Design Environment for Low-Amplitude ThermoAcoustic Engines

DeltaE

Version 5.4

Tutorial and User's Guide

by
Bill Ward and Greg Swift
Los Alamos National Laboratory

LA-CC-01-13
Revision: 9/22/2004
For the latest version, visit http://www.lanl.gov/thermoacoustics/

Contents

I. Introduction

DeltaE—Design Environment for Low-Amplitude ThermoAcoustic Engines—is a computer program that can predict how a given thermoacoustic apparatus will perform, or can allow the user to design an apparatus to achieve desired performance. Version 5.4 is currently running on Windows-based and Macintosh PCs. Older versions were supported across some of the Unix variants (Solaris, IRIX, Alpha/OSF), but Version 5.4 is not available for Unix due to lack of demand or lack of an available compiler installation to support it. A Linux variant may be available by early 2005. DeltaE is substantially menu-oriented. Input data can be modified or entered via DeltaE's menu or using any text editor. Results can be examined via the menus, the operating system's text utilities, or any spreadsheet or graphics software.

For good portability, DeltaE is written in FORTRAN-77. The current executable code for IBM-compatibles requires at least a 386 processor, because it uses a DOS extender to create a flat 32-bit memory environment. (An older, version 2.1 DeltaE is still available which requires 333 kbytes of free RAM, and runs comfortably quickly on a 286 with math coprocessor, or anything more sophisticated.) All calculations are performed in double precision.

A. How It Works

DeltaE solves the one-dimensional wave equation based on the usual low-amplitude, 'acoustic' approximation. It solves the wave equation in a gas or liquid, in a geometry given by the user as a sequence of segments (no more than 200), such as ducts, compliances, transducers, and thermoacoustic stacks or regenerators. A glance through the figures below will orient the reader to the range of cases that DeltaE can handle.

A solution to the appropriate 1-d wave equation is found for each segment, with pressures and volume flow rates matched at the junctions between segments. In stacks, the wave-equation solution is found simultaneously with that of the energy-flow equation in order to find the temperature profile as well as the acoustic pressure. The energy flow through stacks is determined by temperatures and/or heat flows at adjacent heat exchangers.

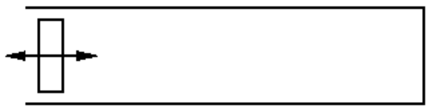


Figure I.1: Driven, lossy plane-wave resonator.

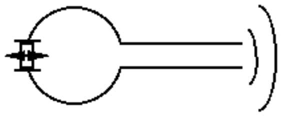


Figure I.2: Driven, radiating Helmholtz resonator.

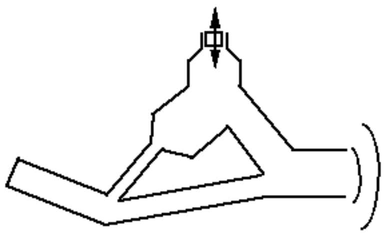


Figure I.3: Duct network.

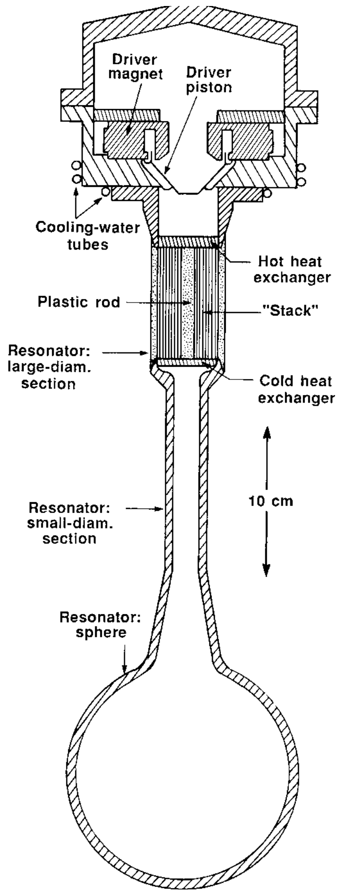


Figure I.4: Thermoacoustic refrigerator (Hofler style).

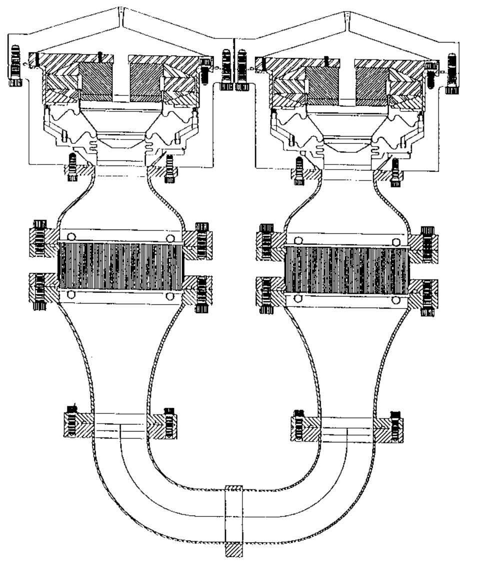


Figure I.5: Thermoacoustic refrigerator (TALSR style).

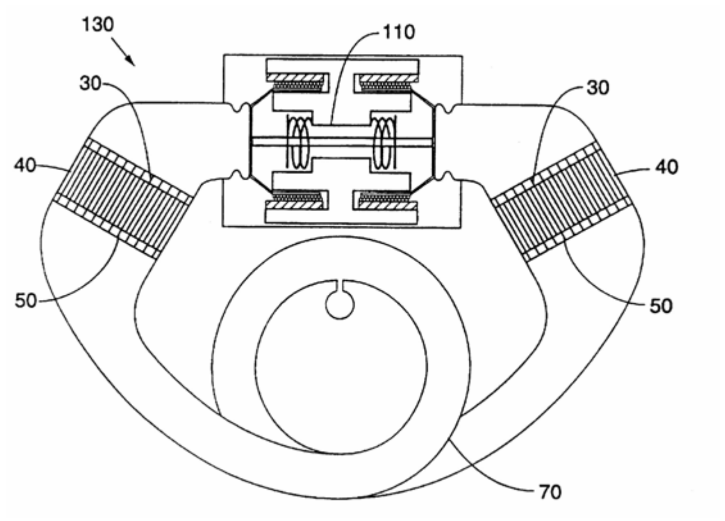


Figure I.6: Thermoacoustic refrigerator (Garrett and Hofler style).

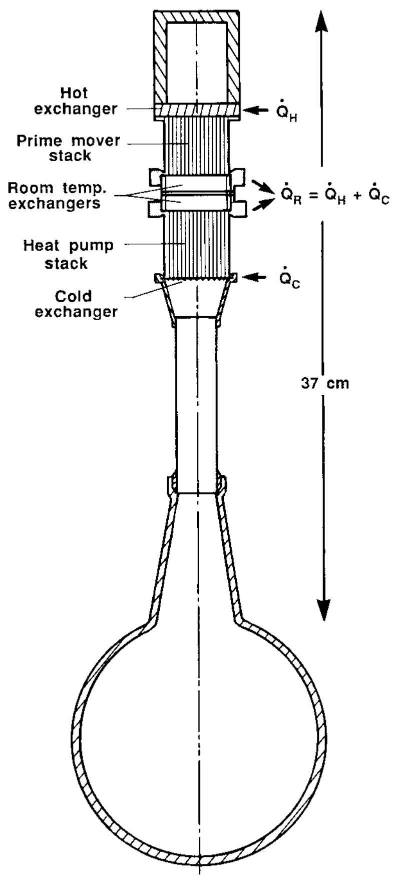


Figure I.7: So-called "beer cooler."

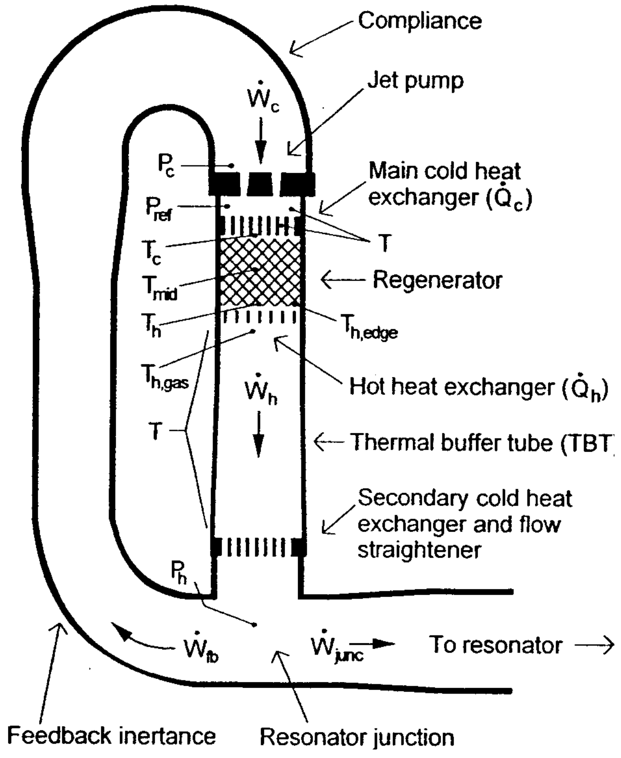


Figure I.8: Thermoacoustic Stirling hybrid engine.

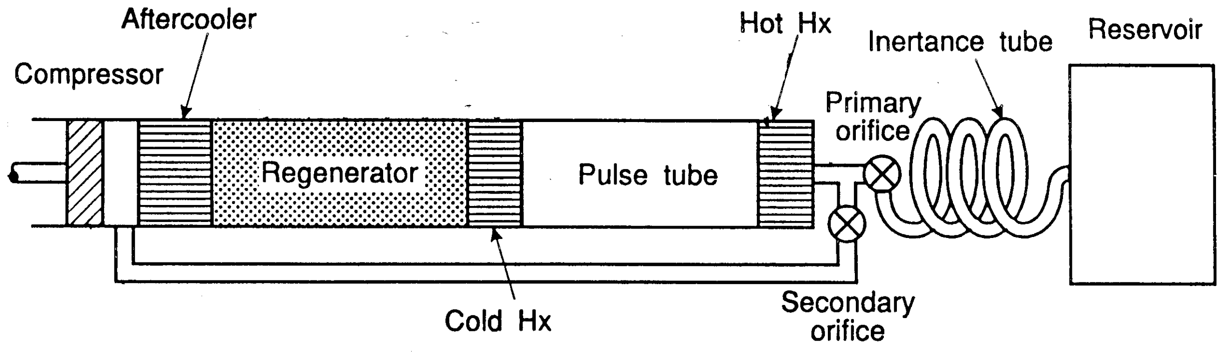


Figure I.9: Double-inlet pulse-tube refrigerator.

The user of DeltaE enjoys considerable freedom in choosing which variables are computed as 'solutions.' For example, in a simple plane-wave resonator (the first example below), DeltaE can compute the input impedance as a function of frequency, or the resonance frequency for a given geometry and gas, or the length required to give a desired resonance frequency, or even the concentration in a binary gas mixture required to give a desired resonance frequency in a given geometry.

Generally, DeltaE does not include any nonlinear effects that arise at high amplitudes, so be cautious using it when Mach numbers or Reynolds numbers are too high. The principal exception to this rule is the optional turbulence algorithm in ducts, discussed in Chapter V. There are a number of other approximations used, which will be discussed below as we encounter them, and in more detail in Chapter VI.

B. How This Manual is Organized

We will teach the use of DeltaE by increasingly complicated examples in Chapters II–IV. Chapter II is just acoustics, without thermoacoustics. It serves to introduce DeltaE's input/output formats and editing and plotting features. Chapter III gives the most complete discussion of the overall principles behind the thermoacoustics computations, and the simplest thermoacoustic examples. The agreement of these examples with published experimental data serves as validation of the code. In Chapter IV, features of DeltaE which are useful in modeling Stirling cycle and more general devices are introduced. Chapter V gives the most in-depth discussion of the advanced options of DeltaE.

Chapter VI is a segment-by-segment reference chapter for the experienced user, summarizing assumptions built into the computations for each segment, the data format for each segment, and thermophysical properties. It is our hope that experienced users can quickly find the information they need in Chapter VI, while new users will find the wordier explanations of the earlier chapters helpful.

The examples we've included are simpler than DeltaE files we use in our own research. We've maintained this simplicity in the User's Guide to avoid clutter. Experienced users will find that the number of segments in their DeltaE files grows and grows, as small effects and non-standard results are included. Some of these examples were run on an MS-DOS machine, others on a Mac. (While the menu interface differs, the file formats and displays for both platforms are the same. When there are differences, they will be obvious.) Early versions of DeltaE were still being debugged and improved while these examples were being run, so there may be some minor errors and formatting oddities in these examples.

We assume that the reader of this manual is very comfortable with linear acoustics and reasonably familiar with thermoacoustics. We will use variables as defined, for example, in the list of symbols in Refs. [1] or [2].

II. Basic Acoustics

In this Chapter we use the simplest acoustic segments, such as ducts and endcaps, to introduce DeltaE's basic features.

A. Plane-Wave Resonator

We begin with a lossy plane-wave resonator, driven from one end by a piston with a fixed volume flow rate. We call the input file `planewav.in` (included in the `examples` directory or folder). This input file could have been created from scratch using any text editor, though this one was made by editing one of DeltaE's own output files. (N.B.: A DeltaE input file must always be a plain text file, in the native text format of the machine it is running on. On some systems, integer numbers must be followed by a decimal point, as in the example below. Also, some systems require the last line in the file to be followed by an end-of-line character, before the end-of-file character occurs.)

```
TITLE     Example 1:  Plane-wave resonator

BEGIN      Initialize things
1.000E+05 a Mean P    Pa
100.      b Freq.     Hz
300.      c T-beg      K
1000.     d |p|@0     Pa
90.       e Ph(p)0    deg
1.000E-02 f |U|@0    m^3/s
000      g Ph(U)0    deg
helium

ENDCAP     First end
1.000E-02 a Area     m^2
helium

DUCT    The heart of the matter
1.000E-02 a Area     m^2
0.354     b Perim     m
5.00      c Length    m
helium

ENDCAP     Second End
1.000E-02 a Area     m^2
helium

HARDEND
```

```
000     a R(1/z)
000     b I(1/z)
helium

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
INVARS      2  0  4  0  5
TARGS       2  4  1  4  2
SPECIALS    0
```

Several features of DeltaE input files are illustrated here. Input files should be named something`.IN` or something`.OUT`. These files consist of a set of segments whose order and format are important. The initial (or 'zeroth') segment is always the `BEGIN` segment, and the last segment is usually `HARDEnd` (or `SOFTEnd` to be discussed in Chapter VI). Intervening segments describe the geometry and other properties of the acoustic engine. The number and order of data in each segment is crucial. All units are MKS. Within each line, only the first number (e.g., "`1.e5`" or "`100.`") or word (e.g., "`helium`" or "`BEGIN`") is important; the rest of the line can be used as a comment field, with, for example, the units or name of the variable whose value appears. Whole-line comments can appear anywhere if they begin with "!" or with 20 or more blanks. Numbers can be in fixed or exponential format. Segment names are all uppercase, and only the first five characters are interpreted (hence, the convention here is to write segment names longer then 5 characters with trailing lower case letters, e.g., `HARDEnd`). All words must have correct CASE and spelling.

