 – model(body) pitch angle
rollgm1 – model(body) roll angle
yawgm1 – model(body) yaw angle

* *note* *rtod = 57.2957795131 {converts radians to degrees}*

* *note* *gmmat and wmmat are double precision*

* *note* *gmmat, wmmat, yawgm1, pitchgm1, and rollgm1*
* *are not written to output data file*

* *note* *datan2(value1, value2) = datan(value1 / value2)*
* *where datan2 is double precision arc tangent of two arguments*

*** gravity to model(body) axis angles**

Let

```
                  gm11 gm12 gm13
    gmmat =       gm21 gm22 gm23     gravity to model(body) axis matrix
                  gm31 gm32 gm33
```

then

```
if ( gm12 = 0.0 & gm11 = 0.0 ) then
  yawgm1 = 0.0
else
  yawgm1 = datan2(-gm12, gm11) * rtod
endif
pitchgm1 =  dasin(-gm13) * rtod

if ( gm23 = 0.0  & gm33 = 0.0 ) then
  rollgm1 = 0.0
else
  rollgm1 = datan2(-gm23, gm33) * rtod
endif
```

*** wind to model(body) axis angles**

Let

```
                      wm11 wm12 wm13
     wmmat =          wm21 wm22 wm23     wind to model(body) axis matrix
                      wm31 wm32 wm33
```

then

```
    beta =  -dasin(wm21) * rtod

    if ( wm31 = 0.0 & wm11 = 0.0 ) then
      alfunc = 0.0
    else
      alfunc  = datan2(wm31, wm11) * rtod
    endif

    if ( wm23 = 0.0  & wm22 = 0.0 ) then
      modrol = 0.0
    else
      modrol = datan2(-wm23, wm22) * rtod
    endif
```

Model Height

Input Variables

elev – cart elevation plus cart offset (inches)
pitchm – mast pitch angle (deg)
rlength – distance from (1) the center of the vertical mast to the model height reference point or (2) the center of the pitch rotation pivot to the model height reference point; constant (inches)
scode – sting code; constant
thetas – strut pitch angle (deg)
vlength – vertical distance from the pivot center on the vertical strut to the model reference point; constant (inches)

Output Variables

hgt – vertical distance from the test section floor to model height reference point (inches)

** note dtor = 0.0174532925199 {converts degrees to radians}*

```
if  ( scode = 0.0 ) then
```

*** no height calculation**

```
  hgt = 0.0

else if  (( scode = 1.0 ) or ( scode = 2.0 )) then
```

*** height calculation for mast-mounted models or for alpha-beta sting**

```
  hgt = 87.0+(elev-87.0)*cos(pitchm*dtor)+rlength*sin(pitchm*dtor)

else if  ( scode = 3.0 ) then
```

*** height calculation for vertical strut**

```
  hgt = elev+rlength*sin(thetas*dtor)+vlength*cos(thetas*dtor)

endif
```

Balance Components Rotated and Translated to Model Axis

Input Variables

af1.5 – axial force load corrected for method of attachment (lbs)
bcmangles - three-element array of angles (Euler input)
bcmflg - three-element array of rotation flags (Euler input); one flag for each angle where values are [-1 = yaw rotation, 0 = pitch rotation, 1 = roll rotation]
bcmin - six-element array (Euler input)
bspan1 – wing span; constant (ft)
chord1– wing aerodynamic chord; constant (ft)
nf1.5 – normal force load corrected for method of attachment (lbs)
phib – model(body) roll angle (deg)
pm1.5 – pitching moment load corrected for method of attachment (in-lbs)
psib – model(body) yaw angle (deg)
q – tunnel dynamic pressure corrected for compressibility; wall corrections (psf)
rm1.5 – rolling moment load corrected for method of attachment (in-lbs)
sarea1 – wing area; constant (sqft)
sf1.5 – side force load corrected for method of attachment (lbs)
thetab – model(body) pitch angle (deg)
xbar1 – moment transfer distance(x direction) measured in the model force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)
ybar1 – moment transfer distance(y direction) measured in the model force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)
ym1.5 – yawing moment load corrected for method of attachment (in-lbs)
zbar1 – moment transfer distance(z direction) measured in the body force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)

Output Variables

bcmout - six-element array (Euler output)
fa1.1 - axial force rotated to the model axis (lbs)
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1.1 - normal force rotated to the model axis (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
fy1.1 - side force rotated to the model axis (lbs)
fy1.2 - side force rotated and translated to the model axis (lbs)
mx1.1 - rolling moment rotated to the model axis (in-lbs)
mx1.2 - rolling moment rotated and translated to the model axis (in-lbs)
my1.1 - pitching moment rotated to the model axis (in-lbs)
my1.2 - pitching moment rotated and translated to the model axis (in-lbs)
mz1.1 - yawing moment rotated to the model axis (in-lbs)
mz1.2 yawing moment rotated and translated to the model axis (in-lbs)

*** forces and moments rotated to the model axis**

For a yaw-pitch-roll rotation, Let

bcmangles(1) = psib
bcmangles(2) = thetab
bcmangles(3) = phib
bcmflg(1) = -1
bcmflg(2) = 0
bcmflg(3) = 1

and

bcmin(1) = af1.5
bcmin(2) = sf1.5
bcmin(3) = nf1.5
bcmin(4) = -rm1.5
bcmin(5) = pm1.5
bcmin(6) = -ym1.5

** note execute subfunction* ***euler*** *which performs angle rotation(s)*

fa1.1 = bcmout(1)
fy1.1 = bcmout(2)
fn1.1 = bcmout(3)
mx1.1 = -bcmout(4)
my1.1 = bcmout(5)
mz1.1 = -bcmout(6)

*** forces and moments rotated and translated to the model axis**

fa1.2 = fa1.1
fy1.2 = fy1.1
fn1.2 = fn1.1
mx1.2 = mx1.1 + ((fn1.1 * ybar1) - (fy1.1 * zbar1))
my1.2 = my1.1 – ((fn1.1 * xbar1) - (fa1.1 * zbar1))
mz1.2 = mz1.1 – ((fy1.1 * xbar1) - (fa1.1 * ybar1))

