MATLAB Programs for Converting Flow Bench Sensor Measurements to Flow Rate

Gerald Recktenwald*

April 27, 2006

1 Overview

This document describes MATLAB functions for converting flow bench sensor measurements to flow rate. The codes can be downloaded from the web site for ME 449, *Thermal Management Measurements*: `www.me.pdx.edu/~gerry/class/ME449/MATLAB`.

2 Pressure Transducers

The pressure transducers in the thermal lab have linearized output. The calibration data supplied by the manufacturers is used to obtain curve fits of the form

$$\delta p = c_1 V + c_2. \tag{1}$$

Table 1 lists MATLAB routines that evaluate the Equation (1) for the transducers in the lab. The source codes for `Omega05dp` and `Omega10dp` are given in Listing 1 and Listing 2, respectively.

*Mechanical and Materials Engineering Department, Portland State University, Portland, Oregon, `gerry@me.pdx.edu`

Table 1: Names of MATLAB routines for evaluating the calibration equation for pressure transducers in the thermal laboratory.

Range (inch H_2O)	Model Number	Serial Number	MATLAB Routine
0–0.5	PX653-0.5D5V	10523258	`Omega05dp.m`
±2.5	PX653-2.5BD5V	00204071	`Omega10dp.m`
0–3	PX653-03D5V	01212460	`Omega25dp.m`
0–10	PX653-10D5V	00100382	`Omega30dp.m`

3 Nozzle Flow Rate

The volumetric flow rate Q through a long radius nozzle is computed with

$$Q = C_d A_n Y \sqrt{\frac{2\Delta p}{\rho(1-\beta^4)}} \tag{2}$$

where C_d is the nozzle discharge coefficient, A_n is the area of the nozzle throat, Y is the expansion factor, which accounts for compressibility, d_t is the throat diameter, Δp is the measured pressure drop across the nozzle, ρ is the fluid density upstream of the nozzle, $\beta = d_t/D$ is the contraction ratio, and D is the diameter of the upstream duct. The generic equation for the discharge coefficient is

$$C_d = 0.9986 - \frac{7.006}{\sqrt{\mathrm{Re}_t}} + \frac{134.6}{\mathrm{Re}_t} \tag{3}$$

where the Reynolds number in he nozzle throat is

$$\mathrm{Re}_t = \frac{V_t d_t}{\nu} = \frac{4Q}{\pi d_t \nu} \tag{4}$$

$V_t = Q/A_n$ is the velocity in the throat, μ is the fluid viscosity evaluated at the pressure and temperature upstream of the nozzle.

The `nozzleFlow` function in Listing 3 evaluates Equation (2). The `nozzleFlow` function is used by several other MATLAB codes to perform data reduction for flow bench measurements.

4 Data Reduction for Fan Curves

The MATLAB codes for flow bench data reduction are demonstrated by obtaining the fan curves from measured data.

One of the trickiest (and potentially frustrating) steps in the process is getting the data into MATLAB variables. Two procedures for doing so are demonstrated: manual storage of the data, and reading the data from a plain text file.

4.1 Manual Data Storage

The `myFanData` function in Listing 4 stores data from a flow bench experiment. The data is hard-coded into assignment statements. For example

```
dn = [1.00; 1.00; 1.00; 1.00; 1.00; 1.00; 1.696; 1.696; ...
      1.696; 1.696; 1.696; 1.696] * 2.54e-2;
```

defines a column vector of nozzle throat diameters in meters. The `dn`, `vdpp`, `vdpn`, and `emf` variables are all column vectors with the same number of elements. The `rzone` and `patm` variables are scalars.

The `fanCurveManual` function in Listing 5 calls `myFanData` to retrieve the data, and then converts the data to a flow rate with the `nozzleFlow` function. Running `myFanManual` produces the plot in Figure 1 and the following text output.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
>> fanCurveManual

Data for Panaflow FBA12G24M:

     T1      dn         dp1      dpn         Q
    (C)     (cm)        (Pa)     (Pa)      (m^3/s)
   20.50    2.5400     17.16     613.8    0.015760
   20.58    2.5400     25.55     174.9    0.008353
   20.65    2.5400     32.07      47.3    0.004292
   20.75    2.5400     13.74    1205.2    0.022114
   20.82    2.5400     14.05    1136.7    0.021480
   20.82    2.5400     20.89     364.8    0.012127
   20.92    4.3078     16.54     107.1    0.018903
   20.92    4.3078      4.74     371.0    0.035440
   20.90    4.3078     20.58      57.9    0.013831
   20.95    4.3078      1.32     427.0    0.038044
   20.92    4.3078      8.46     308.8    0.032305
   20.87    4.3078     13.43     190.5    0.025307