The file shown below works just as well in the computer as the one shown above. However, with fewer comment annotations and only the minimal 5-character segment names, it is harder for humans to follow; it also lacks restart information, so DeltaE might have to prompt us for some more information before proceeding.

```
TITLE

BEGIN
1.e5
100.
300.
1000.
90.
1.0e-2
0
helium

ENDCA
0.01
helium

DUCT
0.01
0.354
5.00
helium

ENDCA
0.01
helium
```

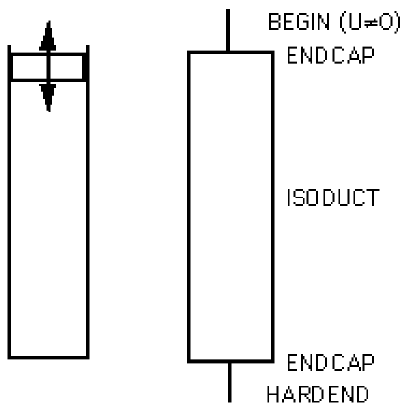


Figure II.1: A plane-wave resonator; conventional and DeltaE representation.

```
HARDE
0
0.
helium
```

BEGIN sets the stage, in this case, with 1-bar room temperature helium gas being driven at 100 Hz with a pressure amplitude of 1000 Pa and a volume flow rate amplitude of 0.01 m^3/s, 90° out of phase with the pressure.

Since BEGIN has no geometrical properties, an ENDCAp comes next to account for oscillatory thermal losses at the first end of the resonator. An endcap is just a surface area giving dissipation. In this example, because we are beginning with a nonzero volume flow rate, ENDCAp can be imagined as the face of the moving piston.

A lossy duct DUCT comes next. Here, we have made the perimeter $\sqrt{4\pi \times \text{area}}$, to make this a circular duct.

The resonator ends with another ENDCAp for oscillatory pressure dissipation.

The input file then ends with the HARDEnd segment. Its two lines are the real and imaginary parts of the inverse of the end impedance, $1/Z = U_1/p_1$. Since we have set these two equal to zero, this is simply a solid immovable boundary, with zero volume flow rate.

Note that the special segments BEGIN and HARDEnd have no geometrical properties; so ENDCAps are needed to put the thermal dissipation loss at the ends of the resonator.

Figures II.1 show the acoustician's usual cartoon of a driven plane-wave resonator and a pictorial representation of how we modeled this resonator for DeltaE. Throughout this tutorial we use generally conventional symbols to represent ordinary segments such as ducts, horns, and heat exchangers.

This input file overdetermines the acoustic system because only some of the variables listed can be specified independently. You can choose which of these variables DeltaE will regard as fixed, which it will regard as initial guesses at solution values, and, occasionally, which it will ignore.

Execute DeltaE and respond `planwave` to the prompt for an input file (on a MacIntosh, double-click `planwave.in`, or open it using the "New Model" menu). You may also type `deltae planwave`, or `deltae planwave.in`, to load the file directly. After your required "carriage return to continue," DeltaE will respond with:

```
Loading planwave.in . . .
Example 1:  Plane-wave resonator
Ready.
```

DeltaE can accept either `.in` or `.out` files as input files. If you do not type the file suffix, DeltaE looks first for a `.out` file. If it does not find it, it looks next for a `.in` file; if this is not found, DeltaE reprompts for the file name. (On a MacIntosh, all of these steps are replaced by a standard file selection dialog.)

On a keyboard menu system (e.g., PC-compatible, VMS, Unix, etc.), the main menu is displayed next, giving the following options:

```
Main Menu:
   r  (r)un model                    p      (p)lot another parameter
   w  (w)rite current model state    P      (P)lot status summary
   n  (n)ew model input file         c      (c)lear from vectors and plots
   R  (R)estore vectors              C      (C)lear|set all guesses&targets
   E  (E)xtras                       u      (u)se in guess/target vector
   d  (d)isplay                      v      (v)ector status summary
   o  (o)utput to printer            m      (m)odify parameter value
   f  (f)orm feed printer            s      (s)pecial modes editing
   D  (D)OS command shell            t      (t)hermophysical properties
   I  (I)nsert segment               K      (K)ill segment
   e  (e)xit DeltaE                  ?      show this menu

MAIN:  (rpwPncRCEudvomfsDtIKe?)>
```

(On the MacIntosh, similar options are available on the menu bar for mouse selection.)

Now select "vector status summary" by typing "v":

```
Iteration Vectors Summary:
GUESS      0d         0e
name  BEGIN:|p|   BEGIN:Ph(p)
value    1.00E+03     90.
units     Pa          deg

TARGET      4a          4b
name  HARDE:R(1/z HARDE:I(1/z
units
```

```
value     .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
```

The GUESS vector, which has two components in this example, shows what DeltaE will regard as solution variables: the magnitude and phase of the beginning pressure. Their initial guesses are also shown. (This particular two-dimensional guess vector came from the computer-generated table at the bottom of the input file. This table is explained more thoroughly in Chapter III, especially in Sec. III C. If your input file has no such table, DeltaE can make a reasonable guess at the guess variables you might want to set when you select `(C)lear|set all guesses&targets` with no guesses defined yet.)

Basically, DeltaE integrates the wave equation from BEGIN to END. We insist that DeltaE refine the two-dimensional GUESS vector to find a solution to this acoustics problem that arrives at the HARDEnd with zero complex volume flow rate. This is accomplished by getting the '0' values of the real and imaginary parts of the inverse of the impedance in the HARDEnd segment into DeltaE's two-dimensional TARGET vector, as shown in this vector summary table.

The last two lines indicate unselected, still-available targets. These are the only remaining output values for which DeltaE has a reserved input variable available to compare with it. In this model, all such outputs are already in use.

Most of the thought required to successfully run DeltaE occurs while trying to figure out which of the variables are appropriate guesses and targets. While the choice of these variables is almost entirely arbitrary, as long as the number of guesses equals the number of targets, some choices for the guess vector would be physically nonsensical. For example, allowing DeltaE to try to achieve resonance in a given length by varying the areas of the endcaps would be futile. For the examples in subsequent Chapters, the choice of good guess and target vector members is not always as obvious as it is here.

For now, we will keep these vectors. "Run" the code (type 'r'), and "(e)xit" from DeltaE to the operating system to inspect its results, which DeltaE has put in two new files, `planewav.dat` and `planewav.out`. `planewav.dat` looks like this:

```
 -= Example 1:  Plane-wave resonator                                       =-
 frequency=     100.000Hz     mean pressure=   1.000E+05Pa

   Tm(K)        Re & Im p1(Pa)      Re & Im U1(m^3/s)       Hdot(W)   Edot(W)
   300.0      2910.6   2553.9      0.01000  0.00000          14.55     14.55
!----------------------------------  1 ----------------------------------
 ENDCAP    First end
 Heat extracted:   7.333E-02 Watts
   300.0      2910.6   2553.9      0.00997 -0.00002          14.48     14.48
!----------------------------------  2 ----------------------------------
 DUCT   The heart of the matter
 Duct wavvec =(   0.623    ,  -6.207E-03) m^-1
 Heat extracted:    14.5     Watts
```

```
   300.0     -2908.0  -2556.2     -0.00003 -0.00002           0.07      0.07
!------------------------------------  3 ------------------------------------
 ENDCAP    Second End
 Heat extracted:   7.331E-02 Watts
   300.0     -2908.0  -2556.2      0.00000  0.00000           0.00      0.00
!------------------------------------  4 ------------------------------------
 HARDEND   Final
 inverse impedance (rho a U/p A)=(  -5.863E-10,   4.052E-10)

   300.0     -2908.0  -2556.2      0.00000  0.00000           0.00      0.00
```

Examination of `planewav.dat` will show that the solution is $\Re(p) = 2911$ Pa, $\Im(p) = 2554$ Pa. Also shown are temperature, complex p_1, and complex U_1, acoustic power flow, and energy flow at the beginning and end of each segment. You can see, for instance, that the driver delivers 14.55 Watts of power to the resonator, that 0.07 Watts is absorbed on each end, and that 14.4 Watts is absorbed by the duct. The acoustic power absorbed in isothermal segments such as these is extracted as heat, *e.g.* by a water jacket in the real world.

The output model file, `planewav.out`, is shown below:

```
TITLE     Example 1:  Plane-wave resonator
!---------------------------------  0 --------------------------------
 BEGIN      Initial
  1.0000E+05 a Mean P    Pa             3972.2    A |p|    G( 0d)     P
   100.0     b Freq.     Hz               41.266  B Ph(p)  G( 0e)     P
   300.0     c T-beg     K
   3872.2    d |p|@0     Pa     G
   41.266    e Ph(p)0    deg    G
  1.0000E-02 f |U|@0    m^3/s
  0.0000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
!---------------------------------  1 --------------------------------
 ENDCAP     First end
  1.0000E-02 a Area      m^2            3972.2    A |p|       Pa
                                          41.266    B Ph(p)     deg
                                         9.9721E-03 C |U|     m^3/s
                                        -0.1407     D Ph(U)     deg
                                          14.481    E Hdot      W
 helium     Gas type                      14.481    F Edot      W
 ideal      Solid type                  -7.1895E-02 G HeatIn    W
!---------------------------------  2 --------------------------------
 DUCT    The heart of the matter
  1.0000E-02 a Area      m^2            3871.8    A |p|       Pa
  0.3540     b Perim      m             -138.68    B Ph(p)     deg
   5.000     c Length     m             3.7130E-05 C |U|     m^3/s
                                        -138.68    D Ph(U)     deg
                                        7.1879E-02 E Hdot      W
 helium     Gas type                    7.1879E-02 F Edot      W
 ideal      Solid type                  -14.409    G HeatIn    W
!---------------------------------  3 --------------------------------
 ENDCAP     Second End
  1.0000E-02 a Area      m^2            3871.8    A |p|       Pa
                                        -138.68    B Ph(p)     deg
                                        1.6873E-10 C |U|     m^3/s
                                          6.6665   D Ph(U)     deg
                                       -2.6871E-07 E Hdot      W
 helium     Gas type                   -2.6871E-07 F Edot      W
 ideal      Solid type                -7.1879E-02 G HeatIn    W
!---------------------------------  4 --------------------------------
 HARDEND    Final
  0.0000     a R(1/z)         = 4G?      3871.8    A |p|       Pa
  0.0000     b I(1/z)         = 4H?     -138.68    B Ph(p)     deg
                                         1.6873E-10 C |U|     m^3/s
                                            6.6665  D Ph(U)     deg
                                        -2.6871E-07 E Hdot      W
                                        -2.6871E-07 F Edot      W
                                        -5.8630E-10 G R(1/z)
 helium     Gas type                     4.0521E-10 H I(1/z)
 ideal      Solid type                     300.0    I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      2  0  4  0  5
TARGS       2  4  1  4  2
SPECIALS    0
```

(Some digits of lesser significance in DeltaE output examples in this manual may vary from the numbers that you get running your version of the code. This is primarily due to differences in floating point arithmetic hardware and software between different compilers and computers, and the finite tolerance against which DeltaE measures the accuracy of its results. This can be reduced from the default (see Chapters V and VI for details) to force iterations to continue until a greater degree of precision is achieved. For tighter tolerance levels, all significant digits in the finished output file will agree for all versions of the code with relatively straightforward models.)

Examination of `planewav.out` will show that it is a slightly modified version of our `planewav.in`: It includes the solution values for magnitude and phase of beginning pressure (3972 Pa. 41.27°), replacing our original guesses (1000 Pa. 90°). DeltaE would have made this file as it is even if we had used the bare-bones, unannotated version of the input file. In `*.out` files, DeltaE numbers the segments and 'letters' the lines in each segment for our convenience, and displays names and units for all variables. It adds the obscure table at the end that reflects our choice of guess and target vectors. The format of DeltaE's `.out` file is acceptable as an input file, so using it as such saves the user a lot of work.

As a new example, we will find the resonance frequency f, which we guess is near 100 Hz. We'll use the same old `planewav.in`, so execute DeltaE again and select it. Display the vector status summary again.

```
Iteration Vectors Summary:
GUESS      0d          0e
name  BEGIN:|p|   BEGIN:Ph(p)
value    1.00E+03     90.
units     Pa          deg
TARGET      4a          4b
name  HARDEND  :R(1/z HARDEND  :I(1/z
units
value     .00         .00
```

Now we want f, $|p|$ in segment `BEGIN` as the 2 components of the guess vector. We will fix the phase of the beginning p_1 at 0, because having p_1 and U_1 in phase at the driver is the condition for resonance. So we want to change this table to look like this:

```
Iteration Vectors Summary:
GUESS      0b          0d
name  BEGIN:Freq. BEGIN:|p|
value    1.00E+02    1.00E+03
units    Hz         Pa
TARGET      4a          4b
name  HARDEND  :R(1/z HARDEND  :I(1/z
units
value     .00         .00
```

We can make this change in guess vector by a "`(c)lear`" of `0e` from the guess vector; "`(u)se`" `0b` instead; and "`(m)odify`" `0e` to be zero degrees instead of 90°. (These vectors happen to be identical to DeltaE's default, so we could have generated this table by selecting `(C)lear|set` twice—first to wipe out the old vectors, and then again to set the defaults.) Now, "`(r)un`" the calculation. Inspect the results by using "`(d)isplay`" from within DeltaE (or by escaping to the operating system, as you did before). You will find that the resonance frequency is 100.9 Hz.