Apply Estimates of Heyson's Corrections - Walls Up

Input Variables

alfunc – uncorrected model(body) angle of attack (deg)
dalup#.# - array of alpha correction coefficients; constants
dolup# - array of drag over lift coefficients; constants
dqup#.# - array of q correction coefficients; constants
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
nbal – balance number; constant
pa – ambient pressure (psf)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
sarea1 – wing area; constant (sqft)
velu – uncorrected test section free stream velocity (ft/sec)
wcode – test section configuration code for classical wall corrections; constant

Output Variables

alpha – model(body) angle of attack with walls up or down corrections (deg)
pstat – test section static pressure with walls up or down corrections (ft/sec)
q – tunnel dynamic pressure corrected for compressibility; walls up or down corrections (psf)
qcovrq – ratio of corrected dynamic pressure to uncorrected dynamic pressure
vel – test section free stream velocity with walls up or down corrections (ft/sec)

** note   dtor = 0.0174532925199 {converts degrees to radians}*

** note   cls1u and cds1u are not written to output data file*

** note*
** array **dolup#** is   dolup1, dolup2, dolup3, dolup4, dolup5*
***
** array **dalup#.#** is   dalup1.1, dalup2.1, dalup3.1,*
**   dalup1.2, dalup2.2, dalup3.2,*
**   dalup1.3, dalup2.3, dalup3.3,*
**   dalup1.4, dalup2.4, dalup3.4,*
**   dalup1.5, dalup2.5, dalup3.5*
***
** array **dqup#.#** is   dqup1.1, dqup2.1, dqup3.1,*
**   dqup1.2, dqup2.2, dqup3.2,*
**   dqup1.3, dqup2.3, dqup3.3,*
**   dqup1.4, dqup2.4, dqup3.4,*
**   dqup1.5, dqup2.5, dqup3.5*

**** note***   *execute **walls up** calculations **if qu > 0.1 nbal = 1 wcode = 1***

*** uncorrected stability axis lift coefficient**

cls1u = ((fn1.2 * cos(alfunc * dtor)) – (fa1.2 * sin(alfunc * dtor)))/ (qu * sarea1)

*** uncorrected stability axis drag coefficient**

cds1u = ((fa1.2 * cos(alfunc * dtor)) + (fn1.2 * sin(alfunc * dtor))) / (qu * sarea1)

** note execute subfunction **walcor** which computes corrected alpha and q*

*** Tunnel parameters corrected for Q**

* note **pstat** calculation below is being reviewed

vel = velu * sqrt(q / qu)

pstat = pa – q

qcovrq = q / qu

Apply Estimates of Heyson's Corrections - Walls Down

Input Variables

alfunc – uncorrected model(body) angle of attack (deg)
daldn#.# - array of alpha correction coefficients; constants
doldn# - array of drag over lift coefficients; constants
dqdn#.# - array of q correction coefficients; constants
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
nbal – balance number; constant
pa – ambient pressure (psf)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
sarea1 – wing area; constant (sqft)
velu – uncorrected test section free stream velocity (ft/sec)
wcode – test section configuration code for classical wall corrections; constant

Output Variables

alpha – model(body) angle of attack with walls up or down corrections (deg)
pstat – test section static pressure with walls up or down corrections (ft/sec)
q – tunnel dynamic pressure corrected for compressibility; walls up or down corrections (psf)
qcovrq – ratio of corrected dynamic pressure to uncorrected dynamic pressure
vel – test section free stream velocity with walls up or down corrections (ft/sec)

** note* *dtor = 0.0174532925199 {converts degrees to radians}*

** note* *cls1u and cds1u are not written to output data file*

```
* note
* array doldn#  is        doldn1, doldn2, doldn3, doldn4, doldn5
*
* array daldn#.# is       daldn1.1,  daldn2.1,  daldn3.1,
*                         daldn1.2,  daldn2.2,  daldn3.2,
*                         daldn1.3,  daldn2.3,  daldn3.3,
*                         daldn1.4,  daldn2.4,  daldn3.4,
*                         daldn1.5,  daldn2.5,  daldn3.5
*
* array dqdn#.# is        dqdn1.1,  dqdn2.1,  dqdn3.1,
*                         dqdn1.2,  dqdn2.2,  dqdn3.2,
*                         dqdn1.3,  dqdn2.3,  dqdn3.3,
*                         dqdn1.4,  dqdn2.4,  dqdn3.4,
*                         dqdn1.5,  dqdn2.5,  dqdn3.5
```

**** note*** *execute* ***walls down*** *calculations* ***if qu > 0.1 nbal = 1 wcode = 2***

*** uncorrected stability axis lift coefficient**
cls1u = ((fn1.2 * cos(alfunc * dtor)) – (fa1.2 * sin(alfunc * dtor)))/ (qu * sarea1)

*** uncorrected stability axis drag coefficient**
cds1u = ((fa1.2 * cos(alfunc * dtor)) + (fn1.2 * sin(alfunc * dtor))) / (qu * sarea1)

** note* *execute subfunction* ***walcor*** *which computes corrected alpha and q*

*** Tunnel parameters corrected for Q**

* note **pstat** calculation below is being reviewed

vel = velu * sqrt(q / qu)

pstat = pa – q

qcovrq = q / qu

Blockage and Jet Boundary Corrections

Input Variables

alfunc – uncorrected model(body) angle of attack (deg)
beta – model(body) sideslip angle (deg)
bodyblok – solid-blockage velocity effect for a body of revolution; constant
bspan1 – wing span; constant (ft)
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
jbcorr2 – jet boundary angle of attack factor; constant
machu – uncorrected free stream mach number
pstatu – uncorrected test section static pressure (psf)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
rhou – uncorrected air density
rnftu – uncorrected test section reynolds number
sarea1 – wing area; constant (sqft)
tsarea – test section area; constant
velu – uncorrected test section free stream velocity
wcode – test section configuration code for classical wall corrections; constant
wingblok – solid-blockage velocity effect for a wing; constant