Polynomial coefficients in fit to fan curve data:
     -2.00223342642e+007  Q^(4)
      2.24361684437e+005  Q^(3)
      5.19040184117e+004  Q^(2)
     -2.23526394578e+003  Q^(1)
      4.06730328445e+001  Q^(0)
```

4.2 The flowBench Utility Function

The `flowBench` function in Listing 6 converts vectors of flow bench sensor measurements to vectors of Δp and Q data. Like the `nozzleFlow` function, `flowBench` is (fairly) generic, and is used as a building block in other MATLAB programs for conversion of flow bench data. The `flowBench` function does contain hard-wired calls to pressure transducer conversion functions (`Omega05dp`

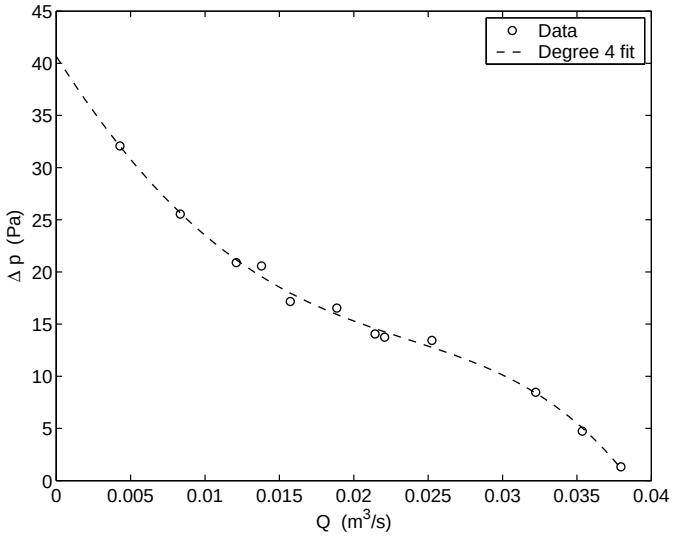

Figure 1: Fan curve plot produced by running `myFanManual`

Copyright © 2006, Gerald Recktenwald. All rights reserved.

and `Omega10dp`). If the transducers are changed, or if the `flowBench` function is to be used with another flow bench having different sensors, the `flowBench` code will need to be revised.

The `fanCurveMan2` function in Listing 7 shows how the `flowBench` function could be used. `fanCurveManal2` is a modified version of `fanCurveManual` that uses `flowBench` to perform the data reduction. Running `fanCurveMan2` produces the same output as running `fanCurveManual`.

4.3 Importing Data from Plain Text Files

The data used by `fanCurveManual` is hard-coded into MATLAB variables in the `myFanData` function. A more flexible approach is to store the data in a plain text file and then read the data into MATLAB variables. Storing the raw data in separate (text) files introduces some small complications (managing the file names and using a flexible means of reading the data), but it has the strong advantages in managing disparate data sets. For example, there is no need for each data set to have the same number of readings, or to be gathered at the same time. Data files can be added and removed from an analysis relatively easily.

The `loadFlowBenchData` function in Listing 9 reads the flow bench data when it is in the well defined form shown in Listing 10. The `loadFlowBenchData` function uses the `loadColData` utility from the NMM Toolbox[1]. The `loadColData` function is designed to read files like `Panaflow20V.dat` in Listing 10. The generic usage of `loadColData` is

```
[x,Y] = loadColData(fname,ncol,nhead,nrlabel)
```

The input parameters are ***fname***, a string containing the name of the data file (e.g. `Panaflow20V.dat`); ***ncol***, the total number of columns of numeric data in the file (excluding the header); ***nhead*** the number of *rows* of text at the top of the file to be discarded; and ***nrlabel***, the number of rows of label text between the header and the numeric data. Note that ***nhead*** can be set to read all of the text at the top of the file, in which case ***nrlabel*** should be set to zero. The output parameters are a vector ***x*** and a matrix ***Y***. The structure of ***x*** and ***Y*** can be represented by a table

x	***Y***				
x_1	$Y_{1,1}$	$Y_{1,2}$	$Y_{1,3}$	$Y_{1,4}$	$\cdots$
x_2	$Y_{2,1}$	$Y_{2,2}$	$Y_{2,3}$	$Y_{2,4}$	$\cdots$
x_3	$Y_{3,1}$	$Y_{3,2}$	$Y_{3,3}$	$Y_{3,4}$	$\cdots$
x_4	$Y_{4,1}$	$Y_{4,2}$	$Y_{4,3}$	$Y_{4,4}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

The columns of ***Y*** are separate variables (`vdpp`, `vdpn`, etc.) for the analysis. Each row of ***x*** and ***Y*** is a single measurement that is (presumably) at a different, steady-state, operating conditions.

The contents of `Pan20V` can be read into MATLAB variables with