If you can't remember the number-letter code for the variable you want when modifying the vectors, "`(d)isplay`" all segments, or "`(d)isplay`" a selected segment number to see a list of the variables. For example, "`(d)isplay`" segment 0 to find out which number-letter code is for frequency:

```
INPUT     # ParType Units  Status       OUTPUT    # ParType Units  Status
---------------------------------  0 ---------------------------------
BEGIN      Initialize things
1.000E+05 a Mean P    Pa                .000      A |p|       0d       P
100.      b Freq.     Hz                .000      B Ph(p)     0e       P
300.      c T-beg     K
1.000E+03 d |p|@0     Pa      G
90.0      e Ph(p)0    deg     G
1.000E-02 f |U|@0    m^3/s
00.0      g Ph(U)0    deg
helium     Gas type
ideal      Solid type
```

If you incorrectly remember a number-letter code and are stuck in a modification, you can either type "return" repeatedly to accept existing values, or type "x" to escape. (If you're really stuck, control-C will escape from nearly anywhere.)

DeltaE can use any physically appropriate variables in the guess vector. You can determine what temperature makes the system resonate at 100 Hz, by putting `0c` in the guess vector. (The answer is 290.7 Kelvin.) Or, by putting `2c` in the guess vector, we could have found out what length the duct needs to be to put the resonance at 100 Hz at 300 K. An advanced feature to be discussed in Chapter V allows use of the concentration in a binary gas mixture to be used (as a member of the guess vector) so that we could determine the argon concentration that would be required in the helium to make the resonance at 100 Hz.

B. Plotting

DeltaE allows for plotting by automatically incrementing (or decrementing) one or two independent variables, and tabulating these together with one or more output variables in a file named something`.plt`. Users can then manipulate and/or plot that file with their favorite graphics or spreadsheet software. We illustrate these features with a continuation of the same example, a plane-wave resonator.

We use the same input file as before, `planewav.in`. Execute DeltaE and choose this as input file. Check the vector status summary:

```
Iteration Vectors Summary:
 GUESS      0d          0e
 name  BEGIN:|p|   BEGIN:Ph(p)
 value    1.00E+03     90.
 units     Pa          deg
TARGET      4a          4b
 name  HARDEND  :R(1/z HARDEND  :I(1/z
 units
 value     .00         .00
```

Now inspect the Plotted parameter summary (type capital "P"):

```
Dependent Variables (outputs):
PLOTS     0A        0B
 name  BEGIN:|p|   BEGIN:Ph(p)
 units    Pa        deg
```

Keep these parameters as the dependent variables to be plotted (they are copies of the guesses). To set up the independent variables, select "`(p)lot another parameter`." We will make a two-dimensional plot, letting f go from 80 Hz to 339.5 Hz in 1.5-Hz steps in the inner loop, and using two values of mean pressure—1000 Pa and 100,000 Pa—in the outer loop. DeltaE prompts for these entries in the "range" selection. As before, if you can't remember the segment-number and line-letter codes for frequency and mean pressure, use "`(d)isplay`" to find out. After entering these values, check the Plotted parameter summary again:

```
Dependent Variables (outputs):
PLOTS     0A        0B
 name  BEGIN:|p|   BEGIN:Ph(p)
 units    Pa        deg
Indpendent Variables (inputs):
```

```
Outer loop:   0a  BEGIN:Mean  Beg=  1.00E+03 End=  1.00E+05 Step=  9.90E+04
Inner Loop:   0b  BEGIN:Freq. Beg=   80.     End=  3.40E+02 Step=   1.5
```

Before beginning, we must also `(m)odify` the inital pressure amplitude (`0d`) to a value that's reasonable for 1,000 Pa mean pressure. We choose 10 Pa: one percent.

Now do a `(r)un`. DeltaE will step through the variables selected (taking a minute or two on a 286, but much less on a newer computer). When it has finished, exit to the operating system, and find two new files. The file `planewav.des` gives headings of what has been tabulated:

```
BEGIN:Mean  BEGIN:Freq. BEGIN:|p|   BEGIN:Ph(p)
     Pa          Hz         Pa         deg
     0a          0b         0A         0B
```

and `planewav.plt` is the table of values:

```
   1000.        80.00        3.247        46.93
   1000.        81.50        3.543        43.67
   1000.        83.00        3.861        39.76
   1000.        84.50        4.194        35.11
   1000.        86.00        4.529        29.63
     .
     .
     .
 1.0000E+05   87.50       -370.7        265.4
 1.0000E+05   86.00       -328.6        265.8
 1.0000E+05   84.50       -293.7        266.0
 1.0000E+05   83.00       -264.1        266.3
 1.0000E+05   81.50       -238.6        266.4
 1.0000E+05   80.00       -216.4        266.6
```

Notice that DeltaE alternates the order in which it calculates the points of the inner loop (frequency, here). This process is motivated by the quality of initial guesses; 'zig-zagging' thus, DeltaE must spend only a brief time searching for the start point of the inner loop each time it increments the outer loop.

We brought this file into a spreadsheet/graphics program to fix it up for plotting. We removed minus signs from $|p_1|$ whenever they occurred, adding 180° to phase(p_1) in those cases to improve the looks of the the graphs. We also plotted $|p_1|/p_m$ (instead of just $|p_1|$). The resulting plots are shown in Figs. II.2. (The lower quality-factor curves are for the lower mean pressure, of course.)

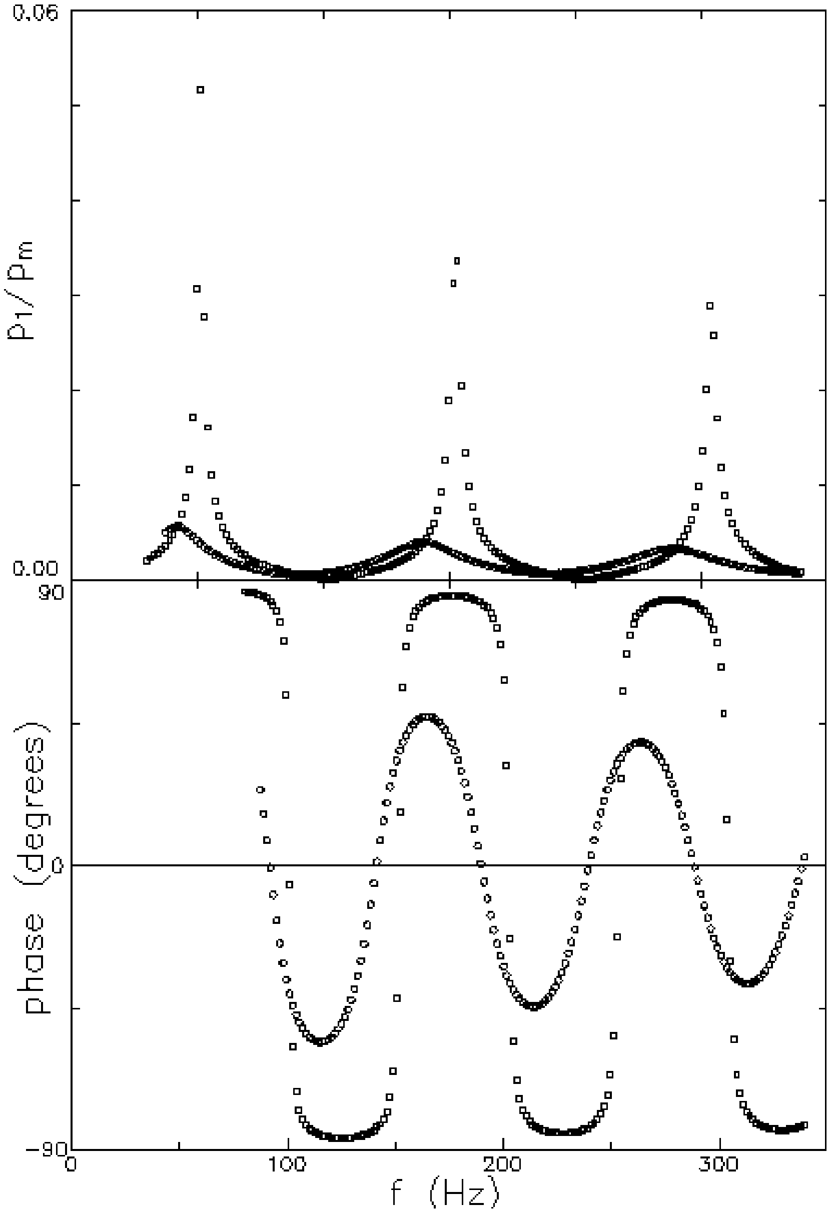


Figure II.2: Pressure and phase vs frequency for the plane-wave resonator.

C. Further Simple Features

Here we describe some additional DeltaE features which are relevant to purely acoustic (not thermoacoustic) apparatus. A list of the most commonly used purely acoustic segment types (including those introduced previously) is given below. More detailed descriptions of each are given in Chapter VI.

`TITLE` Required at the top; comment field is retained in all `.DAT` and `.OUT` files.

`BEGIN` Required immediately after `TITLE`. This is the "zeroth" segment. It defines global parameters such as mean pressure and frequency, and initial conditions for p_1 and U_1.

`ENDCAp` A surface area with oscillatory-pressure loss in its thermal penetration depth. Usually used at ends of ducts.

`DUCT` A duct, with viscous and thermal losses at the wall. Separate entry of area and perimeter accommodates ducts of any cross-sectional shape.

`CONE` A cone to adapt between ducts of different sizes. Uses lossy Webster horn equation.

`COMPLiance` A compliance. With oscillatory-pressure losses on surface.

`IMPEDance` A lumped-parameter series impedance.

`IDUCEr` and `VDUCEr` Current-driven and voltage-driven transducers, with parameters independent of frequency.

`ISPEAker` and `VSPEAker` Current-driven and voltage-driven electrodynamic transducers, parameterized by mass, B-L product, etc., so that impedance coefficients depend on frequency.

`IEDUCer` and `VEDUCer`, `IESPEaker` and `VESPEaker` The four transducers above are in *branched* configurations, where pressure is unchanged by the transducer and the "back side" of the transducer hangs outside of DeltaE's computation space. These Enclosed versions are their *series* counterparts, with one side of the transducer facing the previous segment and the other side facing the subsequent segment, so that the volume flow rate remains constant across the segment.

`BRANCh` A frequency-independent side-branch impedance.

`OPNBRanch` A frequency-dependent side-branch impedance with the characteristic radiation impedance of a duct opening into an infinite or semi-infinite space.

`HARDEnd` One of the allowed final segments. Used to set $U_1 = 0$ through use of the inverse of the acoustic impedance in the `TARGET` vector.

`SOFTEND` The other allowed final segment. Used to set p_1= 0 through use of the acoustic impedance in the `TARGET` vector. Very useful for defining a mirror-image plane in symmetric apparatus with a pressure node at a center of symmetry.

Each segment must have a gas type and a solid type (even some segments that don't actually use such information, such as `BRANCh`). At present, DeltaE supports air, helium, neon, He-Xe, He-Ar, and He-Ne mixtures (see Chapter VI), hydrogen, deuterium, nitrogen, carbon dioxide, natural-gas combustion products (i.e., chimney exhaust), liquid sodium, and eutectic liquid NaK as gases. Solids include Kapton, mylar, stainless steel, molybdenum, tungsten, copper, nickel, and ideal. An ideal solid is one that has essentially infinite heat capacity, density, and thermal conductivity. If no solid type is given in the input file, DeltaE will assign the ideal solid type. There is also a mechanism for specifying additional, user-defined fluids and solids; details are given in Chapter V.

The `sameas nl` feature allows reference to a value (either a number or a gas or solid type) in another segment. This helps prevent typographical errors in the input file, and is especially useful in linking dimensions of adjacent segments which you might want to vary all together while modifying or plotting, such as areas of all segments when increasing the size of the apparatus. You can give just the segment number, if the parameter letters are the same (e.g., "`sameas 0`" is often the gas type in all segments after the zeroth segment), or segment number and line letter (e.g., "`sameas 3a`"). If you try to use `sameas` for two different types of variables—making a length the same as an area, for example—DeltaE will give an error message. The following example is for a closed resonator composed of two ducts joined by a cone:

```
TITLE    illustrating use of sameas
!---------------  0 ------------------------
BEGIN      Initial
  1.380E+06 a Mean P    Pa
   132.     b Freq.     Hz
   586.     c T-beg     K
  6.639E+04 d |p|@0     Pa
   .000     e Ph(p)0    deg
   .000     f |V|@0    m^3/s
   .000     g Ph(V)0    deg
helium     Gas type
!---------------  1 ------------------------
ENDCAP     First End
sameas 2a   a Area     m
sameas  0  Gas type
!---------------  2 ------------------------
DUCT       First Duct
1.292e-2    a Area     m
   .403     b Perim    m
1.0         c Length   m
sameas  0  Gas type
!---------------  3 ------------------------
CONE        adapter between ducts
sameas 2a   a AreaI   m^2
sameas 2b   b PerimI   m
   .100     c Length   m
sameas 4a   d AreaF   m^2
sameas 4b   e PerimF   m
sameas 0
!--------------- 4 ------------------------
DUCT       Second Duct
0.323e-2  a Area      m^2
0.2015    b Perim      m
1.0       c Length     m
sameas 0
!--------------- 5 ------------------------
ENDCAP     Second End
sameas  4a  a Area    m^2
sameas 0
!--------------- 6 ------------------------
HARDEND
   .000     a R(1/z)
   .000     b I(1/z)
sameas 0
```

When you access a parameter specified by `sameas` using `(m)odify`, or `(p)lot` to make it an independent plot variable, or `(u)se` it in a guess vector, the `sameas` relationship is severed and the parameter is given its current actual value. This is required because the value at this point will be changed (either by you, or by DeltaE). But if a variable that is the root of several `sameas` references is caused to change in any of these three ways, all `sameas` references to this root within the model will change with it.

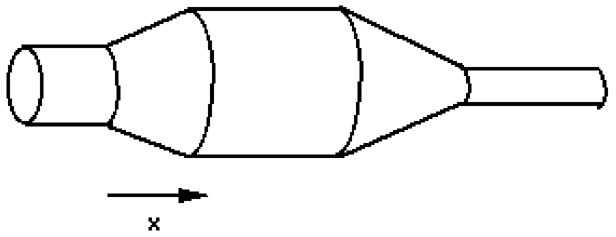
III. Thermoacoustics

The examples given in the previous Chapter were relatively simple cases of linear acoustics. In this Chapter, we introduce the full thermoacoustic computing power of DeltaE. After discussing the principles and assumptions that are built into the computation, we present example calculations.