Output Variables

alpha – model(body) angle of attack corrected for jet boundary (deg)
delalp – jet boundary alpha correction
mach – free stream mach number corrected for blockage
pstat – test section static pressure; walls up or down or blockage corrections (psf)
q – tunnel dynamic pressure corrected for compressibility;
walls up or walls down or blockage corrections (psf)
qcovrq – ratio of corrected dynamic pressure to uncorrected dynamic pressure
rho – air density corrected for blockage (slugscf)
rnft– test section reynolds number corrected for blockage (1/feet)
vel – test section free stream velocity; walls up or walls down or blockage corrections (ft/sec)

** note* *pie, csaf, fds1u, cls1u, cd1u, acd, machu2, and bcf are*
*** *not written to output data file*

** note* *dtor = 0.0174532925199 {converts degrees to radians}*

```
    if  (wcode = 3) then

      pie = 3.14159265

      csaf =   sarea1 / (tsarea * 4.0)


* uncorrected stability axis drag force
      fds1u = (fa1.2 * cos(alfunc * dtor)) + (fn1.2 * sin(alfunc * dtor))

* uncorrected stability axis lift coefficient
      cls1u = ((fn1.2 * cos(alfunc * dtor)) – (fa1.2 * sin(alfunc * dtor))) / (qu * sarea1)

* uncorrected wind axis drag coefficient
      cd1u = ((fds1u * cos(beta * dtor)) – (fy1.2 * sin(beta * dtor))) / (qu * sarea1)

* Blockage calculations


      acd = cd1u – (cls1u * cls1u * sarea1) / (pie * (bspan1 / 12.0)**2)

      machu2 = machu * machu

* jbcorr4(blockage correction) is wingblok + bodyblok

      bcf = (wingblok + bodyblok) / (1.0 – machu2)**1.5 + (1.0 + 0.4 * machu2)
            * acd * csaf / (1.0 – machu2)

      mach = machu * (1.0 + (1.0 + 0.2 * machu2) * bcf)

      pstat = pstatu * (1.0 – 1.4 * machu2 * bcf)

      q = qu * (1.0 + (2.0 – machu2) * bcf)

      qcovrq = q / qu

      rho = rhou * (1.0 – machu2 * bcf)

      rnft = rnftu * (1.0 + (1.0 – 0.7 * machu2) * bcf)

      vel = velu * (1.0 + bcf)

* Jet boundary angle of attack correction

      delalp = jbcorr2 * cls1u

      alpha = alfunc + delalp

    endif
```

Heyson's Boundary(Wall) Interference Functions

Harry Heyson wrote several documents which described calculating boundary(wall) interference factors for a variety of configurations. In reference 2, Heyson discussed sixteen software functions. Curently, the 14- by 22-Foot Subsonic Tunnel's software utilizes function 2. Any of the other functions may be implemented upon request. The function number and the type of interference for a specific model are as follows.

Function	***Interference***
1	wind tunnel interference near a vanishingly small model
2	average wind tunnel interference over a swept wing
3	distribution od wind tunnel interference over the span of a swept wing
4	average wind tunnel interference over a tail behind a swept wing
5	average wind tunnel interference over a swept wing caused by the presence of lifting jets
6	distribution of wind tunnel interference over the span of a swept wing caused by the presence of lifting jets
7	average wind tunnel interference over a tail caused by the presence of lifting jets
8	average wind tunnel interference over a single rotor
9	distribution of wind tunnel interference over the lateral axis of a single rotor
10	distribution of wind tunnel interference over the longitudinal axis of a single rotor
11	average wind tunnel interference over a tail behind a single rotor
12	average wind tunnel interference over tandem rotors
13	average wind tunnel interference over unloaded rotor configuration
14	average wind tunnel interference over a tail behind an unloaded rotor configuration
15	average wind tunnel interference over side-by-side rotor configuration

16	average wind tunnel interference over a tail behind a side-by-side rotor configuration

Refer to references 2, 3, and 4 for details concerning Heyson's boundary interference calculations.

Heyson's Boundary(Wall) Interference
Function 2 - Average Wind Tunnel Inteference Over a Swept Wing

Input Variables

alfunc – uncorrected model(body) angle of attack (deg)
beta – model(body) sideslip angle (deg)
bspan1 – wing span; constant (ft)
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
gamma – ratio of tunnel width to height; constant
hannon – used in eta1 calculation; constant
hgt – height of the model height reference point above the test section floor (inches)
lambda – angle of sweep angle; constant
nbal – balance number; constant
q – tunnel dynamic pressure with compressibility; walls up or walls down or blockage corrections (psf)
rho – air density corrected for blockage (slugscf)
rlength – distance from (1) the center of the vertical mast to the model height reference point
or (2) the center of the pitch rotation pivot to the model height reference point;
constant (inches)
sarea1 – reference area; constant (sqft)
semispan – semispan model flag; constant
sigma – ratio of wing apn to the tunnel width; constant
tsarea – test section area; constant (sqft)
tsconf – test section configuration code for Heyson wall corrections; constant
vlength – vertical distance from the pivot center on the vertical strut to the model reference point;
constant (inches)
wingar – wing aspect ratio; constant (ft)
wingload – wind load configuration; constant
zonecon – ratio of wind tunnel semi-height to height of model above tunnel floor; constant
zoneflg – zeta1 calculation flag; constant

Output Variables

alfac – model(body) angle of attack with Heyson wall corrections (deg)
cdc – drag coefficient with Heyson wall corrections
clc – lift coefficient with Heyson wall corrections
dragc – drag corrected with Heyson wall corrections
liftc – lift corrected with Heyson wall corrections
lodc – lift over drag with Heyson wall corrections
qc – tunnel dynamic pressure corrected for compressibility; blockage & Heyson wall corrections (psf)
xeff – wake deflection angle (deg)