```
>> [dn,vdpp,vdpn,emf] = loadFlowBenchData('Panaflow20V.dat',true);
```

[1] See G. Recktenwald, *Numerical Methods with* MATLAB 2000, Prentice-Hall.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
Data for Panaflow20V.dat:

    dn       emf       vdpp      vdpn
  (cm)      (mv)       (V)       (V)
 2.5400    0.0400    1.5500    1.9900
 2.5400    0.0430    1.8200    1.2850
 2.5400    0.0460    2.0300    1.0800
 2.5400    0.0500    1.4400    2.9400
 2.5400    0.0530    1.4500    2.8300
 2.5400    0.0530    1.6700    1.5900
 4.3078    0.0570    1.5300    1.1760
 4.3078    0.0570    1.1500    1.6000
 4.3078    0.0560    1.6600    1.0970
 4.3078    0.0580    1.0400    1.6900
 4.3078    0.0570    1.2700    1.5000
 4.3078    0.0550    1.4300    1.3100
```

Note that the `loadFlowBenchData` function does not read the atmospheric pressure, resistance zone box thermistor, and plenum diameter from the text file.

4.4 Multiple Fan Curves

Often it is desirable to combine multiple fan curves on a single plot. Doing so requires managing the input of data from different settings (say different fan voltages), and combining the data from multiple fan curves in a single plot. The `myFanCurves` function in Listing 8 shows how these tasks can be achieved.

To get data from multiple sources, `myFanCurves` uses the `loadColData` utility to read data from a plain text file. Running`myFanCurves` produces the following text output and the plot in Figure 2.

```
>> myFanCurves

Data for Panaflow20V.dat:

    dn       emf       vdpp      vdpn
  (cm)      (mv)       (V)       (V)
 2.5400    0.0400    1.5500    1.9900
 2.5400    0.0430    1.8200    1.2850
 2.5400    0.0460    2.0300    1.0800
 2.5400    0.0500    1.4400    2.9400
 2.5400    0.0530    1.4500    2.8300
 2.5400    0.0530    1.6700    1.5900
 4.3078    0.0570    1.5300    1.1760
 4.3078    0.0570    1.1500    1.6000
 4.3078    0.0560    1.6600    1.0970
 4.3078    0.0580    1.0400    1.6900
 4.3078    0.0570    1.2700    1.5000
 4.3078    0.0550    1.4300    1.3100


Reduced flow bench data:

     T1       dn        dp1       dpn         Q
   (C)      (cm)       (Pa)      (Pa)      (m^3/s)
  21.56    2.5400     17.16     613.8    0.015770
  21.63    2.5400     25.55     174.9    0.008358
  21.70    2.5400     32.07      47.3    0.004295
  21.80    2.5400     13.74    1205.2    0.022128
  21.88    2.5400     14.05    1136.7    0.021494
  21.88    2.5400     20.89     364.8    0.012135
```

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
   21.98    4.3078     16.54     107.1    0.018915
   21.98    4.3078      4.74     371.0    0.035463
   21.95    4.3078     20.58      57.9    0.013840
   22.00    4.3078      1.32     427.0    0.038068
   21.98    4.3078      8.46     308.8    0.032325
   21.93    4.3078     13.43     190.5    0.025323

Data for Panaflow24V.dat:

    dn       emf       vdpp      vdpn
  (cm)      (mv)       (V)       (V)
 2.5400    0.0160    2.3400    1.0510
 2.5400    0.0150    2.1400    1.2210
 2.5400    0.0150    2.0500    1.3380
 2.5400    0.0150    1.9600    1.5040
 2.5400    0.0130    1.9000    1.6230
 2.5400    0.0110    1.8300    1.7010
 2.5400    0.0060    2.0300    1.3980
 2.5400    0.0080    1.7700    1.8700
 2.5400    0.0080    1.7500    1.9560
 2.5400    0.0050    1.6800    2.2050


Reduced flow bench data:

     T1       dn         dp1       dpn         Q
   (C)      (cm)        (Pa)      (Pa)      (m^3/s)
  21.32    2.5400     41.70      29.2    0.003363
  21.30    2.5400     35.49     135.1    0.007340
  21.30    2.5400     32.69     207.9    0.009137
  21.30    2.5400     29.90     311.2    0.011210
  21.25    2.5400     28.03     385.3    0.012487
  21.20    2.5400     25.86     433.9    0.013257
  21.08    2.5400     32.07     245.3    0.009932
  21.13    2.5400     23.99     539.1    0.014790
  21.13    2.5400     23.37     592.6    0.015512
  21.05    2.5400     21.20     747.7    0.017433