A. Principles of Computations

DeltaE deals with one-dimensional strings of acoustic and thermoacoustic elements (see Chapter V for branches), so the one-dimensional "wave" equation is of the greatest importance. We always assume a time dependence of $e^{i\omega t}$, so the "wave" equation is the second-order Helmholtz differential equation for the complex pressure amplitude $p_1(x)$:

$$p_1 + \frac{a^2}{\omega^2}\frac{d^2p_1}{dx^2} = 0. \tag{III.1}$$



It is sometimes easier to think of this second-order equation as two coupled first-order equations in pressure p_1 and volume flow rate U_1 :

$$\begin{aligned}\frac{dp_1}{dx} &= -\frac{i\omega\rho}{A}U_1,\\ \frac{dU_1}{dx} &= -\frac{i\omega A}{\rho a^2}p_1.\end{aligned} \tag{III.2}$$

This point of view is taken in Refs. [3] and [2]. In this form, the equations are ready for numerical integration along x.

More complexity is added, as needed, for given geometries. For example, in boundary-layer approximation in a duct or shallow cone, the governing equations are

$$\begin{aligned}\frac{dp_1}{dx} &= -\frac{i\omega\rho}{A}\left[1-\frac{1-i}{2}\frac{\Pi}{A}\delta_\nu\right]^{-1} U_1,\\ \frac{dU_1}{dx} &= -\frac{i\omega A}{\rho a^2}\left[1+\frac{1-i}{2}\frac{\Pi}{A}\frac{\gamma-1}{1+\epsilon_s}\delta_\kappa\right] p_1. \qquad \text{(III.3)}\end{aligned}$$

where A is the cross-sectional area, Π is the perimeter, and ϵ_s is a correction for thermal properties of the solid wall that is usually negligible. In a stack, we use Rott's wave equation:

$$(1+\frac{(\gamma-1)f_\kappa}{1+\epsilon_s})p_1+\frac{\rho_m a^2}{\omega^2}\frac{d}{dx}(\frac{1-f_\nu}{\rho_m}\frac{dp_1}{dx})-\beta\frac{a^2}{\omega^2}\frac{(f_\kappa-f_\nu)}{(1-\sigma)(1+\epsilon_s)}\frac{dT_m}{dx}\frac{dp_1}{dx}=0, \qquad \text{(III.4)}$$

which can be rewritten as two coupled first-order equations:

$$\begin{aligned}\frac{dp_1}{dx} &= -\frac{i\omega\rho_m}{A}(1-f_\nu)^{-1}U_1,\\ \frac{dU_1}{dx} &= -\frac{i\omega A}{\rho_m a^2}\left(1+\frac{(\gamma-1)f_\kappa}{1+\epsilon_s}\right)p_1+\frac{(f_\kappa-f_\nu)}{(1-\sigma)(1+\epsilon_s)}\beta\frac{dT_m}{dx}U_1. \qquad \text{(III.5)}\end{aligned}$$

In DeltaE, the computation uses the wave equation that is appropriate for each segment. DeltaE uses continuity of p_1 and U_1 to pass from the end of one segment to the beginning of the next. Within each segment, wave propagation is controlled by local parameters such as area and perimeter. Although DeltaE uses analytic solutions to the wave equation for some of the simplest segment types, it usually must integrate the wave equation numerically, so it is generally correct to imagine DeltaE beginning at the `BEGIN` segment and numerically integrating "the wave equation" through each segment, in turn, to the end of the list, using local parameters, such as area and perimeter, as it goes.

It is clear that the solution $p_1(x)$, $U_1(x)$ is only determined uniquely if four real boundary conditions are imposed, because the governing equation can be expressed as two coupled first-order equations in two complex variables or as a single second order equation in one complex variable. This is true whether considering a single segment or a one-dimensional string of segments with each joined to its neighbor(s) by continuity of p_1 and U_1. If all four boundary conditions are given at one end of the apparatus (i.e., if we know the complex p_1 and complex U_1 at the `BEGIN` segment) then the integration is utterly straightforward. But usually some of the boundary conditions are given elsewhere. For example, in the plane-wave resonator in the previous Chapter, the boundary conditions were $U_1 = (0.01, 0)$ m/s at the `BEGIN` segment, and $U_1 = (0, 0)$ at the `HARDEnd`. In such conditions DeltaE uses a shooting method,[1] by guessing any unknowns among the four numbers defining p_1 and U_1

[1]Precisely speaking, DeltaE forms a system of nonlinear equations from the model using the targets that the user selects and manipulates the guesses to drive the differences between the targets and results to zero. The routine incorporated in the code is called `DNSQ`, and it is part of the SLATEC Common Mathematical Library, which is freely available through the internet software repository at "http://www.netlib.org." The algorithm used is a modification of the Powell hybrid method.

at the `BEGIN` segment, integrating to the `HARDEnd`, comparing the results with the boundary conditions imposed at the `HARDEnd`, and adjusting its guesses until it comes out right.

The boundary conditions imposed at the `HARDEnd` are in DELTAE's `TARGET` vector. The unknown conditions at the `BEGIN`ning, which DELTAE is supposed to find, are in DELTAE's `GUESS` vector. The number of elements in the `TARGET` vector must equal the number of elements in the `GUESS` vector; otherwise the system is over- or under-determined.

One of DELTAE's most powerful features is that the elements of the `GUESS` vector are not limited to the conventional choices consisting of real and imaginary parts of p_1 and U_1 at the `BEGIN`ning. Any variables that have an effect on the `TARGET` variables can be used. This enables DELTAE to calculate resonance frequencies, geometrical dimensions, temperatures, or even concentration in binary gas mixtures in order to satisfy given boundary conditions.

To add thermoacoustic computation ability to this linear acoustic picture, only one more equation is required, i.e., that for the temperature $T_m(x)$. As for $p_1(x)$ and $U_1(x)$, the equation for $T_m(x)$ depends on the type of segment, and continuity of $T_m(x)$ is imposed at the junctions between segments. Most segments, such as ducts and cones, obey simply $dT_m/dx = 0$. Stacks have a more complicated, but still only first-order, differential equation for $T_m(x)$, Rott's energy equation:

$$\begin{aligned}\dot{H}_2 &= \frac{1}{2}\Re\left[p_1\tilde{U}_1\left(1-\frac{f_\kappa-\tilde{f}_\nu}{(1+\epsilon_s)(1+\sigma)(1-\tilde{f}_\nu)}\right)\right] \\ &+ \frac{\rho_m c_p|U_1|^2}{2A_{\text{fluid}}\omega(1-\sigma)|1-f_\nu|^2}\frac{dT_m}{dx}\Im\left[\tilde{f}_\nu+\frac{(f_\kappa-\tilde{f}_\nu)(1+\epsilon_s f_\nu/f_\kappa)}{(1+\epsilon_s)(1+\sigma)}\right] \\ &- (A_{\text{fluid}}K+A_{\text{solid}}K_s)\frac{dT_m}{dx} \qquad \text{(III.6)}\end{aligned}$$

So, for thermoacoustic calculations, DELTAE integrates from `BEGIN`ning to `HARDEnd`, with respect to five real variables: real $T_m(x)$, complex $p_1(x)$, and complex $U_1(x)$. It uses the appropriate wave equation and temperature equation for each segment. Within each segment, wave propagation is controlled by local parameters, such as area and perimeter, and by global parameters, such as frequency and mean pressure. Spatial evolution of temperature profile is also controlled by such local parameters, which include energy flow. (Energy flow includes both longitudinal conduction in the solid elements of an element, and enthalpy flow. Energy flow is a conceptually difficult parameter because it depends on the heat flows into adjacent heat exchangers and on acoustic power flowing along the apparatus. It is therefore like the frequency in a resonant system, in that it is a parameter that controls wave propagation in a segment but whose value is determined elsewhere.)

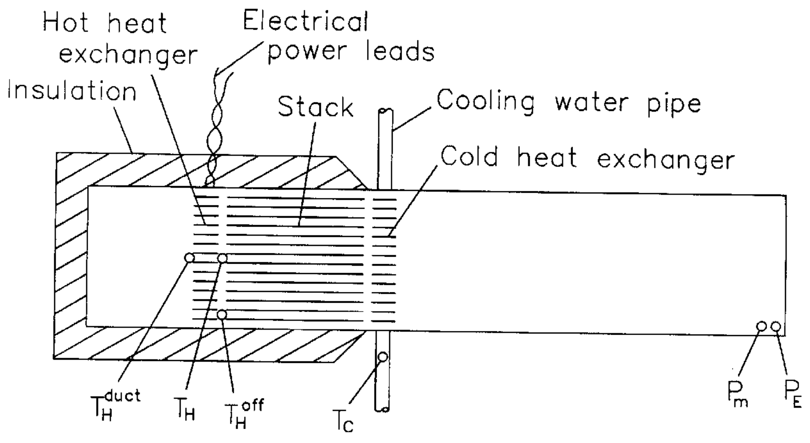


Figure III.1: Five-inch engine.

B. The 5-Inch Engine

The 5-inch engine is described in detail in [4]. The device described there is used to illustrate the fully thermoacoustic capabilities of DeltaE here; we reproduce some of the figures in that paper.

The apparatus is shown in Fig. III.1. Beginning with the input file (`5inch.in`, in the `examples` directory) to illustrate stack and heat exchanger segment types:

```
TITLE   Five-Inch Thermoacoustic Engine

BEGIN   Initial
13.8e5  (Pa) mean pressure
100.    (Hz) freq
500.    Starting Temp
8.e4    Mag(Pa) acoustic pressure @x=0
0.      Phase (deg) acoustic pressure @x=0
0.      Mag(vdot) vol. veloc @x=0
0.      Phase (deg) vol. veloc @x=0
helium  gas type

ENDCAP  Hot End
0.01292 (m^2) area
sameas 0  gas type

DUCT Hot Duct
sameas 1 (m^2) total area
0.403   (m) perim
0.279   (m) length
sameas 0  gas type

HX    Hot HX
sameas 1  (m^2) total area
0.393     gas area/total area
0.060     (m) length
0.483e-3 (m) y0 (this is half the gap)
2210.20  (W) heat
550.     (K) temperature
```

```
sameas 0   gas type

STKCIRC Honeycomb Stack
sameas 1  (m^2) total area
0.81     gas area/total area
0.279    (m) length
0.50e-3  (m) radius of each 'circular' pore
0.05e-3  (m) L:half of sht thcknss
sameas 0    gas type
stainless stack material

HX     Cold HX
0.01267  (m^2) total area
0.486    gas area/total area
0.0508   (m) length
0.406e-3 (m) y0
0.0      (W) heat
303.     (K) temperature
sameas 0    gas type

DUCT  Cold Duct
sameas 5  (m^2) total area
0.399    (m) perim
3.654  (m) length
sameas 0    gas type

ENDCAP  Cold End
sameas 5  (m^2) area
sameas 0   gas type

HARDEND
0.      Re(zinv)
0.      Re(zinv)
sameas 0   gas type
```

HX's are assumed to have parallel-plate geometry, with plate spacing $2y_o$. Other geometry is given in straightforward format. Wave propagation through heat exchangers is computed using a complex wavevector including both viscous and thermal dissipation in this geometry.

One additional feature of HX's is heat flow into the thermoacoustic system from an external heat source or out of the thermoacoustic system to an external heat sink. Positive heat flows into the apparatus. Generally, the heat flow determines the change in energy flux in the heat exchanger. Thus, heat flow can either be fixed (and optionally, an independent plot variable) or it can be a member of the guess vector. Here, this example will use the hot heat exchanger's heat flow as the independent plot variable and the cold heat exchanger's heat flow as a simple guessed result that will be largely ignored here.

A second additional feature of the HX's is the temperature difference between the mean-gas temperature and the heat exchanger metal temperature, proportional to the heat flow. Its dependence on the geometry of the heat exchanger is given in Chapter VI. [This temperature difference can presently be computed only within an accuracy of about a factor of 2, even in the acoustic approximation; nevertheless, it is included, to prevent naive users from being led to designs with heat exchangers of negligible surface area that have negligible losses and that would appear to have no disadvantages if the temperature difference were not included. We hope that future revisions of DeltaE will have an accurate calculation algorithm for this effect. Meanwhile, however, if you prefer not to use this feature, use the gas mean temperature instead of the metal temperature by using a `RPNTArget` (see Section VI A) to access the temperature in the adjacent stack segment (parameter G or H).] Metal temperature can be a target or a result. In this example, the cold heat exchanger's temperature is used as a target and the hot heat exchanger's temperature is used as a result and plotted.

Seven types of stacks are allowed: `STKSLab` for parallel-plate geometry, `STKCIrc` for circular pores, `STKREct` for rectangular and square pores, `STKPIns` for pin-array stacks, and `STKDUcts` for boundary-layer-approximation stacks (with all dimensions much greater than thermal and viscous penetration depths). `STKSCreen` and `STKPOwerlaw` for regenerators for Stirling systems will be introduced in Chapter IV. The geometry for each type is given in Chapter VI. Evolutions of T_m, p_1, and U_1 are computed as described in Refs. [1], [2], [19], [20].

You can execute DeltaE using the input file above and use `(C)lear|set` to ask for default targets:

```
No vectors defined...do you want enable a default
set of targets&guesses for this model? (y/n)   y
```

Examining the vector summary, we find:

```
Iteration Vectors Summary:
 GUESS      0b          0c          5e
 name  BEGIN:Freq. BEGIN:T-beg    HX:HeatI
 value    1.00E+02    5.00E+02    0.000
 units     Hz          K           W
TARGET      3f          5f          8a          8b
 name     HX:Est-T    HX:Est-T HARDE:R(1/z HARDE:I(1/z
 units      K           K
 value   550.0       303.0         .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
```

DeltaE has made plausible default choices for guess and target vector elements, but they are not exactly what we want. Using `(c)lear` and `(u)se`, we change the table to

```
Iteration Vectors Summary:
 GUESS      0b          0c          0d
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|p|
 value    1.00E+02    5.00E+02    8.00E+04
 units     Hz          K          Pa
TARGET      5f          8a          8b
 name     HX:Est-T HARDE:R(1/z HARDE:I(1/z
 units      K
 value    303.0        .00         .00

Potential TARGETS still available are:
```

```
Addr Seg:Par-Type    Current Value
 3f   HX  :Est-T =   550.0         K
```

which will give us a three-dimensional search, with end impedance and cold heat-exchanger temperature as targets.

Other choices could be made for this table. For instance, the cold-duct length could be substituted for the frequency in the guess vector. A fourth component, such as the hot heat-exchanger temperature `3f` could be added to the target vector and, simultaneously, the hot heat-exchanger heat `3e` could be added to the guess vector. For now, however, these vectors will be left as they are, since they reflect the point of view adopted in Ref. [4].