** note*   *dtor = 0.0174532925199 {converts degrees to radians}*

** note*   *if* ***hannon = 0.0 rlength = 0.0 vlength = 0.0*** *then* ***eta1 = 1.0***
***   ***14x22-Foot Subsonic Tunnel*** *is* ***21.75 feet wide*** *and* ***14.5 feet high***
***   *gamma = 21.75 / 14.5 = 1.5*
***   *sigma = bpsan1 / 21.75 where bspan1 is in feet*

```
*                    wingar = (bpsan1 * bspan1) / sarea1  where  bpsan1 is in feet and
*                                                                 sarea1 is in square feet
*                    tsarea = 21.75 * 14.5 = 315.375
*                    bhtw =  half the tunnel width (inches); not written to output data file


      bhtw = 10.875 * 12.0
      xeff = 0.0

      if  (nbal = 1.0 and tsconf > 0.0)  then

        if  ( zoneflg = 1.0 ) then
          if ( hgt > 0.0 ) then
            zeta1 = zonecon / hgt
          else
            zeta1 = zonecon
          endif
        else
          zeta1 = zonecon
        endif

        xwid = (hannon+(rlength*cos(alpha*dtor))-(vlength*sin(alpha*dtor)))
                  * sin(beta*dtor)
        eta1 = (bhtw-xwid)/bhtw

        if  (semispan = 0.0) then
```

*** full wing span corrections**

```
           nforce = fn1.2
           aforce = fa1.2
           wrefarea = sarea1
           tsecarea = tsarea
           wingspan = bspan1

        else if  (semispan = 1.0) then
```

*** semi wing span corrections**

```
           nforce = fn1.2 * 2.0
           aforce = fa1.2 * 2.0
           wrefarea = sarea1 * 2.0
           tsecarea = tsarea * 2.0
           wingspan = bspan1 * 2.0

        endif
```

```
* note   execute subfunction  heyson  as follows
*                   call  heyson(tsconf, wingload, zeta1, eta1, gamma, sigma, lambda,
*                                   nforce, aforce, qoc, rhoc, alfunc, wrefarea, wingar,
*                                   tsecarea, wingspan, liftc, dragc, clc, cdc, qc, alfac, xeff)

          if  (semispan = 1.0) then
             liftc = liftc * 0.5
             dragc = dragc * 0.5
          endif

          lodc = clc / cdc

      endif
```

Base(Cavity) Pressures

Input Variables

alpha - model angle of attack with jet boundary corrections (deg)
arpba#.# – array of axial force base pressure areas; constant (sqft)
arpbn#.# – array of normal force base pressure areas; constant (sqft)
arpbp#.# – array of pitching moment base pressure moment arm*areas; constant(in-sqft)
arpbr#.# – array of rolling moment base pressure moment arm*areas; constant (in-sqft)
arpbs#.# – array of side force base pressure areas; constant (sqft)
arpby#.# – array of yawing moment base pressure moment arm*areas; constant (in-sqft)
bspan1 – wing span; constant (ft)
chord1 – wing aerodynamic chord; constant (ft)
pbase# – array of base pressures (psf)
pstat - tunnel static pressure with walls up/down or blockage corrections (psf)
q – tunnel dynamic pressure with compressibility; walls up or walls down or blockage corrections (psf)
sarea1 – wing area; constant (sqft)

Output Variables

cabase1 - axial force base pressure tare coefficient calculated using axial force
base pressure tare applied in final axial force calculation
cdbase1 – drag base pressure tare coefficient
cnbase1 – normal force base pressure tare coefficient
cpbase# – array of base pressure coefficients
cpmbase1 – pitching moment base pressure tare coefficient
crmbase1 – rolling moment base pressure tare coefficient
cybase1 – side force base pressure tare coefficient calculated using side force
base pressure tare applied in final side force calculation
cymbase1 – yawing moment base pressure tare coefficient
dpbase# – array of differential base pressures (psf)
fabase1 - axial force base pressure tare (lbs)
fnbase1 - normal force base pressure tare (lbs)
fybase1 - side force base pressure tare (lbs)
pmbase1 - pitching moment base pressure tare (ft-lbs)
rmbase1 - rolling moment base pressure tare (ft-lbs)
ymbase1 - yawing moment base pressure tare (ft-lbs)

** note* *maximum number of base(cavity) pressures per balance is 25*
* *cavity pressures are calculated using base pressure names & equations*

*** base(cavity) pressure areas (constants with default values of 0.0)**

arpba1.1 *(axial force; base pressure 1, balance 1)*
arpba2.1 *(axial force; base pressure 2, balance 1)*
arpba3.1 *(axial force; base pressure 3, balance 1)*
.
.
.
arpba23.1 *(axial force; base pressure 23, balance 1)*

arpba24.1 *(axial force; base pressure 24, balance 1)*
arpba25.1 *(axial force; base pressure 25, balance 1)*
.
.
.
arpby1.1 *(yawing moment; base pressure 1, balance 1)*
arpby2.1 *(yawing moment; base pressure 2, balance 1)*
arpby3.1 *(yawing moment; base pressure 3, balance 1)*
.
.
.
arpby23.1 *(yawing moment; base pressure 23, balance 1)*
arpby24.1 *(yawing moment; base pressure 24, balance 1)*
arpby25.1 *(yawing moment; base pressure 25, balance 1)*

*** differential base(cavity) pressures**

dpbase1 = pbase1-pstat *(base pressure 1)*
dpbase2 = pbase2-pstat *(base pressure 2)*
dpbase3 = pbase3-pstat *(base pressure 3)*
.
.
.
dpbase23 = pbase23-pstat *(base pressure 23)*
dpbase24 = pbase24-pstat *(base pressure 24)*
dpbase25 = pbase25-pstat *(base pressure 25)*

*** base(cavity) pressure coefficients**

cpbase1 = dpbase1/q *(base pressure 1)*
cpbase2 = dpbase2/q *(base pressure 2)*
cpbase3 = dpbase3/q *(base pressure 3)*
.
.
.
cpbase23 = dpbase23/q *(base pressure23)*
cpbase24 = dpbase24/q *(base pressure 24)*
cpbase25 = dpbase25/q *(base pressure 25)*