Curve fit coefficients for Panaflow20V.dat
  -1.99663975318e+07  Q^(4)
   2.23527253823e+05  Q^(3)
   5.18483763443e+04  Q^(2)
  -2.23392510758e+03  Q^(1)
   4.06729584655e+01  Q^(0)

Curve fit coefficients for Panaflow24V.dat
   9.44944317982e+08  Q^(4)
  -3.74076390511e+07  Q^(3)
   5.15182496406e+05  Q^(2)
  -4.37545430507e+03  Q^(1)
   5.18858651633e+01  Q^(0)
```

Copyright © 2006, Gerald Recktenwald. All rights reserved.

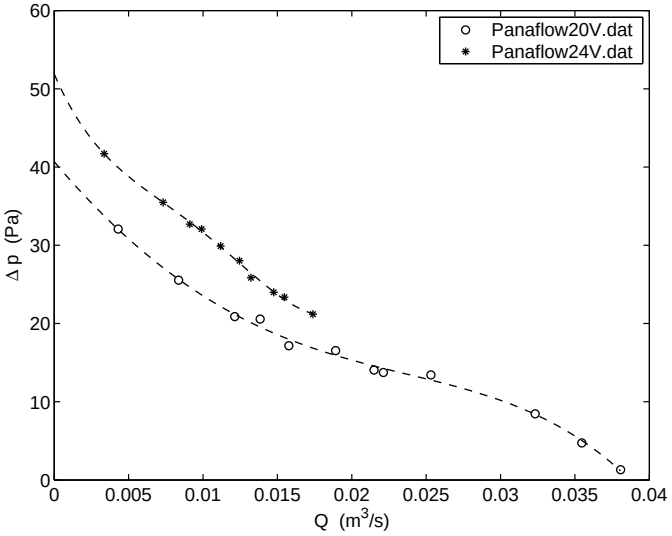

Figure 2: Two fan curves produced by `myFanCurves`

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function dp = Omega05dp(v,units)
% Omega25dp  Convert voltage to pressure for Omega model PX653-0.5D5V transducer
%            Calibration is for serial number 10523258
%            Output is 1 to 5 Volts for pressure range 0 to 0.5 inch H20
%
% Synopsis:  dp = Omega05dp
%            dp = Omega05dp(v)
%            dp = Omega05dp(v,units)
%
% Input:  v = voltage output of transducer
%         units = (string) to indicate the system of units for pressure.
%                   If units = 'SI', (default) then dp is in Pascals
%                   If units = 'BG', dp is in inches of H20
%                 Default: units = 'SI'. Other value is units = 'BG'
%
% Output:  dp = EITHER the calibration uncertainty for the pressure
%               transducer or the pressure differential across the transducer
%               dp = Omega05dp; (with no input arguments) returns the
%               estimate of the calibration uncertainty.  dp = Omega05dp(v)
%               returns the pressure differential for voltage v

inchH2O_toPa = 998*9.8*2.54e-2;  % multiplier to convert inch H2O to Pa
if nargin<1
  dp = 0.0015*inchH2O_toPa;  % calibration uncertainty in Pascal
  return
end
if nargin<2, units='SI';  end

% -- Range check on transducer output
if any(v<1 | v>5)
  fprintf('\nWarning in %s:\n',mfilename);
  fprintf('\tv = %g (volts) is out of range\n',v(v<1|v>5));
end
% --- Calibration equation
h = 0.125039641376784 * v - 0.124734996011382;  % delta p, inch H20
if strcmpi(units(1:2),'SI')
  dp = h*inchH2O_toPa;                    % convert inch H2O to Pa
else
  dp = h;
end
```

Listing 1: The `Omega05dp` function evaluates Equation (1) for the pressure transducer with 0–0.5 H_2O range.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function dp = Omega10dp(v,units)
% Omega10dp  Convert voltage to pressure for Omega model PX653-10D5V transducer
%            Calibration is for serial number 00100382
%            Output is 1 to 5 Volts for pressure range 0 to 10 inch H20
%
% Synopsis:  dp = Omega10dp
%            dp = Omega10dp(v)
%            dp = Omega10dp(v,units)
%
% Input:  v = voltage output of transducer
%         units = (string) to indicate the system of units for pressure.
%                   If units = 'SI', (default) then dp is in Pascals
%                   If units = 'BG', dp is in inches of H20
%                 Default: units = 'SI'. Other value is units = 'BG'
%
% Output:  dp = EITHER the calibration uncertainty for the pressure
%               transducer or the pressure differential across the transducer
%               dp = Omega10dp; (with no input arguments) returns the
%               estimate of the calibration uncertainty.  dp = Omega10dp(v)
%               returns the pressure differential for voltage v

inchH2O_toPa = 998*9.8*2.54e-2;  % multiplier to convert inch H2O to Pa
if nargin<1
  dp = 0.029*inchH2O_toPa;  % calibration uncertainty in Pascal
  return
end
if nargin<2, units='SI';  end