If you run this case, you will get the following `.dat` file:

```
 -= Five-Inch Thermoacoustic Engine                                         =-
 frequency=     121.023Hz     mean pressure=   1.380E+06Pa
  T(K)      Real and Imag p1(Pa)  Re & Im U1(m^3/s)       Hdot(W)   Edot(W)
   557.7     73450.       0.0      0.00000  0.00000          0.00      0.00
!------------------------------------ 1 -----------------------------------
 ENDCAP    Hot End
 Heat extracted:    1.22     Watts
   557.7     73450.       0.0     -0.00003  0.00000         -1.22     -1.22
!------------------------------------ 2 -----------------------------------
 DUCT      Hot Duct
 Duct wavvec =(   0.549    ,  -2.010E-03) m^-1
 Heat extracted:    10.5     Watts
   557.7     72589.       6.9     -0.00032 -0.08748        -11.76    -11.76
!------------------------------------ 3 -----------------------------------
 HX        Hot HX
 Heat exch wavvec =(   0.669    ,  -0.194    ) m^-1
 Heat =    2210.200 (W)    metal temp=     563.295 Kelvin
   557.7     71424.     482.4     -0.00202 -0.09651       2198.44    -95.37
!------------------------------------ 4 -----------------------------------
 STKCIRC   Honey Stack
   306.4     65548.    3147.5      0.01282 -0.15903       2198.44    169.94
!------------------------------------ 5 -----------------------------------
 HX        Cold HX
 Heat exch wavvec =(   0.858    ,  -0.162    ) m^-1
 Heat =   -2113.895 (W)    metal temp=     303.000 Kelvin
   306.4     62913.    3568.5      0.01215 -0.16675         84.55     84.55
!------------------------------------ 6 -----------------------------------
 DUCT      Cold Duct
 Duct wavvec =(   0.740    ,  -1.647E-03) m^-1
 Heat extracted:    83.9     Watts
   306.4    -69442.   -4136.6     -0.00002  0.00000          0.64      0.64
!------------------------------------ 7 -----------------------------------
 ENDCAP    Cold End
 Heat extracted:   0.642     Watts
   306.4    -69442.   -4136.6      0.00000  0.00000          0.00      0.00
!------------------------------------ 8 -----------------------------------
 HARDEND
 inverse impedance (rho a U/p A)=(  -3.410E-12,   2.163E-10)

   306.4    -69442.   -4136.6      0.00000  0.00000          0.00      0.00
```

This run will also produce the following `.out` file:

```
TITLE     Five-Inch Thermoacoustic Engine
!--------------------------------- 0 --------------------------------
 BEGIN      Initial
  1.3800E+06 a Mean P    Pa              121.0     A Freq.  G( 0b)     P
   121.0     b Freq.     Hz     G        557.7     B T-beg  G( 0c)     P
   557.7     c T-beg     K      G       7.3450E+04 C |p|@0  G( 0d)     P
  7.3450E+04 d |p|@0     Pa     G
  0.0000     e Ph(p)0    deg
  0.0000     f |U|@0    m^3/s
  0.0000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
!--------------------------------- 1 --------------------------------
 ENDCAP     Hot End
  1.2920E-02 a Area      m^2            7.3450E+04 A |p|       Pa
                                        0.0000     B Ph(p)     deg
                                        3.3241E-05 C |U|     m^3/s
                                         180.0     D Ph(U)     deg
                                        -1.221     E Hdot      W
 sameas  0  Gas type                    -1.221     F Edot      W
 ideal      Solid type                  -1.221     G HeatIn    W
!--------------------------------- 2 --------------------------------
 DUCT       Hot Duct
 sameas  1a  a Area      m^2            7.2589E+04 A |p|       Pa
  0.4030     b Perim      m             5.4761E-03 B Ph(p)     deg
  0.2790     c Length     m             8.7480E-02 C |U|     m^3/s
                                        -90.21     D Ph(U)     deg
                                        -11.76     E Hdot      W
 sameas  0  Gas type                    -11.76     F Edot      W
 ideal      Solid type                  -10.54     G HeatIn    W
!--------------------------------- 3 --------------------------------
 HX         Hot HX
 sameas  1a  a Area      m^2            7.1425E+04 A |p|       Pa
  0.3930     b GasA/A                   0.3869     B Ph(p)     deg
  6.0000E-02 c Length     m             9.6528E-02 C |U|     m^3/s
  4.8300E-04 d y0         m             -91.20     D Ph(U)     deg
   2210.     e HeatIn     W              2198.     E Hdot      W
   550.0     f Est-T      K      (t)    -95.37     F Edot      W
 sameas  0  Gas type                     2210.     G Heat       W
 ideal      Solid type                   563.3     H MetalT     K
!--------------------------------- 4 --------------------------------
 STKCIRC    Honey Stack
 sameas  1a  a Area      m^2            6.5624E+04 A |p|       Pa
  0.8100     b GasA/A                    2.749     B Ph(p)     deg
  0.2790     c Length     m             0.1595     C |U|     m^3/s
  5.0000E-04 d radius     m             -85.39     D Ph(U)     deg
  5.0000E-05 e Lplate     m              2198.     E Hdot      W
                                         169.9     F Edot      W
                                         557.7     G T-beg      K
 helium     Gas type                     306.4     H T-end      K
 stainless  Solid type                   265.3     I StkEdt     W
!--------------------------------- 5 --------------------------------
 HX         Cold HX
  1.2670E-02 a Area      m^2            6.3014E+04 A |p|       Pa
  0.4860     b GasA/A                    3.246     B Ph(p)     deg
  5.0800E-02 c Length     m             0.1672     C |U|     m^3/s
  4.0600E-04 d y0         m             -85.83     D Ph(U)     deg
  0.0000     e HeatIn     W              84.55     E Hdot      W
   303.0     f Est-T      K    = 5H?     84.55     F Edot      W
 helium     Gas type                    -2114.     G Heat       W
 ideal      Solid type                   303.0     H MetalT     K
!--------------------------------- 6 --------------------------------
 DUCT       Cold Duct
 sameas  5a  a Area      m^2            6.9565E+04 A |p|       Pa
  0.3990     b Perim      m             -176.6     B Ph(p)     deg
   3.654     c Length     m             1.8467E-05 C |U|     m^3/s
```

```
                                                 -176.6     D Ph(U)     deg
                                                 0.6423     E Hdot      W
 helium     Gas type                             0.6423     F Edot      W
 ideal      Solid type                           -83.90     G HeatIn    W
!--------------------------------  7 --------------------------------
 ENDCAP     Cold End
 sameas  5a  a Area      m^2                     6.9565E+04 A |p|       Pa
                                                 -176.6     B Ph(p)     deg
                                                 8.5383E-11 C |U|     m^3/s
                                                 -85.69     D Ph(U)     deg
                                                -4.6811E-08 E Hdot      W
 helium     Gas type                            -4.6811E-08 F Edot      W
 ideal      Solid type                           -0.6423    G HeatIn    W
!--------------------------------  8 --------------------------------
 HARDEND
  0.0000     a R(1/z)          = 8G?             6.9565E+04 A |p|       Pa
  0.0000     b I(1/z)          = 8H?             -176.6     B Ph(p)     deg
                                                 8.5383E-11 C |U|     m^3/s
                                                 -85.69     D Ph(U)     deg
                                                -4.6811E-08 E Hdot      W
                                                -4.6811E-08 F Edot      W
                                                -3.4102E-12 G R(1/z)
 helium     Gas type                             2.1633E-10 H I(1/z)
 ideal      Solid type                            306.4     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this data only if you really know your model!
INVARS      3  0  2  0  3  0  4
TARGS       3  5  6  8  1  8  2
SPECIALS    0
```

The `.dat` file is a segment-by-segment listing of results of the run. The three members of the guess vector (f, $T_{\rm begin}$, and $|p_1|_{\rm begin}$), which we had guessed would be near 100 Hz, 500 Kelvin, and 80,000 Pa, have turned out to be 121.020 Hz, 557.6 Kelvin, and 73,419 Pa; these values appear in the first few lines of `5inch.dat`. Temperature; real and imaginary pressure and volume flow rate; energy flow; and acoustic power flow are listed at each transition between segments. Be sure the complex volume flow rate at `HARDEnd` is zero, as required by two members of the target vector.

Some segments have additional information listed in the `.dat` file. Ducts and heat exchangers list wavevector (mostly real in the wide-open ducts; with large imaginary components in the much more lossy heat exchangers). Heat exchangers also list heat flow and metal temperature. Note that the metal is hotter than the gas in the hot heat exchanger, where the (positive) heat flows from metal to gas, and that the metal is cooler than the gas in the cold heat exchanger, where the (negative) heat flow is from gas to metal. Note also that DELTAE successfully hit the target metal temperature of 303 Kelvin in the cold heat exchanger.

Now examine the energy (`Hdot`) and acoustic power (`Edot`) flow columns in `5inch.dat`. The hot endcap absorbs 1.2 W of acoustic power, and the hot duct absorbs $11.8 - 1.2 = 10.6$ W of acoustic power. The minus signs on energy and acoustic power indicate energy flows ‘up’ the apparatus, toward the `BEGIN`ning.

The hot heat exchanger absorbs $95.4 - 11.8 = 83.6$ W of acoustic power. Because 2210.2 W of heat are added through it, the energy flow must increase by that amount; hence, the energy flow changes from -11.8 W to 2198.4 W in the hot heat exchanger.

The energy flow remains constant at 2198.4 W through the stack, which produces $169.9 - (-95.4) = 265.3$ W of acoustic power. Part of that power (95.4 W) flows up to supply work to the hot parts of the engine; the rest (169.9 W) flows down to supply work to the cold parts of the engine.

An examination of the cold heat exchanger listing parallels that of the hot heat exchanger, and the cold duct and endcap parallel the hot ones.

Some of this information is also available in the `.out` file, where it appears in a format that can be used as an input file for subsequent runs. The `.out` file is also a segment-by-segment listing, with a restart table appended. In the segment-by segment listing, the variables on the left are used in the input file. They include anything that can be used as a guess or target. Anything that was used as guess or independent plot variable contains its most recent value instead of the initial value supplied by the `.in` file. The variables on the right can be used as dependent variables in plots and can be compared to targets. We will encounter examples of each as we examine typical segments of this file.

The left portion of the `BEGIN` segment is in the `.in`-file format. `Freq, T-beg`, and `|p|` are marked with "`G`" signifying their membership in the guess vector. They also appear in the right column, marked with "`P`," signifying their status as default dependent plot variables. The right column of the `BEGIN` segment is a special case: it contains a copy of each guess vector variable with the values that were used in DeltaE's last iteration. To identify their origin, the units for each of these 'output' variables are replaced by the address (e.g., "`0b`") that they were copied from. This occurs *only* in the `BEGIN` segment.

Now examine the cold heat-exchanger segment. Again, the left column is the familiar input-file format. `Est-T` is marked "`=5H?`" to show that it is indeed a target variable, to be compared to the computed MetalT variable that appears in the right column.

`HARDEnd` has two more examples of the markers that indicate target variables. There, the target values are 0.0, and DeltaE's solution has reached $-3.4e - 12$ and $2.2e - 10$, which it judges to be close enough to zero.

The restart table at the end is translated thus:

```
INVARS      3  0  2  0  3  0  4     means 3 variables: 0b, 0c, 0d
TARGS       3  5  6  8  1  8  2     means 3 variables: 5f, 8a, 8b
```

This is an encoded version of the same information that is indicated by the guess and target flags, explained above, and is visible in the vector status summary table. Here, DELTAE would find this information automatically when using this `.out` file as a new input file.

To plot some results for this 5-inch engine case, execute DELTAE with this file again and modify the Plot summary to be

```
Dependent Variables (outputs):
PLOTS       0A          0B          0C          3H         8A
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|p|      HX:Metal HARDE:|p|
 units      Hz          K           Pa          K          Pa
Indpendent Variables (inputs):
 Outer loop:   3e   HX:HeatI Beg=  9.50E+02 End=   50.     Step=  -33.
```

Accomplishing this process required that we "plot another parameter" three times to add 3H and 8A to the dependent variable list and establish `3e` as independent variable and set its initial, final, and step values. (`T-beg` and `|p|` are of minor interest now, but could not be deleted from the list of plot variables because members of the guess vector appear here by default.)

Next, we modified mean pressure to be 19.2 bar, and ran the code. When completed, we modified mean pressure to 13.8 bar, and ran it again, appending the new results to the `.plt` file. Three more runs with mean pressures of 9.6, 6.9, and 5.2 bar completed the data set. We exited from DELTAE, and checked to see that it has created the `.des` and `.plt` files:

```
HX:HeatI BEGIN:Freq. BEGIN:T-beg BEGIN:|p|      HX:Metal HARDE:|p|
   W          Hz          K           Pa           K          Pa
  3e          0A          0B          0C          3H          8A

 950.0       120.6       562.7      5.9741E+04   566.2       5.6827E+04
 916.7       120.5       562.6      5.8637E+04   566.1       5.5777E+04
 883.4       120.5       562.6      5.7511E+04   566.0       5.4706E+04
 850.1       120.5       562.5      5.6362E+04   565.9       5.3614E+04
 816.8       120.5       562.5      5.5190E+04   565.7       5.2499E+04
   .
   .
   .
```

We read this `.plt` file into a spreadsheet/graphics program for minimal massaging: convert pressure amplitude at the cold end from Pascals to bar, and then square that number; subtract 303 Kelvin from T_h, and add the heat leak to the room to Q_h. Plotting these results then yields the curves shown in Fig. III.3, resembling Figs. 5, 6, and 7 in Ref. [4]. These curves differ slightly from those in the article, because of DELTAE's inclusion of the small gas-to-metal temperature differences in the heat exchangers.

More detailed comparison between DELTAE computations and measurements with this apparatus can be found in [5].