*** base(cavity) pressure tares**

fabase1= -((dpbase1*arpba1.1) + (dpbase2*arpba2.1) + (dpbase3*arpba3.1) + ….
+ (dpbase23*arpba23.1) + (dpbase24*arpba24.1)+ (dpbase25*arpba25.1))

fybase1 = (dpbase1*arpbs1.1) + (dpbase2*arpbs2.1)+ (dpbase3*arpbs3.1) + ….
+ dpbase23*arpbs23.1) + (dpbase24*arpbs24.1)+ (dpbase25*arpbs25.1)

fnbase1 = (dpbase1*arpbn1.1) + (dpbase2*arpbn2.1)+ (dpbase3*arpbn3.1) + ….
+ dpbase23*arpbn23.1) + (dpbase24*arpbn24.1)+ (dpbase25*arpbn25.1)

rmbase1 = (dpbase1*arpbr1.1) + (dpbase2*arpbr2.1)+ (dpbase3*arpbr3.1) + ….
+ dpbase23*arpbr23.1) + (dpbase24*arpbr24.1)+ (dpbase25*arpbr25.1)

pmbase1 = (dpbase1*arpbp1.1) + (dpbase2*arpbp2.1)+ (dpbase3*arpbp3.1) + ….
+ dpbase23*arpbp23.1) + (dpbase24*arpbp24.1)+ (dpbase25*arpbp25.1)

ymbase1 = (dpbase1*arpby1.1) + (dpbase2*arpby2.1)+ (dpbase3*arpby3.1) + ….
+ dpbase23*arpby23.1) + (dpbase24*arpby24.1)+ (dpbase25*arpby25.1)

** note dtor = 0.0174532925199 {converts degrees to radians}*

*** base(cavity) pressure tare coefficients**

cabase1 = fabase1/(q*sarea1)
cybase1 = fybase1/(q*sarea1)
cnbase1 = fnbase1/(q*sarea1)
crmbase1 = rmbase1/(q*sarea1*bspan1)
cpmbase1 = pmbase1/(q*sarea1*chord1)
cymbase1 = ymbase1/(q*sarea1*bspan1)

*** base(cavity) pressure tare drag coefficient**

cdbase1 = (cabase1*cos(alpha*dtor))+(cnbase1*sin(alpha*dtor))

Model Axis Components and Coefficients

Input Variables

bspan1 – wing span; constant (ft)
chord1– wing aerodynamic chord; constant (ft)
fabase1 - axial force base pressure tare (lbs)
fnbase1 - normal force base pressure tare (lbs)
fybase1 - side force base pressure tare (lbs)
pmbase1 - pitching moment base pressure tare (ft-lbs)
q – tunnel dynamic pressure corrected for compressibility; walls up or walls down or blockage corrections (psf)
rmbase1 - rolling moment base pressure tare (ft-lbs)
sarea1 – wing area; constant (sqft)
ymbase1 - yawing moment base pressure tare (ft-lbs)

Output Variables

ca1 - model axis axial force coefficient corrected for base pressure tare
cmx1 – model axis rolling moment coefficient corrected for base pressure tare
cmy1 – model axis pitching moment coefficient corrected for base pressure tare
cmz1 – model axis yawing moment coefficient corrected for base pressure tare
cn1 – model axis normal force coefficient corrected for base pressure tare
cy1 – model axis side force coefficient corrected for base pressure tare
fa1 – model(body) axis axial force corrected for base pressure tare (lbs)
fa1.2 - axial force rotated and translated to the model axis (lbs)
fn1 – model(body) axis normal force corrected for base pressure tare (lbs)
fn1.2 - normal force rotated and translated to the model axis (lbs)
fy1 – model(body) axis side force corrected for base pressure tare (lbs)
fy1.2 - side force rotated and translated to the model axis (lbs)
mx1 – model(body) axis rolling moment corrected for base pressure tare (in-lbs)
mx1.2 - rolling moment rotated and translated to the model axis (in-lbs)
my1 – model(body) axis pitching moment corrected for base pressure tare (in-lbs)
my1.2 - pitching moment rotated and translated to the model axis (in-lbs)
mz1 – model(body) axis yawing moment corrected for delcm and base pressure tare (in-lbs)
mz1.2 - yawing moment rotated and translated to the model axis (in-lbs)

*** balance components**

fa1 = fa1.2
fy1 = fy1.2
fn1 = fn1.2
mx1 = mx1.2
my1 = my1.2
mz1 = mz1.2

*** balance components & coefficients corrected for base pressure tares**

```
fa1 = fa1 - fabase1
fy1 = fy1 - fybase1
fn1 = fn1 - fnbase1
mx1 = mx1 - rmbase1
my1 = my1 - pmbase1
mz1 = mz1 - ymbase1

ca1 = fa1 / (q * sarea1)
cy1 = fy1 / (q * sarea1)
cn1 = fn1 / (q * sarea1)
cmx1 = mx1 / (q * sarea1 * bspan1)
cmy1 = my1 / (q * sarea1 * chord1)
cmz1 = mz1 / (q * sarea1 * bspan1)
```

Stability Axis Components and Coefficients

Input Variables

alpha – model angle of attack with jet boundary corrections (deg)
bspan1 – wing span; constant (ft)
chord1 – wing aerodynamic chord; constant (ft)
fa1 – model(body) axis axial force corrected for base pressure tare (lbs)
fn1 – model(body) axis normal force corrected for base pressure tare (lbs)
fy1 – model(body) axis side force corrected for base pressure tare (lbs)
mx1 – model(body) axis rolling moment corrected for base pressure tare (in-lbs)
my1 – model(body) axis pitching moment corrected for base pressure tare (in-lbs)
mz1 – model(body) axis yawing moment corrected for base pressure tare (in-lbs)
q – tunnel dynamic pressure corrected for compressibility; walls up or walls down or
blockage corrections (psf)
sarea1 – wing area; constant (sqft)

Output Variables

cds1 - stability axis drag coefficient corrected for base pressure tare
clsqr1 – lift coefficient squared corrected for base pressure tare
cls1 – stability axis lift coefficient corrected for base pressure tare
cmxs1 – stability axis rolling moment coefficient corrected for base pressure tare
cmys1 – stability axis pitching moment coefficient corrected for base pressure tare
cmzs1 – stability axis yawing moment coefficient corrected for base pressure tare
cys1 – model axis side force coefficient corrected for base pressure tare
dell – jet boundary drag correction
delp – jet boundary pitch correction
fds1 – stability axis drag corrected for base pressure tare (lbs)
fls1 – stability axis lift corrected for base pressure tare (lbs)
fys1 – stability axis side force corrected for base pressure tare (lbs)
ls/ds1 – lift-over-drag ratio corrected for base pressure tare
mxs1 – stability axis rolling moment corrected for base pressure tare (in-lbs)
mys1 – stability axis pitching moment corrected for base pressure tare (in-lbs)
mzs1 – stability axis yawing moment corrected for base pressure tare (in-lbs)