% -- Range check on transducer output
if any(v<1 | v>5)
  fprintf('\nWarning in %s:\n',mfilename);
  fprintf('\tv = %g (volts) is out of range\n',v(v<1|v>5));
end
% --- Calibration equation
h = 2.50600948477838 * v - 2.51613612227667;  % delta p, inch H20
if strcmpi(units(1:2),'SI')
  dp = inchH2O_toPa*h;                    % convert inch H2O to Pa
else
  dp = h;
end
```

Listing 2: The `Omega10dp` function evaluates Equation (1) for the pressure transducer with 0–10 H_2O range.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function Q = nozzleFlow(d,dplen,dpn,p,T)
% nozzleFlow  Volumetric flow rate of air through a long radius nozzle.
%
% Synopsis:  Q = nozzleFlow(d,dplen,dpn)
%            Q = nozzleFlow(d,dplen,dpn,p)
%            Q = nozzleFlow(d,dplen,dpn,p,T)
%
% Input: d = throat diameter,  (m)
%        dplen = upstream pipe (or plenum) diameter (m)
%        dpn = pressure drop across nozzle, (Pa)
%        p = (optional) pressure upstream of nozzle;  Default: p = 101325 (Pa)
%        T = (optional) temperature upstream of nozzle;  Default: T = 20 (C)
%
% Output: Q = volumetric flow rate, (m^3/s)
if nargin<4,  p = 101325;  end   %  standard atmosphere, (Pa)
if nargin<5,  T = 20;      end   %  degrees C
if dpn<eps || d<eps,  Q=0;  return;  end  %  allows for no-flow case on fan curves

% --- Evaluate fluid properties and other constants
mu = airViscosity(T);        %  kinematic viscosity
rho = p/(287*(T+273.15));    %  air density from ideal gas law
bbeta = d/dplen;
y = expansionFactor(p,dpn,bbeta,1.4);
area = 0.25*pi*d^2;
qcon = area*y*sqrt(2*dpn/(rho*(1-bbeta^4)));  rcon = rho*d/(area*mu);

% --- Initialize and loop until cd converges
tol = 5e-6;  it = 0;  maxit = 25;   cdold = 0;  cd = 0.9;
while abs(cdold-cd)>tol && it<maxit
  cdold = cd;
  Q = cd*qcon;
  Re = rcon*Q;
  cd = 0.9986 - 7.006/sqrt(Re) + 134.6/Re;
  it = it + 1;
end
if it>=maxit
  warning(sprintf('cd computations not converged after %d iterations',it));
end
```

Listing 3: The `nozzleFlow` function computes flow rate through long radius nozzles.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function [dn,dplen,vdpp,vdpn,emf,rzone,patm,name] = myFanData(verbose)
% myFanData  Manually store flow bench data for a fan curve
%
% Synopsis:  [dn,dplen,dpp,dpn,emf,rt,patm,name] = myFanData
%            [dn,dplen,dpp,dpn,emf,rt,patm,name] = myFanData(verbose)
%
% Input: verbose = (optional) flag to turn printing on/off
%                  Default: verbose = 1 (true), print raw data
%
% Output: dn = nozzle diamter (m)  (can be different for each measurement)
%         dplen = diameter of upstream plenum (m)
%         vdpp = voltage for pressure transducer across the plenum (V)
%         vdpn = voltage for pressure transducer across the nozzle (V)
%         emf = emf for thermocouple upstream of the nozzle (V)
%         rt = resistance of thermistor in the zone box (Ohms)
%         patm = atmospheric pressure in the lab (Pa)
%         name = string used to identify the data set

if nargin<1, verbose=1;  end

% --- Assign values from measurements
name = 'Panaflow FBA12G24M';   %  Label for this data set
rzone = 0.5*(12660+12397);     %  Thermistor measured at start and finish  (Ohms)
patm = 750.32 * 101325/760;    %  Atmospheric pressure measured with barometer (Pa)
dplen = 22*2.54e-2;            %  Upstream plenum diameter

dn = [1.00; 1.00; 1.00; 1.00; 1.00; 1.00; 1.696; 1.696; ...
      1.696; 1.696; 1.696; 1.696] * 2.54e-2;
vdpp = [1.55; 1.82; 2.03; 1.44; 1.45; 1.67; 1.53; 1.15; ...
        1.66; 1.04; 1.27; 1.43];
vdpn = [1.99; 1.285; 1.08; 2.94; 2.83; 1.59; 1.176; 1.60; ...
       1.097; 1.69;  1.50; 1.31];
emf = [0.040; 0.043; 0.046; 0.050; 0.053; 0.053; 0.057; 0.057; ...
       0.056; 0.058; 0.057; 0.0550]*1e-3;  %  convert mV to V