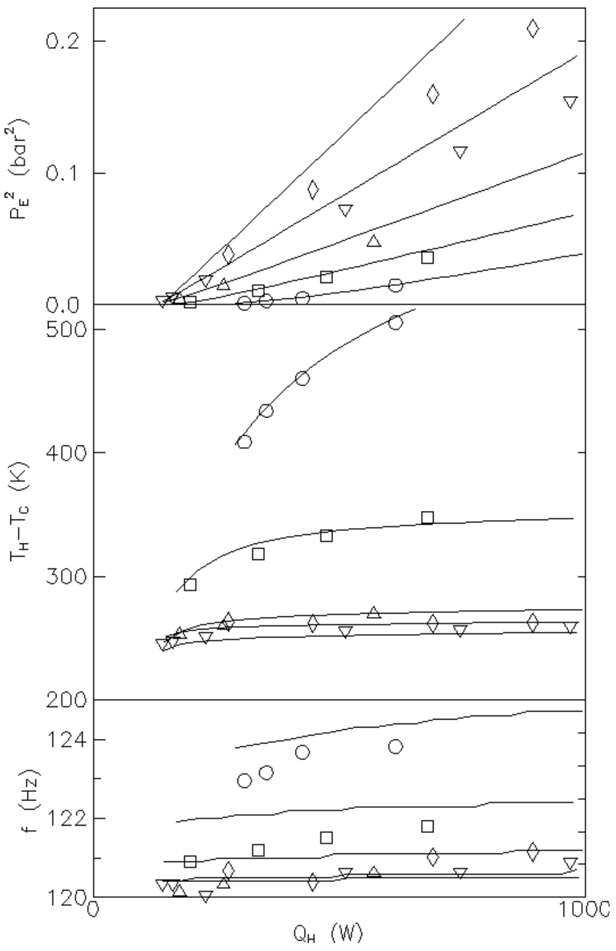


Figure III.2: 5-inch engine results. Lines are DeltaE results; points are from experimental data.

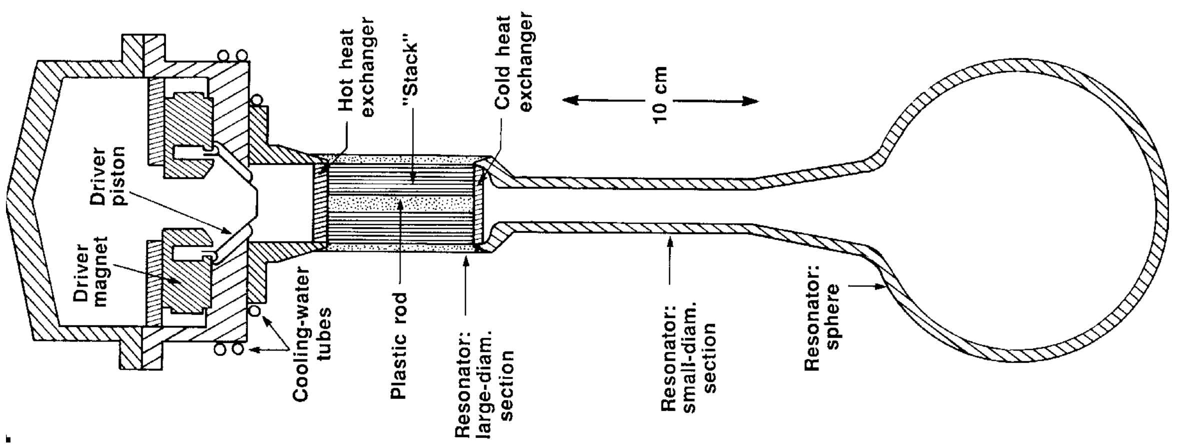


Figure III.3: Hofler's thermoacoustic refrigerator.

C. Hofler's Thermoacoustic Refrigerator

Tom Hofler's thermoacoustic refrigerator was described in detail in his Ph. D. thesis, Ref. [6]. The work was also summarized in Ref. [7]. We use this case to further illustrate capabilities of DeltaE, generating curves similar to Figs. 16 and 17 in Ref. [6] (Figs. 5 and 6 in Ref. [7]).

The apparatus is shown in Fig. III.3. We began with an input file (`hofler.in`, in the `examples` directory) whose geometry is that of Hofler's "long" apparatus:

```
TITLE     Hofler's 1986 thermoacoustic refrigerator

! Geometry comes from Hofler thesis, pages 28, 64, 68, 115, 130, 133.

BEGIN
1.0e6 Pa    Mean P
500. Hz     Freq.
300. K      T-beg
3.0e4 Pa    |p|@0
0.0 deg     Ph(p)0
5.0e-4 m3/s |U|@0
0.000 deg   Ph(U)0
helium      Gas

ENDCAP       driver end               1
1.134e-3 m2  Area
SAMEAS 0     Gas

DUCT       room temp duct             2
SAMEAS 1   Area
0.119 m    Perim
4.26e-2 m  Length
SAMEAS 0   Gas

HX        room temp heat exchanger    3
SAMEAS 1  Area
0.600     GasA/A
6.35e-3 m Length
```

```
1.9e-4 m  y0
-20.0 W   HeatIn
300. K    Est-T  (I hope this was the experimental value.)
SAMEAS 0  Gas

STKSLAB   Stack                       4
SAMEAS 1  Area
0.724     GasA/A
7.85e-2 m Length
1.8e-4 m  y0
4.0e-5 m  Lplate
SAMEAS 0  Gas
kapton    Solid

HX        Cold heat exchanger         5
SAMEAS 1  Area
0.67      GasA/A
2.54e-3 m Length
2.55e-4 m y0
3.0 W     Heatin
200. K    Est-T
SAMEAS 0  Gas

DUCT        Cold Duct                 6
3.84e-4 m2  Area
0.0694 m    Perim
0.167 m     Length
SAMEAS 0    Gas

CONE                                  7
SAMEAS 6   Initial Area
SAMEAS 6   In Perim
6.68e-2 m  Length
1.16e-3 m2 Final area
0.121 m    Final perim
SAMEAS 0

COMPLIANCE  end bulb                  8
0.049 m2    Area
1.06e-3 m3  Volume
SAMEAS 0    Gas

HARDEND
0.000     R(1/z)
0.000     I(1/z)
SAMEAS 0  Gas type
```

Note the use of segment type `STKSLab` to model the parallel-plate stack geometry, and the use of segment types `CONE` and `COMPLiance` to model parts of the cold portion of the resonator.

Executing DeltaE with this input file, we used `(C)lear|set` to ask for default targets,

```
No vectors defined...do you want enable a default
set of targets&guesses for this model? (y/n)   y
```

and examine the vector status summary:

```
Iteration Vectors Summary:
 GUESS      0b          0c          5e
 name  BEGIN:Freq. BEGIN:T-beg    HX:HeatI
 units     Hz          K           W
 value    5.00E+02    3.00E+02     3.00
TARGET      3f          5f          9a          9b
 name     HX:Est-T    HX:Est-T HARDE:R(1/z HARDE:I(1/z
 units      K           K
 value   300.0       200.0         0.000       0.000
result     0.000       0.000       0.000       0.000
Potential TARGETS still available:
Addr Seg:Par-Type    Current Value
```

Again, we are not fully satisfied with DeltaE's default choice of elements of this table. We add the driver's volume flow rate as a guess, to ensure that enough acoustic power is supplied to the system. And we remove the cold heat load as a guess and the cold temperature as a target. After making the necessary changes, the vector summary looks like

```
Iteration Vectors Summary:
 GUESS      0b          0c          0f
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|U|
 units     Hz          K          m^3/s
 value    5.00E+02    3.00E+02    5.00E-04
TARGET      3f          9a          9b
 name     HX:Est-T HARDE:R(1/z HARDE:I(1/z
 units      K
 value    3.00E+02     .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
 5f HXLAST:Est-T =   200.0         K
```

Running this case produced the following `.dat` file:

```
 -= Hofler's 1986 thermoacoustic refrigerator                              =-
 frequency=     499.183Hz     mean pressure=   1.000E+06Pa

  T(K)      Real and Imag p1(Pa)  Re & Im U1(m^3/s)       Hdot(W)   Edot(W)
   300.7     30000.       0.0      0.00051  0.00000           7.66      7.66
!------------------------------------ 1 ------------------------------------
 ENDCAP    Driver end                   1
 Heat extracted:   3.464E-02 Watts
   300.7     30000.       0.0      0.00051  0.00000           7.62      7.62
!------------------------------------ 2 ------------------------------------
 DUCT   Room temp duct
 Duct wavvec =(    3.09    ,  -1.301E-02) m^-1
 Heat extracted:   0.153     Watts
   300.7     29740.     -93.8      0.00049 -0.00273           7.47      7.47
!------------------------------------ 3 ------------------------------------
 HX    Room temp duct heat
 Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
 Heat =      -9.640 (W)    metal temp=     300.000 Kelvin
   300.7     29573.     -69.3      0.00044 -0.00302          -2.17      6.66
!------------------------------------ 4 ------------------------------------
 STKSLAB   Stack                       4
   218.6     26127.     640.8      0.00025 -0.00678          -2.17      1.05
!------------------------------------ 5 ------------------------------------
 HX    Cold HX
 Heat exch wavvec =(    4.03    ,  -0.505    ) m^-1
 Heat =       3.000 (W)    metal temp=     218.857 Kelvin
   218.6     25948.     662.0      0.00024 -0.00689           0.83      0.83
!------------------------------------ 6 ------------------------------------
 DUCT   Cold Duct                    6
 Duct wavvec =(    3.63     ,  -2.005E-02) m^-1
 Heat extracted:   0.678     Watts
   218.6      1754.      21.0      0.00027 -0.00862           0.15      0.15
!------------------------------------ 7 ------------------------------------
 CONE   7
 Heat extracted:   0.127     Watts
   218.6     -4216.    -138.7      0.00027 -0.00841           0.02      0.02
!------------------------------------ 8 ------------------------------------
 COMPLIAN  End Bulb                   8
 Heat extracted:   2.251E-02 Watts
   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
!------------------------------------ 9 ------------------------------------
 HARDEND   9
 inverse impedance (rho a U/p A)=(   3.213E-10,   1.141E-09)
   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
```

Close examination of this result for reasonableness reveals a problem: The stack is pumping 2.2 W of energy uphill, but 3.0 W of heat is being removed from the cold heat exchanger! How can this be? The problem is in our use of `DUCT` and `CONE` in the cold portion of the apparatus. In the default "isothermal" mode we have used thus far in the User's Guide, DeltaE assumes that these segments are held isothermal by external means. In this case, in the duct, cone, and compliance, where 0.83 W of acoustic power is dissipated into heat, some external means removes that heat. In Hofler's experiment, of course, no such "external means" existed; a good thermal connection between these parts and the cold heat exchanger caused this heat to appear as a load on the cold heat exchanger.

There are three methods to deal with this problem, i.e., to account for the fact that heat dissipated in these segments must show up in the cold heat exchanger. The first is to simply subtract the 0.83 W from the 3 W when we want to know the "actual" net refrigeration power available at the cold heat exchanger. This is not very elegant. The second is to use the insulated segment types `INSDUct` and `INSCOne` instead of `DUCT` and `CONE`. However, these segments do not work well in all circumstances and, beginning with DeltaE Version 5.0, we are discouraging their use. The third, and preferred, method is to enable DeltaE's "insulated" mode of operation, which is a new feature beginning with Version 5.0.

In the "conduction" mode that has been used in the User's Guide until now, ducts, cones, and similar segments can reject heat to environment, as necessary, as if a water jacket or air cooler or other external means can conduct heat away to a thermal reservoir at the segment's local temperature. [Note, for example, that $\dot{H}_2$ changes from the beginning to the end of all duct segments, and $\dot{H}_2 = \dot{E}_2$ at end of all duct segments, in the planewave-resonator example of Chapter II and the 5-inch engine example of the present chapter.]

In the "insulated" mode, ducts, cones, and similar segments are laterally thermally insulated, just like stacks, so that $\dot{H}_2$ at the end of the segment must be the same as at the beginning of the segment. This mode corresponds to the usual experimental state of affairs, such as in Hofler's refrigerator, where the heat created in a duct by dissipation of acoustic power must somehow find its way to a nearby heat exchanger.

To use the thermally insulated mode, insert segment `INSULate` somewhere before the segments that you want to insulate. This will change the behavior of $\dot{H}_2$ in most subsequent segments. To return to conduction mode later in the model, insert segment `CONDUct`.

So, to let DELTAE recognize the thermally insulated nature of the bottom end of Hofler's refrigerator, we insert `INSULate` just before the cold heat exchanger. We target the exiting $\dot{H}_2$ in the `HARDEND` segment to be zero, so that no energy can flow out of the end of the model. Corresponding to this target, we use the cold heat exchanger's heat flow as a guess. After making the necessary changes, the vector summary looks like

```
Iteration Vectors Summary:
 GUESS      0b          0c          0f          6e
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|U|      HX:HeatI
 units     Hz          K          m^3/s         W
 value    5.00E+02    3.00E+02    5.00E-04     3.000
TARGET      3f          9a          9b         10c
 name     HX:Est-T HARDE:R(1/z HARDE:I(1/z  HARDE: Hdot
 units      K                                   W
 value    300.0        0.00       0.00         0.00
```

Running this case produced

```
-= Hofler's 1986 thermoacoustic refrigerator                              =-
 frequency=     499.183Hz     mean pressure=   1.000E+06Pa

   Tm (K)      Re & Im p1 (Pa)      Re & Im U1(m3/s)        Hdot(W)   Edot(W)
   300.7    30000.0         0.0      0.00051  0.00000           7.66      7.66
!-----------------------------------  1 -----------------------------------
ENDCAP    driver end               1
 Heat extracted:   3.464E-02 Watts
   300.7    30000.0         0.0      0.00051  0.00000           7.62      7.62
!-----------------------------------  2 -----------------------------------
DUCT      room temp duct           2
 Duct wavvec =(    3.09    ,  -1.308E-02) m^-1
 Heat extracted:   0.155    Watts
   300.7    29740.5       -93.9      0.00049 -0.00273           7.47      7.47
!-----------------------------------  3 -----------------------------------
HX        room temp heat exchanger    3
 Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
 Heat =      -9.640 (W)    metal temp=     300.000 Kelvin
   300.7    29573.0       -69.4      0.00044 -0.00302          -2.17      6.66
!-----------------------------------  4 -----------------------------------
STKSLAB   Stack                    4
   218.6    26127.4       640.8      0.00025 -0.00678          -2.17      1.05
!-----------------------------------  5 -----------------------------------
INSUL     insulate the tail
   218.6    26127.4       640.8      0.00025 -0.00678          -2.17      1.05
!-----------------------------------  6 -----------  ++ therm insul mode ++
HX        Cold heat exchanger        5
 Heat exch wavvec =(    4.03    ,  -0.505    ) m^-1
 Heat =       2.171 (W)    metal temp=     218.771 Kelvin
   218.6    25948.4       661.9      0.00024 -0.00689           0.00      0.83
!-----------------------------------  7 -----------  ++ therm insul mode ++
DUCT      Cold Duct                  6
 Duct wavvec =(    3.63    ,  -2.019E-02) m^-1
   218.6     1754.3        20.9      0.00027 -0.00862           0.00      0.15
!----------------------------------  8 ------------   ++ therm insul mode ++
CONE      7
   218.6    -4215.9      -138.6      0.00027 -0.00841           0.00      0.02
!----------------------------------  9 ------------   ++ therm insul mode ++
COMPLIANCEend bulb                   8
   218.6    -4215.9      -138.6      0.00000  0.00000           0.00      0.00
!---------------------------------- 10 ------------   ++ therm insul mode ++
HARDEND   9
 inverse impedance (rho a U/p A)=(  -1.232E-17,   1.452E-16)

   218.6    -4215.9      -138.6      0.00000  0.00000           0.00      0.00
```

Thus, the acoustic power dissipated in the cold portion showed up automatically in the cold heat exchanger.