** note dtor = 0.0174532925199 {converts degrees to radians}*

*** balance components & coefficients corrected for base pressure tares**

fds1 = (fa1 * cos(alpha*dtor)) + (fn1 * sin(alpha*dtor))
fys1 = fy1
fls1 = (fn1 * cos(alpha*dtor)) - (fa1 * sin(alpha*dtor))
mxs1 = (mx1 * cos(alpha*dtor)) + (mz1 * sin(alpha*dtor))
mys1 = my1
mzs1 = (mz1 * cos(alpha*dtor)) - (mx1 * sin(alpha*dtor))

* *jet boundary drag force correction*

```
dell = (jbcorr1 * fls1 * fls1) / (q * sarea1)
fds1 = fds1 + dell
```

* *jet boundary pitching moment correction*

```
delp = jbcorr3 * fls1 * chord1
mys1 = mys1 –  delp

cds1 = fds1 / (q * sarea1)
cys1 = fys1 / (q * sarea1)
cls1 = fls1 / (q * sarea1)
cmxs1 = mxs1 / (q * sarea1 * bspan1)
cmys1 = mys1 / (q * sarea1 * chord1)
cmzs1 = mzs1 / (q * sarea1 * bspan1)

ls/ds1 = cls1 / cds1
clsqr1 = cls1 * cls1
```

Wind Axis Components and Coefficients

Input Variables

beta – model sideslip angle (deg)
bspan1 – wing span; constant (ft)
chord1 – wing aerodynamic chord; constant (ft)
fds1 – stability axis drag corrected for base pressure tare (lbs)
fls1 – stability axis lift corrected for base pressure tare (lbs)
fys1 – stability axis side force corrected for base pressure tare (lbs)
mxs1 – stability axis rolling moment corrected for base pressure tare (in-lbs)
mys1 – stability axis pitching moment corrected for base pressure tare (in-lbs)
mzs1 – stability axis yawing moment corrected for base pressure tare (in-lbs)
q – tunnel dynamic pressure corrected for compressibility; walls up or walls down or blockage corrections (psf)
sarea1 – wing area; constant (sqft)

Output Variables

cc1 – wind axis crosswind coefficient corrected for base pressure tare
cd1 - wind axis drag coefficient corrected for base pressure tare
cl1 – wind axis lift coefficient corrected for base pressure tare
cmxw1 – wind axis rolling moment coefficient corrected for base pressure tare
cmyw1 – wind axis pitching moment coefficient corrected for base pressure tare
cmzw1 – wind axis yawing moment coefficient corrected for base pressure tare
fc1 – wind axis crosswind corrected for base pressure tare (lbs)
fd1 – wind axis drag corrected for base pressure tare (lbs)
fl1 – wind axis lift corrected for base pressure tare (lbs)
ld1 – lift-over-drag ratio corrected for base pressure tare
mxw1 – wind axis rolling moment corrected for base pressure tare (in-lbs)
myw1 – wind axis pitching moment corrected for base pressure tare (in-lbs)
mzw1 – wind axis yawing moment corrected for base pressure tare (in-lbs)

** note dtor = 0.0174532925199 {converts degrees to radians}*

*** balance components & coefficients corrected for base pressure tares**

fd1 = (fds1 * cos(beta*dtor)) - (fys1 * sin(beta*dtor))
fc1 = (fys1 * cos(beta*dtor)) + (fds1 * sin(beta*dtor))
fl1 = fls1
mxw1 = (mxs1 * cos(beta*dtor)) + (mys1 * sin(beta*dtor))
myw1 = (mys1 * cos(beta*dtor)) - (mxs1 * sin(beta*dtor))
mzw1 = mzs1

```
cd1 = fd1 / (q * sarea1)
cc1 = fc1 / (q * sarea1)
cl1 = fl1 / (q * sarea1)
cmxw1 = mxw1 / (q * sarea1 * bspan1)
cmyw1 = myw1 / (q * sarea1 * chord1)
cmzw1 = mzw1 / (q * sarea1 * bspan1)

ld1 = cl1 / cd1
```

Reference Axis Components and Coefficients

Input Variables

bspan1 – wing span; constant (ft)
chord1– wing aerodynamic chord; constant (ft)
fa1 – model(body) axis axial force corrected for base pressure tare (lbs)
fn1 – model(body) axis normal force corrected for base pressure tare (lbs)
fy1 – model(body) axis side force corrected for base pressure tare (lbs)
mx1 – model(body) axis rolling moment corrected for base pressure tare (in-lbs)
my1 – model(body) axis pitching moment corrected for base pressure tare (in-lbs)
mz1 – model(body) axis yawing moment corrected for base pressure tare (in-lbs)
phir1 – reference axis roll angle (deg)
psir1 – reference axis yaw angle (deg)
q – tunnel dynamic pressure corrected for compressibility; walls up or walls down or blockage corrections (psf)
refangles - three-element array of angles (Euler input)
refflg - three-element array of rotation flags (Euler input); one flag for each angle where values are [-1 = yaw rotation, 0 = pitch rotation, 1 = roll rotation]
refin - six-element array (Euler input)
sarea1 – wing area; constant (sqft)
thetar1 – reference axis pitch angle (deg)
xref1 – moment transfer distance(x direction) measured in the model force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)
yref1 – moment transfer distance(y direction) measured in the model force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)
zref1 – moment transfer distance(z direction) measured in the body force axis system from the moment center to the desired moment center(positive in the direction of positive model thrust(axial), side and normal force (ft)