if (verbose)
  % --- Print table of raw data
  fprintf('\nData for %s:\n',name);
  fprintf('\tPlenum diameter      = %12.3e m  (%8.3f inch)\n',dplen,dplen/2.54e-2);
  fprintf('\tZone box thermistor  = %12.3e Ohm\n',rzone);
  fprintf('\tAtmospheric pressure = %12.4e Pa\n',patm);
  fprintf('\n    dn       emf       vdpp      vdpn\n');
  fprintf('   (cm)      (mv)      (V)       (V)\n');
  for i=1:length(dn)
    fprintf('%8.4f  %8.4f  %8.4f  %8.4f\n',dn(i)*100,emf(i)*1000,vdpp(i),vdpn(i));
  end
end
```

Listing 4: The `myFanData` function manually assigns flow bench data for a fan curve measurement.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function fanCurveManual(nfit)
% fanCurveManual  Data reduction for fan curve. Data entered manually
%
% Synopsis:  fanCurveManual
%            fanCurveManual(nfit)
%
% Input:  nfit = (optional) degree of polynomial curve fit to fan curve
%                Default:  nfit = 4;

if nargin<1, nfit=4;  end

% --- Get data
[dn,dplen,vdpp,vdpn,emf,rzone,patm,name] = myFanData(false);

% --- Convert pressure and temperature data.  All routines are vectorized
dp = Omega05dp(vdpp);               %  Delta p across plenum:  Pa
p  = patm + dp;                     %  Absolute p upstream of nozzle, Pa
dpn = Omega10dp(vdpn);              %  Delta p across nozzle, Pa
T = Ttemp(emf,thermistorT(rzone));  %  T upstream of nozzle, C

% --- Loop over converted data to compute flow rate.  nozzleFlow is not vectorized
Q = zeros(size(dn));   %  preallocate
b = dn/dplen;          %  b = beta
for i=1:length(dn)
  Q(i) = nozzleFlow(dn(i),dn(i)/dplen,dpn(i),p(i),T(i));    % nominal flow rate
end

% --- Print table of results
fprintf('\nData for %s:\n',name);
fprintf('\n     T1      dn         dp1      dpn         Q\n');
fprintf('    (C)     (cm)        (Pa)     (Pa)      (m^3/s)\n');
for i=1:length(dn)
  fprintf('%8.2f  %8.4f  %8.2f  %8.1f  %10.6f\n',T(i),dn(i)*100,dp(i),dpn(i),Q(i));
end

% --- Obtain curve fit, plot data
c = polyfit(Q,dp,nfit);
Qfit = linspace(0,max(Q));
dpfit = polyval(c,Qfit);
plot(Q,dp,'o',Qfit,dpfit,'--');
legend('Data',sprintf('Degree %d fit',nfit));
xlabel('Q  (m^3/s)');    ylabel('\Delta p  (Pa)');

% --- Print curve fit coefficients
fprintf('\nPolynomial coefficients in fit to fan curve data:\n');
for i=1:length(c)
  fprintf('\t%20.11e  Q^(%d)\n',c(i),nfit-i+1);
end
```

Listing 5: The `fanCurveManual` function converts flow bench measurements to a fan curve.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function  [Q,dp] = flowBench(dn,vdpp,vdpn,emf,Rzone,patm,dplen,verbose)
% flowBench  Convert flow bench measurements to flow rate and presure drop
%
% Synopsis:  [Q,dp] = flowBench(dn,vdpp,vdpn,emf)
%            [Q,dp] = flowBench(dn,vdpp,vdpn,emf,Rzone)
%            [Q,dp] = flowBench(dn,vdpp,vdpn,emf,Rzone,patm)
%            [Q,dp] = flowBench(dn,vdpp,vdpn,emf,Rzone,patm)
%            [Q,dp] = flowBench(dn,vdpp,vdpn,emf,Rzone,patm,dplen,verbose)
%
% Input:  dn = nozzle diamter (m)
%         vdpp = voltage for pressure transducer across the plenum (V)
%         vdpn = voltage for pressure transducer across the nozzle (V)
%         emf = emf for thermocouple upstream of the nozzle (mV)
%         Rzone = (optional) resistance of YSI 44006 thermistor in zone box (Ohm)
%                  Default: Rzone = 12000;
%         patm = (optional) atmospheric pressure (Pa).  Default: patm = 101325
%         dplen = (optional) hydraulic diameter of the plenum (m).
%                Default: dplen = 22*2.54e-2; (m) (size for small flow bench)
%         verbose = (optional) flag to turn printing on/off
%                  Default: verbose = 0 (false), print raw data
%
% Note:  Input paramters dn, vdpp, vdpn, and emf can be vectors or scalars.
%        These parameters must all have the same number of rows and columns.