To generate plots for comparison to Hofler's data, we return to conduction mode, by eliminating the `INSULATE` segment, because Hofler added the dissipation in these components to his applied heat load for plotting. We let the heat at the cold heat exchanger be the independent variable, ranging from 2 to 8 W in 0.5 W steps. To plot the temperature ratio and the coefficient of performance (COP) relative to Carnot's COP, we include acoustic power at segment 1, T_c, and T_h in the list of plotted variables:

```
Dependent Variables (outputs):
PLOTS     0A          0B          0C          0D          1F        3H    5H
 name BEGIN:Freq. BEGIN:T-beg BEGIN:|U|@0 BEGIN:HeatI ENDCA:Edot HX:Metal HX:
 units    Hz          K         m^3/s        W           W          K       K
Indpendent Variables (inputs):
 Outer loop:   5e  HXLAS:HeatI Beg=   2.0     End=   8.0     Step=  0.50
```

(The first three dependent variables listed are unclearable defaults that we ignore.) After running this case, we changed $|p_1|$ to 0.015 of p_m, changed the range of Q_c to 0.7 to 3.7 W in steps of 0.5 W, and ran it again. Exiting DeltaE, we found the following `.des` and `.plt` files for the first case:

```
HXLAS:HeatI BEGIN:Freq .BEGIN:T-beg BEGIN:|U|@0 ENDCA:Edot HXFRS:Metal
HXLAS:Metal
     W          Hz          K          m^3/s       W           K           K
    5e          0A          0B          0C         1F         3H          5H

    2.000      493.1       300.6      4.7359E-04  7.069       300.0       212.9
    2.500      496.2       300.6      4.9203E-04  7.346       300.0       215.9
    3.000      499.2       300.7      5.1039E-04  7.321       300.0       218.9
    3.500      502.1       300.7      5.2869E-04  7.896       300.0       221.8
      .
      .
      .
```

Reading this file into spreadsheet/graphics software, and forming T_c/T_h and COPR, yielded the curves in Figs. III.4. These plots come reasonably close to the measurements presented in Figs. 16 and 17 of Hofler's thesis.

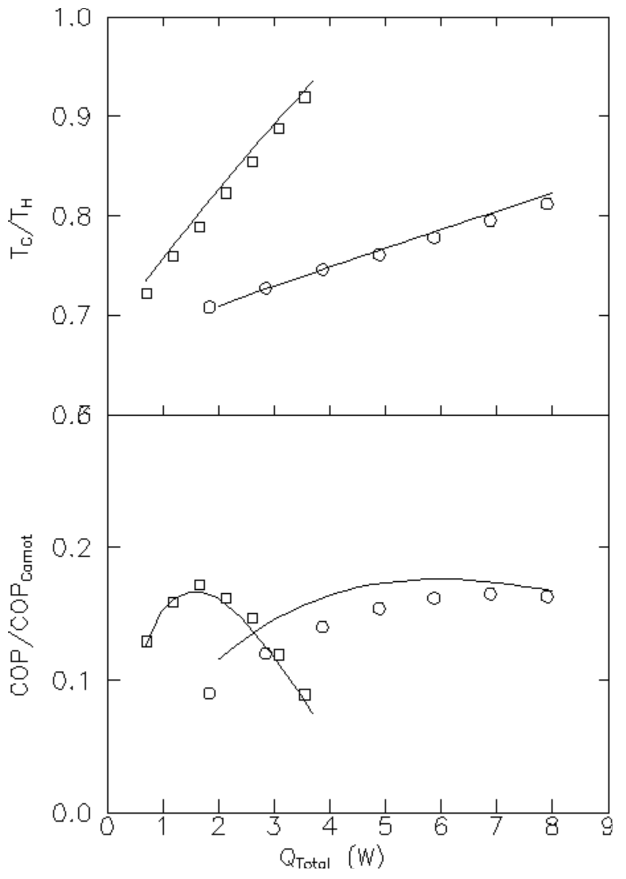


Figure III.4: Hofler refrigerator results. Lines are DeltaE results; points are from experimental data presented in Hofler's thesis. Squares , $p_1 = 0.015p_m$. Circles, $p_1 = 0.03p_m$.

Returning to insulated mode, we now use this example to introduce some more segment types. We add a more realistic driver to the system, using `VSPEAKer`. We edit the input file, adding `VSPEAKer` near the beginning. We deleted the `ENDCAp` segment that was near the beginning because `VSPEAker` accounts for the oscillatory pressurization losses on its surface area.

We also added a difference target using segment `RPNTArget` at the end. This feature will be described more fully in Chapter V. The first line of an `RPNTArget` segment (line "a") is the target value. The second line (line "b") is an instruction line expressing the desired algebraic procedure in Reverse Polish Notation, the parenthesis-free algebra encoding technique used by Hewlett-Packard calculators. Briefly, as numbers are encountered in the instruction line they are pushed onto a stack, and as arithmetic operators such as + and − are encountered they combine the previous numbers on the stack appropriately and put the result back on the stack. In the present example, the RPN instruction line causes the difference between results 1B and 1L to be calculated; the `RPNTARget` targeting feature allows this result to be targeted to zero.

```
TITLE     Hofler's 1986 thermoacoustic refrigerator, w speaker
 BEGIN                           0
   1.000E+06 a Mean P     Pa
    500.     b Freq.      Hz
    300.     c T-beg      K
   3.000E+04 d |p|@0      Pa
   150.0     e Ph(p)0    deg
    .000     f |U|@0    m^3/s
    .000     g Ph(U)0    deg
 helium      Gas type
 ideal       Solid type

 VSPEAKER                        1
   6.000E-04 a Area       m^2
    6.00     b   R       ohms
    .000     c L          H
    8.00     d B x L     T-m
   5.000E-03 e   M       kg
    .000     f   K       N/m
    .000     g  Rm      N-s/m
    20.      h AplVol     V
 SAMEAS  0  Gas type
 ideal       Solid type

 DUCT    room temp duct       2
   1.134E-03 a Area       m^2
    .119     b Perim       m
   4.260E-02 c Length      m
 SAMEAS  0  Gas type
 ideal       Solid type

 HX     room temp heat excha 3
 SAMEAS  2a  a Area       m^2
    .600     b GasA/A
   6.350E-03 c Length      m
   1.900E-04 d y0          m
   -10.      e HeatIn      W
    300.     f Est-T       K
 SAMEAS  0  Gas type
 ideal       Solid type
```

```
STKSLAB    Stack                 4
SAMEAS  2a  a Area      m^2
   .724     b GasA/A
  7.850E-02 c Length     m
  1.800E-04 d y0         m
  4.000E-05 e Lplate     m
SAMEAS  0  Gas type
kapton     Solid type

INSULATE

HX     Cold heat exchanger
SAMEAS  2a  a Area      m^2
   .670     b GasA/A
  2.540E-03 c Length     m
  2.550E-04 d y0         m
   2.19     e HeatIn     W
   200.     f Est-T      K
SAMEAS  0  Gas type
ideal      Solid type

DUCT    Cold duct
  3.840E-04 a Area      m^2
  6.940E-02 b Perim      m
   .167     c Length     m
SAMEAS  0  Gas type
ideal      Solid type

CONE
SAMEAS  6a  a AreaI     m^2
SAMEAS  6b  b PerimI     m
  6.680E-02 c Length     m
  1.160E-03 d AreaF     m^2
   .121     e PerimF     m
SAMEAS  0  Gas type
ideal      Solid type

COMPLIANCE end bulb
  4.900E-02 a Area      m^2
  1.060E-03 b Volum     m^3
SAMEAS  0  Gas type
ideal      Solid type

HARDEND
   .000     a R(1/z)
   .000     b I(1/z)
SAMEAS  0  Gas type
ideal      Solid type

RPNTARGET
   .000     a Target
 1B 1L -
```

The mass, resistance, and force constant for the speaker roughly reflect the values given in Hofler's thesis. We estimate it will take about 20 V to drive it.

We used the difference target `RPNTArget` segment to maintain resonance, by ensuring that the phases of p_1 and U_1 are equal at the driver. We did this by forcing their difference, computed by subtracting the values appearing as results 1B and 1L, to be zero, the value given in line `10a`. Examination of a `VSPEAKer` segment output

```
VSPEAKER
 6.0000E-04 a Area      m^2           3.0000E+04 A |p|        Pa
  6.000     b   R      ohms            153.8     B Ph(p)      deg
 0.0000     c L         H             5.0874E-04 C |U|      m^3/s
  8.000     d B x L    T-m             153.8     D Ph(U)      deg
 5.0000E-03 e   M      kg              7.631     E Hdot       W
 0.0000     f   K      N/m             7.631     F Edot       W
 0.0000     g   Rm   N-s/m             31.17     G EdotIn     W
  22.63     h AplVol    V      G       22.63     H Volts      V
                                       2.800     I Amps       V
                                      -10.30     J Ph(Ze)   deg
                                      5.0996E-04 K  |Ux|  m^3/s
sameas  0  Gas type                    153.8     L Ph(-Ux   deg
ideal      Solid type                 -23.54     M HeatIn     W
```

shows us that lines 1B and 1L contain the necessary information.

Running DeltaE with this input file, we modified guesses and targets to arrive at

```
Iteration Vectors Summary:
 GUESS      0b          0c          0e          1h         3e       6e
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:Ph(p) VSPEA:AplVo   HX:HeatI HX:HeatI
 units     Hz           K           deg          V           W        W
 value    5.00E+02    3.00E+02    150.         20.        -10.0     2.00
TARGET      3f          6f          10a         10b         10c      11a
 name     HX:Est-T    HX:Est-T HARDE:R(1/z HARDE:I(1/z  HARDE:Hdot RPNTA:Tar
 units      K            K
 value    3.00E+02     218.89     0.00         0.00        0.00     0.00
```

This shows our six-dimensional search. It is the most complicated vector summary table we have yet encountered, so we pause to discuss how we chose our vectors. We definitely needed the three `HARDEnd` impedances in the target vector (there is no hole in the end of the apparatus that would allow U_1 or $\dot{H}_2$ to escape). Experimentally, we maintain the hot heat exchanger at 300 Kelvin; but DeltaE computes that as a result of each integration pass, so it must also be a target. We have chosen a point of view where we want to choose the cold temperature to be 218.80 K and have DeltaE tell us the cooling power at that temperature, but DeltaE computes T_c as a result of each integration and requires Q_c as an input for each integration, so we must use T_C as a target and Q_c as a guess. So far we have discussed five targets and one guess, so we require at least four more guesses. Look first at the `BEGIN` segment for candidate guesses. Clearly the beginning temperature should be a guess: we need to guess beginning T to arrive at the first `HX` T correctly. Next, we must guess the frequency to maintain resonance. But how is resonance determined experimentally? By comparing the phases of p_1 and U_1 at the driver: hence, we added their difference $= 0$ as a sixth target. We need the phase of p_1 at the beginning to be a guess, since the phase of everything is determined relative to that of the speaker voltage phase, which is fixed at 0°. The heat rejected in the first heat exchanger must be guessed because we don't control it experimentally yet it is required by DeltaE in each pass. By now we have six targets and five guesses; we needed one more guess. Our guess could be $|p_1|$ at the beginning, which would be an experimental result if we controlled the drive voltage. Instead, however, we let the drive voltage be the guess because the experimenter used it to get $|p_1|$ to be 0.03 p_m.

Although most situations we encounter are not this confusing, choosing the vector members is never easy for a complicated thermoacoustic system. To choose them wisely, there is no substitute for careful thought about the system and what you want it to do. We offer a few general guides for this careful thought process. It is helpful to think about what variables are (or could be, in principle) experimentally controlled and what variables are experimentally observed. These must be compared with the variables that DeltaE needs as inputs during each integration pass through the system and those that DeltaE computes as results during each integration pass.

Table III.1: Interpreting input and result variables: experimental viewpoint

	Experimentally Controlled Variable	Experimentally a Result
Variable needed as input for each pass of DeltaE's integration	simply fixed in input file	guess
Variable computed as result of each pass of DeltaE's integration	target	simply a result in output files

Table III.2: Interpreting input and result variables: design viewpoint.

	Variable *we* want to think of as fixed	Variable *we* want to think of as a result
Variable needed as input for each pass of DeltaE's integration	simply fixed in input file	guess
Variable computed as result of each pass of DeltaE's integration	target	simply a result in output files

Note that our definition of an experimental result is more general than usual. In the Hofler refrigerator case, we considered the drive voltage an experimental result because it is determined experimentally by the condition that the pressure amplitude have the desired value. The viewpoint expressed in this Table III.1 is appropriate for comparison of DeltaE and experimental data. In this case, geometrical parameters are simply fixed. Targets are experimentally fixed or controlled variables that are results of a single pass of numerical integration, chosen from among T_m, p_1, and U_1 (everywhere but in `BEGIN`); current magnitudes and phases in VDUCERs and voltage magnitudes and phases in IDUCERs; etc. Guesses are known or unknown experimental results chosen from among f, the magnitude and phase of U_1-BEGIN and p_1-BEGIN, T_m-BEGIN, heats at heat exchangers, and the magnitude and phase of voltage at VDUCERs, etc.

To understand which variables are candidate guesses and which are candidate targets, you must know which are *needed* by DELTAE for each pass of its integration, and which are *computed* by DELTAE during each pass. To achieve this understanding, there is no good substitute for studying the summaries of the computation algorithms for each segment, as discussed briefly at the beginning of this chapter and more fully in Chapter VI.

When designing hardware instead of analyzing it, a different viewpoint may be adopted. In this case, many geometrical parameters are not yet fixed, but desired operating temperatures, powers, frequency, etc. have been chosen. Often, several geometrical parameters are included as guesses, and more temperatures and other numerical results are included as targets. Hence, another useful way to think about guesses and targets is represented by Table III.2.