Output Variables

faref1 – model(body) axis axial force rotated and translated to an arbitrary reference axis (lbs)
farefr1 – model(body) axis axial force rotated to an arbitrary reference axis (lbs)
fnref1 – model (body) axis normal force rotated and translated to an arbitrary reference axis (lbs)
fnrefr1 – model (body) axis normal force rotated to an arbitrary reference axis (lbs)
fyref1 – model (body) axis side force rotated and translated to an arbitrary reference axis (lbs)
fyrefr1 – model (body) axis side force rotated to an arbitrary reference axis (lbs)
mxref1 – model (body) axis rolling moment rotated and translated to an arbitrary reference axis (in-lbs)
mxrefr1 – model (body) axis rolling moment rotated to an arbitrary reference axis (in-lbs)
myref1 – model (body) axis pitching moment rotated and translated to an arbitrary reference axis (in-lbs)
myrefr1 – model (body) axis pitching moment rotated to an arbitrary reference axis (in-lbs)
mzref1 – model (body) axis yawing moment rotated and translated to an arbitrary reference axis (in-lbs)
mzrefr1 – model (body) axis yawing moment rotated to an arbitrary reference axis (in-lbs)
refout - six-element array (Euler output)

*** forces and moments rotated to the model axis**

For a yaw-pitch-roll rotation, Let

```
refangles(1) = psir1
refangles(2) = thetar1
refangles(3) = phir1
refflg(1) = -1
 refflg(2) = 0
 refflg(3) = 1
```

and

```
refin(1) = fa1
refin(2) = fy1
refin(3) = fn1
refin(4) = -mx1
refin(5) = my1
refin(6) = -mz1
```

** note execute subfunction **euler** which performs angle rotation(s)*

*** model(body) axis components rotated to an arbitrary reference axis**

```
farefr1 = refout(1)
fyrefr1 = refout(2)
fnrefr1 = refout(3)
mxrefr1 = -refout(4)
myrefr1 = refout(5)
mzrefr1 = -refout(6)
```

*** model(body) axis components and coefficients rotated and translated to**
*** an arbitrary reference axis**

```
faref1 = farefr1
fyref1 = fyrefr1
fnref1 = fnrefr1
mxref1 = mxrefr1 + (fnrefr1 * yref1) - (fyrefr1 * zref1)
myref1 = myrefr1 - (fnrefr1 * xref1) - (farefr1 * zref1)
mzref1 = mzrefr1 - (fyrefr1 * xref1) - (farefr1 * yref1)

caref1 = faref1 / (q * sarea1)
cyref1 = fyref1 / (q * sarea1)
cnref1 = fnref1 / (q * sarea1)
cmxref1 = mxref1 / (q * sarea1 * bspan1)
cmyref1 = myref1 / (q * sarea1 * chord1)
cmzref1 = mzref1 / (q * sarea1 * bspan1)
```

Model and Wall Pressures

Input Variables

dpat – difference between the settling chamber total pressure and the ambient pressure "ptot - pa" (psf)
dpinf – dynamic pressure uncorrected for compressibility (psf)
dprt – WICS set pressure (psf)
p# – array of esp pressures (psf)
q– tunnel dynamic pressure corrected for compressibility; wall corrections (psf)
qu – uncorrected tunnel dynamic pressure with compressibility (psf)
wsp# – array of WICS esp pressures (psf)

Output Variables

cpp# – array of pressure coefficients
wpp# – array of WICS pressure coefficients

** note   maximum number of esp modules = 16*
**   maximum number of pressures per esp module = 64*

** note   qcalc is not written to output data file*

*** pressure coefficients**

```
if  (q.eq.0.0) then
  qcalc = 1.0
else
  qcalc = q
endif

cpp101 = (p101-dpat+dpinf)/qcalc              (module 1, port 1)
cpp102 = (p102-dpat+dpinf)/qcalc              (module 1, port 2)
cpp103 = (p103-dpat+dpinf)/qcalc              (module 1, port 3)
  .
  .
  .
cpp162 = (p162-dpat+dpinf)/qcalc              (module 1, port 62)
cpp163 = (p163-dpat+dpinf)/qcalc              (module 1, port 63)
cpp164 = (p164-dpat+dpinf)/qcalc              (module 1, port 64)
  .
  .
  .
cpp1601 = (p1601-dpat+dpinf)/qcalc   (module 16, port 1)
cpp1602 = (p1602-dpat+dpinf)/qcalc   (module 16, port 2)
cpp1603 = (p1603-dpat+dpinf)/qcalc   (module 16, port 3)
  .
  .
  .
cpp1662 = (p1662-dpat+dpinf)/qcalc   (module 16, port 62)
cpp1663 = (p1663-dpat+dpinf)/qcalc   (module 16, port 63)
```

```
cpp1664 = (p1664-dpat+dpinf)/qcalc   (module 16, port 64)
```

*** Wall Interference Correction System pressures**

```
if  (qu.eq.0.0) then
  qcalc = 1.0
else
  qcalc = qu
endif
```

*   ***South Wall Pressures***

```
wpp5001 = (wsp5001+dpinf-dprt)/qcalc      (module 50, port 1)
wpp5002 = (wsp5002+dpinf-dprt)/qcalc      (module 50, port 2)
wpp5003 = (wsp5003+dpinf-dprt)/qcalc      (module 50, port 3)
  .
  .
  .
wpp5062 = (wsp5062+dpinf-dprt)/qcalc      (module 50, port 62)
wpp5063 = (wsp5063+dpinf-dprt)/qcalc      (module 50, port 63)
wpp5064 = (wsp5064+dpinf-dprt)/qcalc      (module 50, port 64)


wpp5101 = (wsp5101+dpinf-dprt)/qcalc      (module 51, port 1)
wpp5102 = (wsp5102+dpinf-dprt)/qcalc      (module 51, port 2)
wpp5103 = (wsp5103+dpinf-dprt)/qcalc      (module 51, port 3)
  .
  .       .
wpp5162 = (wsp5162+dpinf-dprt)/qcalc      (module 51, port 62)
wpp5163 = (wsp5163+dpinf-dprt)/qcalc      (module 51, port 63)
wpp5164 = (wsp5164+dpinf-dprt)/qcalc      (module 51, port 64)
```