if nargin<5 || isempty(Rzone), Rzone = 12000;    end
if nargin<6 || isempty(patm),  patm = 101325;    end
if nargin<7 || isempty(dplen), dplen = 22*2.54e-2;  end
if nargin<8, verbose = false;  end

% --- Convert pressure and temperature data.  All routines are vectorized
dp = Omega05dp(vdpp);               %  Delta p across plenum:  Pa
p  = patm + dp;                     %  Absolute p upstream of nozzle, Pa
dpn = Omega10dp(vdpn);              %  Delta p across nozzle, Pa
T = Ttemp(emf,thermistorT(Rzone));  %  T upstream of nozzle, C

% --- Loop over converted data to compute flow rate.  nozzleFlow is not vectorized
Q = zeros(size(dn));   %  preallocate
for i=1:length(dn)
  Q(i) = nozzleFlow(dn(i),dplen,dpn(i),p(i),T(i));    % nominal flow rate
end

% --- Print table of results
if verbose
  fprintf('\n\nReduced flow bench data:\n\n');
  fprintf('     T1      dn         dp1      dpn         Q\n');
  fprintf('    (C)     (cm)        (Pa)     (Pa)      (m^3/s)\n');
  for i=1:length(dn)
    fprintf('%8.2f  %8.4f  %8.2f  %8.1f  %10.6f\n',T(i),dn(i)*100,dp(i),dpn(i),Q(i));
  end
end
```

Listing 6: The `flowBench` function reads performs data reduction of flow bench sensor data.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function fanCurveMan2(nfit,verbose)
% fanCurveMan2  Data reduction for fan curve using the flowBench function.
%               Data entered manually via the myFanData function
%
% Synopsis:  fanCurveManual2
%            fanCurveManual2(nfit)
%            fanCurveManual2(nfit,verbose)
%
% Input:  nfit = (optional) degree of polynomial curve fit to fan curve
%                Default:  nfit = 4;
%         verbose = (optional) flag to control printing of raw and
%                   reduced data.  Default: verbose = false, no printing
if nargin<1 || isempty(nfit), nfit=4;  end
if nargin<2,  verbose = false;  end

% --- Get data and convert it
[dn,dplen,vdpp,vdpn,emf,rzone,patm,name] = myFanData(verbose);
[Q,dp] = flowBench(dn,vdpp,vdpn,emf,rzone,patm,dplen,verbose);

% --- Obtain curve fit, plot data
c = polyfit(Q,dp,nfit);
Qfit = linspace(0,max(Q));
dpfit = polyval(c,Qfit);
plot(Q,dp,'o',Qfit,dpfit,'--');
legend('Data',sprintf('Degree %d fit',nfit));
xlabel('Q  (m^3/s)');    ylabel('\Delta p  (Pa)');

% --- Print curve fit coefficients
fprintf('\nPolynomial coefficients in fit to fan curve data:\n');
for i=1:length(c)
  fprintf('\t%20.11e  Q^(%d)\n',c(i),nfit-i+1);
end
```

Listing 7: The `fanCurveManual2` function uses the `flowBench` utility to convert flow bench measurements to a fan curve.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function myFanCurves(nfit1,nfit2,verbose)
% myFanCurves  Simple m-file function to put two fan curves on a single plot
%
% Synopsis:  myFanCurves
%            myFanCurves(nfit1)
%            myFanCurves(nfit1,nfit2)
%            myFanCurves(nfit1,nfit2,verbose)
%
% Input:  nfit1 = degree of curve fit for data set 1.  Default: nfit1=4
%         nfit2 = degree of curve fit for data set 2.  Default: nfit2=4
%         verbose = flag to control printing of raw and reduced data.
%                   Default: verbose = true,  print the data
%
% Output:  Print out of curve fit coefficients, plot of fan curve, and
%          optional print out of raw and reduced data

if nargin<1 || isempty(nfit1), nfit1=4;      end
if nargin<2 || isempty(nfit2), nfit2=nfit1;  end
if nargin<3,  verbose = true;  end

% --- Specify the file names and degree of polynomial curve fits
fname1 = 'Panaflow20V.dat';  patm1 = (752/760)*101325;  Rz1 = 0.5*(12002+11980);
fname2 = 'Panaflow24V.dat';  patm2 = (749/760)*101325;  Rz2 = 0.5*(11875+11750);
dplen = 22*2.54e-2;

% --- Read data and let flowBench reduce it
[dn,vdpp,vdpn,emf] = loadFlowBenchData(fname1,verbose);
[Q1,dp1] = flowBench(dn,vdpp,vdpn,emf,Rz1,patm1,dplen,verbose);

[dn,vdpp,vdpn,emf] = loadFlowBenchData(fname2,verbose);
[Q2,dp2] = flowBench(dn,vdpp,vdpn,emf,Rz2,patm2,dplen,verbose);

% --- Plot fan curve data from measurements
plot(Q1,dp1,'o',Q2,dp2,'*');
xlabel('Q  (m^3/s)');    ylabel('\Delta p  (Pa)');   legend(fname1,fname2);

% --- Obtain curve fits and print coefficients to the command window
c1 = polyfit(Q1,dp1,nfit1);
fprintf('\nCurve fit coefficients for %s\n',fname1);
for i=1:length(c1)
  fprintf('\t%20.11e  Q^(%d)\n',c1(i),nfit1-i+1);
end
c2 = polyfit(Q2,dp2,nfit2);
fprintf('\nCurve fit coefficients for %s\n',fname2);
for i=1:length(c1)
  fprintf('\t%20.11e  Q^(%d)\n',c2(i),nfit2-i+1);
end

% --- Evaluate curve fit over the range of data (include Q=0 as lower limit)
Qfit1 = linspace(0,max(Q1));   dpfit1 = polyval(c1,Qfit1);
Qfit2 = linspace(0,max(Q2));   dpfit2 = polyval(c2,Qfit2);

% --- Add curve fits without erasing previous plot
hold('on');  plot(Qfit1,dpfit1,'r--',Qfit2,dpfit2,'b--');  hold('off');
```

Listing 8: The `myFanCurves` function uses `flowBench` to convert two sets of flow bench measurements. The fan curves are plotted in the same figure.

Copyright © 2006, Gerald Recktenwald. All rights reserved.

```
function [dn,vdpp,vdpn,emf] = loadFlowBenchData(fname,verbose)

% --- Constants for dealing with file headers and numbers of columns
nlabels = 0;     %  Don't read column labels
nhead = 10;      %  number of header lines to ignore
ncol = 6;        %  Total number of columns in the file

% --- Read data and let flowBench reduce it:  data files must use consistent format
[dn,D] = loadColData(fname,ncol,nhead,nlabels);
dn = dn*2.54e-2;          %  convert inches to m
vdpp = D(:,3);            %  p transducer between plenum and ambient, V
vdpn = D(:,4);            %  p transducer across nozzle, V
emf = D(:,5)*1e-3;        %  emf of tcouple upstream of nozzle, V

if (verbose)
  % --- Print table of raw data
  fprintf('\nData for %s:\n',fname);
  fprintf('\n    dn       emf       vdpp      vdpn\n');
  fprintf('   (cm)      (mv)      (V)       (V)\n');
  for i=1:length(dn)
    fprintf('%8.4f  %8.4f  %8.4f  %8.4f\n',dn(i)*100,emf(i)*1000,vdpp(i),vdpn(i));
  end
end
```

Listing 9: The `loadFlowBenchData` function reads raw flow bench data stored in a plain text file.

```
Group Members:                    Ms. Jones, Mr. Smith
Date of data collection:          4/18/2003
Ambient Pressure:                 750.32 mm Hg
Ambient Temperature:              21 C
Fan Model Number & Manufacturer:  Panaflow FBA12G24M
Nominal Fan Voltage:              20 V
Initial Resistance of thermistor: 12660 Ohms
Final Resistance of thermistor:   12397 Ohms
Nozzle_dia  dpp_manometer  dpn_gage  dpp_transducer  dpn_transducer  EMF
(inch)  (inch H20)  (inch H20)  (Volts)  (Volts)  (mV)
1.0      0.06     2.5     1.55    1.99     0.04
1.0     0.09     0.7     1.82    1.285    0.043
1.0     0.12     0.18    2.03    1.08     0.046
1.0     0.04     3       1.44    2.94     0.05
1.0     0.05     3       1.45    2.83     0.053
1.0     0.075    1.5     1.67    1.59     0.053
1.696    0.06     0.41    1.53    1.176    0.057
1.696    0.01     1.5     1.15    1.6      0.057
1.696    0.07     0.2     1.66    1.097    0.056
1.696    0        1.7     1.04    1.69     0.058
1.696    0.02     1.24    1.27    1.5      0.057
1.696    0.045    0.78    1.43    1.31     0.055
```

Listing 10: The `PanaFlow20V.dat` file contains flow bench sensor measurements for a fan curve experiment.

Copyright © 2006, Gerald Recktenwald. All rights reserved.