*   ***Ceiling Pressures***

```
wpp5201 = (wsp5201+dpinf-dprt)/qcalc      (module 52, port 1)
wpp5202 = (wsp5202+dpinf-dprt)/qcalc      (module 52, port 2)
wpp5203 = (wsp5203+dpinf-dprt)/qcalc      (module 52, port 3)
  .
  .
  .
wpp5262 = (wsp5262+dpinf-dprt)/qcalc      (module 52, port 62)
wpp5263 = (wsp5263+dpinf-dprt)/qcalc      (module 52, port 63)
wpp5264 = (wsp5264+dpinf-dprt)/qcalc      (module 52, port 64)


wpp5301 = (wsp5301+dpinf-dprt)/qcalc      (module 53, port 1)
wpp5302 = (wsp5302+dpinf-dprt)/qcalc      (module 53, port 2)
wpp5303 = (wsp5303+dpinf-dprt)/qcalc      (module 53, port 3)
```

.
.
.
wpp5362 = (wsp5362+dpinf-dprt)/qcalc *(module 53, port 62)*
wpp5363 = (wsp5363+dpinf-dprt)/qcalc *(module 53, port 63)*
wpp5364 = (wsp5364+dpinf-dprt)/qcalc *(module 53, port 64)*

* ***North Wall Pressures***

wpp5401 = (wsp5401+dpinf-dprt)/qcalc *(module 54, port 1)*
wpp5402 = (wsp5402+dpinf-dprt)/qcalc *(module 54, port 2)*
wpp5403 = (wsp5403+dpinf-dprt)/qcalc *(module 54, port 3)*
.
.
.
wpp5462 = (wsp5462+dpinf-dprt)/qcalc *(module 54, port 62)*
wpp5463 = (wsp5463+dpinf-dprt)/qcalc *(module 54, port 63)*
wpp5464 = (wsp5464+dpinf-dprt)/qcalc *(module 54, port 64)*

wpp5501 = (wsp5501+dpinf-dprt)/qcalc *(module 55, port 1)*
wpp5502 = (wsp5502+dpinf-dprt)/qcalc *(module 55, port 2)*
wpp5503 = (wsp5503+dpinf-dprt)/qcalc *(module 55, port 3)*
.
.
.
wpp5562 = (wsp5562+dpinf-dprt)/qcalc *(module 55, port 62)*
wpp5563 = (wsp5563+dpinf-dprt)/qcalc *(module 55, port 63)*
wpp5564 = (wsp5564+dpinf-dprt)/qcalc *(module 55, port 64)*

* ***South Wall Centerline Pressures***

wpp5701 = (wsp5701+dpinf-dprt)/qcalc *(module 57, port 1)*
wpp5702 = (wsp5702+dpinf-dprt)/qcalc *(module 57, port 2)*
wpp5703 = (wsp5703+dpinf-dprt)/qcalc *(module 57, port 3)*
.
.
.
wpp5730 = (wsp5730+dpinf-dprt)/qcalc *(module 57, port 30)*
wpp5731 = (wsp5731+dpinf-dprt)/qcalc *(module 57, port 31)*
wpp5732 = (wsp5732+dpinf-dprt)/qcalc *(module 57, port 32)*

***** ***Ceiling Centerline Pressures***

wpp5801 = (wsp5801+dpinf-dprt)/qcalc *(module 58, port 1)*
wpp5802 = (wsp5802+dpinf-dprt)/qcalc *(module 58, port 2)*
wpp5803 = (wsp5803+dpinf-dprt)/qcalc *(module 58, port 3)*
.

.
.
wpp5830 = (wsp5830+dpinf-dprt)/qcalc   *(module 58, port 30)*
wpp5831 = (wsp5831+dpinf-dprt)/qcalc   *(module 58, port 31)*
wpp5832 = (wsp5832+dpinf-dprt)/qcalc   *(module 58, port 32)*

* ***North Wall Centerline Pressures***

wpp5901 = (wsp5901+dpinf-dprt)/qcalc   *(module 59, port 1)*
wpp5902 = (wsp5902+dpinf-dprt)/qcalc   *(module 59, port 2)*
wpp5903 = (wsp5903+dpinf-dprt)/qcalc   *(module 59, port 3)*
.
.
.
wpp5930 = (wsp5930+dpinf-dprt)/qcalc   *(module 59, port 30)*
wpp5931 = (wsp5931+dpinf-dprt)/qcalc   *(module 59, port 31)*
wpp5932 = (wsp5932+dpinf-dprt)/qcalc   *(module 59, port 32)*

References

1. Berrier, B. L., Leavitt, L. D., “Operating Characteristics of the Multiple Critical Venturi System and Secondary Calibration Nozzles Used for Weight-Flow Measurements in the Langley 16-Foot Transonic Tunnel,” NASA TM 86405, September 1985.

2. Heyson, Harry H., “FORTRAN Programs for Calculating Wind-Tunnel Boundary Interference,” NASA TM X-1740, 1969.

3. Heyson, Harry H., “Linearized Theory of Wind-Tunnel Jet-Boundary Corrections and Ground Effect for VTOL-STOL Aircraft,” NASA TR R-124, 1962.

4. Heyson, Harry H., “Use of Superposition in Digital Computers to Obtain Wind-Tunnel Interference Factors for Arbitrary Configurations, With Particular Reference to V/STOL Models,” NASA TR R-302, 1969.

5. Mercer, Charles E., Berrier, Bobby L., Capone, Francis J., Grayston,, Alan M., “Data Reduction Formulas for the 16-Foot Transonic Tunnel NASA Langley Research Center Revision 2,” NASA TM 107646, July 1992.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 08 - 2014	Technical Memorandum	

4. TITLE AND SUBTITLE
Data Reduction Functions for the Langley 14- by 22-Foot Subsonic Tunnel

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Boney, Andy D.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
122711.03.07.07.05.80

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-19877

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM-2014-218513

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 61
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Langley 14- by 22-Foot Subsonic Tunnel's data reduction software utilizes six major functions to compute the acquired data. These functions calculate engineering units, tunnel parameters, flowmeters, jet exhaust measurements, balance loads/model attitudes, and model /wall pressures. The input(required) variables, the output(computed) variables, and the equations and/or subfunction(s) associated with each major function are discussed.

15. SUBJECT TERMS
Balance loads; Model attitudes; Data reduction; Engineering units; Flowmeters; Functions; Jet exhaust measurements; Model pressures; Wall pressures; Subfunctions; Tunnel parameters

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	78	(443) 757-5802