Now we return to our example. Running this case produces the following dat file:

```
-= Hofler's 1986 thermoacoustic refrigerator                            =-
 frequency=     499.302Hz     mean pressure=   1.000E+06Pa

   Tm (K)      Re & Im p1 (Pa)      Re & Im U1(m3/s)        Hdot(W)   Edot(W)
   300.7   -26910.5     13259.8      0.00000  0.00000          0.00      0.00
!------------------------------------ 1 ------------------------------------
VSPEAKER  the loudspeaker
 (    22.6    ,   0.000E+00) Volts,(    2.76    ,   0.501    ) Amps
 Heat extracted: I^2 R=  23.5    , u^2 Rm= 0.000E+00, B-Layer= 1.833E-02 Watts.
   300.7   -26910.5     13259.8     -0.00046  0.00022          7.63      7.63
!------------------------------------ 2 ------------------------------------
DUCT      room temp duct           2
 Duct wavvec =(    3.09    ,  -1.308E-02) m^-1
 Heat extracted:   0.155     Watts
   300.7   -26636.1     13229.4      0.00076  0.00267          7.48      7.48
!------------------------------------ 3 ------------------------------------
HX        room temp heat exchanger    3
 Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
 Heat =      -9.670 (W)    metal temp=     300.000 Kelvin
   300.7   -26496.6     13133.4      0.00094  0.00290         -2.19      6.67
!------------------------------------ 4 ------------------------------------
STKSLAB   Stack                    4
   218.7   -23719.3     10973.0      0.00278  0.00620         -2.19      1.06
!------------------------------------ 5 ------------------------------------
INSUL     insulate the tail
   218.7   -23719.3     10973.0      0.00278  0.00620         -2.19      1.06
!------------------------------------ 6 ------------   ++ therm insul mode ++
HX        Cold heat exchanger         5
 Heat exch wavvec =(    4.03    ,  -0.506    ) m^-1
 Heat =       2.191 (W)    metal temp=     218.888 Kelvin
   218.7   -23568.0     10874.9      0.00283  0.00629          0.00      0.83
!------------------------------------ 7 ------------   ++ therm insul mode ++
DUCT      Cold Duct                6
 Duct wavvec =(    3.63    ,  -2.020E-02) m^-1
   218.7    -1582.7       756.5      0.00357  0.00786          0.00      0.15
!------------------------------------ 8 ------------   ++ therm insul mode ++
CONE      7
   218.7     3843.1     -1739.1      0.00348  0.00766           0.00      0.02
!------------------------------------  9 ------------   ++ therm insul mode ++
COMPLIANCEend bulb                   8
   218.7     3843.1     -1739.1      0.00000  0.00000           0.00      0.00
!------------------------------------ 10 ------------   ++ therm insul mode ++
HARDEND   9
 inverse impedance (rho a U/p A)=(  -6.143E-16,  -2.581E-15)
   218.7     3843.1     -1739.1      0.00000  0.00000           0.00      0.00
!------------------------------------ 11 ------------   ++ therm insul mode ++
RPNTARG   difference, to keep phase in driver ok
RPN stack: =    0.0000 ,
Opstring=  1B  1L -
   218.7     3843.1     -1739.1      0.00000  0.00000           0.00      0.00
```

This run also produces the following `.out` file:

```
TITLE     Hofler's 1986 thermoacoustic refrigerator
!---------------------------------  0 ---------------------------------
 BEGIN
  1.0000E+06 a Mean P     Pa                499.30    A Freq.  G( 0b)      P
   499.30    b Freq.      Hz      G         300.67    B T-beg  G( 0c)      P
   300.67    c T-beg      K       G         153.77    C Ph(p)  G( 0e)      P
  3.0000E+04 d  |p|       Pa                 22.634   D AplVol G( 1h)      P
   153.77    e Ph(p)      deg     G          -9.6705  E HeatIn G( 3e)      P
     0.0000  f  |U|     m^3/s                 2.1909  F HeatIn G( 6e)      P
     0.0000  g Ph(U)      deg
 helium     Gas type
 ideal      Solid type
!---------------------------------  1 ---------------------------------
 VSPEAKER   the loudspeaker
  6.0000E-04 a Area       m^2              3.0000E+04 A |p|       Pa
     6.0000  b   R        ohms              153.77    B Ph(p)     deg
     0.0000  c L          H                5.0899E-04 C |U|     m^3/s
     8.0000  d B x L      T-m               153.77    D Ph(U)     deg
  5.0000E-03 e   M        kg                  7.6348  E Hdot      W
     0.0000  f   K        N/m                 7.6348  F Edot      W
     0.0000  g   Rm    N-s/m                31.180    G WorkIn    W
    22.634   h AplVol     V       G         22.634    H Volts     V
                                              2.8004  I Amps      A
                                            -10.308   J Ph(Ze)   deg
                                           5.1021E-04 K  |Ux|  m^3/s
 sameas  0  Gas type                        153.77    L Ph(-Ux   deg
 ideal      Solid type                      -23.546   M HeatIn    W
!---------------------------------  2 ---------------------------------
 DUCT        room temp duct            2
  1.1340E-03 a Area       m^2              2.9741E+04 A |p|       Pa
     0.1190  b Perim      m                 153.59    B Ph(p)     deg
  4.2600E-02 c Length     m                2.7747E-03 C |U|     m^3/s
                                             74.032   D Ph(U)     deg
                                              7.4796  E Hdot      W
 sameas  0  Gas type                          7.4796  F Edot      W
 ideal      Solid type                       -0.1552  G HeatIn    W
!---------------------------------  3 ---------------------------------
 HX          room temp heat exchanger    3
  1.1340E-03 a Area       m^2              2.9573E+04 A |p|       Pa
     0.6000  b GasA/A                       153.63    B Ph(p)     deg
  6.3500E-03 c Length     m                3.0513E-03 C |U|     m^3/s
  1.9000E-04 d y0         m                  72.138   D Ph(U)     deg
    -9.6705  e HeatIn     W       G          -2.1909  E Hdot      W
   300.00    f Est-T      K     = 3H?         6.6718  F Edot      W
 sameas  0  Gas type                         -9.6705  G Heat       W
 ideal      Solid type                      300.00    H MetalT     K
```

```
!---------------------------------  4 --------------------------------
 STKSLAB    Stack                       4
  1.1340E-03 a Area      m^2             2.6134E+04 A |p|       Pa
     0.7240  b GasA/A                     155.17    B Ph(p)     deg
  7.8500E-02 c Length     m              6.7911E-03 C |U|     m^3/s
  1.8000E-04 d y0         m                65.855   D Ph(U)     deg
  4.0000E-05 e Lplate     m                -2.1909  E Hdot      W
                                             1.0554  F Edot      W
                                           300.67    G T-beg      K
 sameas  0  Gas type                       218.69    H T-end      K
 kapton     Solid type                      -5.6164  I StkEdt     W
!---------------------------------  5 +++++++++ therm insul mode +++++++++
 INSUL      insulate the tail
                                         2.6134E+04 A |p|       Pa
                                          155.17    B Ph(p)     deg
                                         6.7911E-03 C |U|     m^3/s
                                           65.855   D Ph(U)     deg
                                           -2.1909  E Hdot      W
                                             1.0554  F Edot      W
                                             0.0000  G HeatIn    W
!---------------------------------  6 +++++++++ therm insul mode +++++++++
 HX         Cold heat exchanger         5
  1.1340E-03 a Area      m^2             2.5956E+04 A |p|       Pa
     0.6700  b GasA/A                     155.23    B Ph(p)     deg
  2.5400E-03 c Length     m              6.8957E-03 C |U|     m^3/s
  2.5500E-04 d y0         m                65.761   D Ph(U)     deg
     2.1909  e HeatIn     W      G       7.2831E-14 E Hdot      W
   218.89    f Est-T      K    = 6H?       0.8292   F Edot      W
 sameas  0  Gas type                         2.1909  G Heat       W
 ideal      Solid type                     218.89    H MetalT     K
!---------------------------------  7 +++++++++ therm insul mode +++++++++
 DUCT       Cold Duct                   6
  3.8400E-04 a Area      m^2               1754.2     A |p|       Pa
  6.9400E-02 b Perim      m                 154.45    B Ph(p)     deg
     0.1670  c Length     m              8.6296E-03 C |U|     m^3/s
                                           65.584   D Ph(U)     deg
                                         7.2831E-14 E Hdot      W
 sameas  0  Gas type                         0.1494  F Edot      W
 ideal      Solid type                       0.0000  G HeatIn    W
!---------------------------------  8 +++++++++ therm insul mode +++++++++
 CONE
 sameas  7a  a AreaI     m^2               4218.3     A |p|       Pa
 sameas  7b  b PerimI     m                -24.348    B Ph(p)     deg
  6.6800E-02 c Length     m              8.4165E-03 C |U|     m^3/s
  1.1600E-03 d AreaF     m^2                65.580   D Ph(U)     deg
     0.1210  e PerimF     m              7.2831E-14 E Hdot      W
 sameas  0  Gas type                     2.2523E-02 F Edot      W
 ideal      Solid type                       0.0000  G HeatIn    W
!---------------------------------  9 +++++++++ therm insul mode +++++++++
 COMPLIANCE end bulb                    8
  4.9000E-02 a SurfAr    m^2               4218.3     A |p|       Pa
  1.0600E-03 b Volum     m^3               -24.348    B Ph(p)     deg
                                         2.8626E-16 C |U|     m^3/s
                                          -127.74    D Ph(U)     deg
                                         7.2831E-14 E Hdot      W
 sameas  0  Gas type                    -1.3982E-13 F Edot      W
 ideal      Solid type                       0.0000  G HeatIn    W
!--------------------------------- 10 +++++++++ therm insul mode +++++++++
 HARDEND
     0.0000  a R(1/z)          =10G?     4218.3     A |p|       Pa
     0.0000  b I(1/z)          =10H?      -24.348    B Ph(p)     deg
     0.0000  c  Hdot    W      =10E?     2.8626E-16 C |U|     m^3/s
                                          -127.74    D Ph(U)     deg
                                         7.2831E-14 E Hdot      W
                                        -1.3982E-13 F Edot      W
                                        -6.1433E-16 G R(1/z)
 sameas  0  Gas type                     -2.5806E-15 H I(1/z)
 ideal      Solid type                      218.69    I   T       K
!---------------------------------- 11 +++++++++ therm insul mode +++++++++
 RPNTARG    difference, to keep phase in driver ok
     0.0000  a Target          =11A?        0.0000  A RPNval
  1B  1L -

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      6   0  2   0  3   0  5   1  8   3  5   6  5
TARGS       6   3  6   6  6  10  1  10  2  10  3  11  1
SPECIALS    0
```

These acoustic and thermal results are the same as for without the speaker, except that everything is shifted in phase by -26 degrees. This shift occurred because we had set the phase of U_1 at the driver, arbitrarily, at zero before, but now the phase of the speaker voltage determines the zero of phase for the system, and the nonzero imaginary part of its mechanical impedance causes a phase shift between the voltage and the flow rate. New results appear in the VSPEAker segment; note for example that $|I|^2R/2$ is the difference between the acoustic power into the segment ($1/2\Re(I\tilde{V})$) and the acoustic power out of it.

D. Further Thermoacoustic Features

In this section we list the commonly used thermoacoustic segment types. More details on each, and a complete list, can be found in Chapter VI.

STKCIrc A stack with circular pores. We use this to model hexagonal honeycomb stacks.

STKSLab A stack with parallel-plate geometry.

STKREct A stack with rectangular (box) pore geometry.

STKPIns A stack comprised of an array of pins parallel to x.

STKDUct A stack with lateral dimensions much larger than δ_κ, computed in boundary-layer approximation.

STKSCreen A screen regenerator for Stirling systems.

HX A parallel-plate heat exchanger.

TX Tube-array heat exchanger, with the thermoacoustic working fluid inside the tubes.

SX Stacked-screen heat exchangers, valid only for δ_κ greater than hydraulic radius.

`PX` Power-law heat exchangers, with friction factor and heat-transfer coefficient characterized by power laws in Reynolds number.

`DUCT` A duct, with hydraulic radius much greater than δ_κ.

`CONE` A cone.

E. Advanced Operations

These menu options are not necessary for operation of the code, but they offer substantial conveniences for experienced users.

`(I)nsert segment.` DeltaE will prompt you for the correct number of parameters, giving the parameter name and units. This function is not perfectly interactive. If you make errors in typing in new parameter values, you will be left with a segment that is partly the same as the previous occupant of this spot. You may be able to recover by using the `(m)odify value` option in the main menu for numerical parameters. In the worst case (a bad segment type, for example), you may have to `(K)ill` the mistyped segment and start over again. `(I)nsert` before #segments+1 is permitted to add a segment at the very end.

`(K)ill segment.` This option simply removes a segment from your model. It works on any type of segment, and it does nothing intelligent with any lengths that are removed.

`(R)estore vectors.` Before beginning iterations during a `(r)un` operation, DeltaE saves copies of the guess vector values. Whenever an unsuccessful run overwrites the guess vector (leaving you and DeltaE hopelessly lost), you can use this option to restore all the parameters that were changed to their starting point. Simply `(R)estore`, modify some value(s), and try again. There are warnings about trying to use this option after the vector table has been edited, which of course would make no sense.

If you do not respond '`y`'es to the prompt about vector restoration and you have one or both plot loops enabled, you will be given an additional option:

```
Restore to state before last (B)egin or (r)un (y|n)? n
```

```
Restore from a recently plotted point? y
```

DeltaE will now proceed to display the `.plt` file one line at a time. After each line this prompt appears:

```
Return to this state (y|n|Q)? y
```

Typing '`y`' at this point causes the independent plot variable(s) and all members of the guess vector to be returned to those values displayed in the file. Typing '`n`' (or simply <CR>) causes the next line to be displayed. '`Q`' skips to the end of the file and makes no changes. No outputs are changed when this option is executed, so the model must be `(r)un` again to update them; however, be sure to disable the outer plot loop first if you want only one point. (Alternatively, you can change the step or endpoints of the plot loop and start plotting again.)

This option only works on the current (open) plot file, and it is not useful until after a run which has produced plot points.

`(E)xtras` The following model editing features are found under the `(E)xtras` submenu. Some less commonly used options (described in the next chapter) are also in this menu:

`(S)plit segment.` This option automates the laborious process of splitting a duct segment (or anything else that has a length) into two segments, each having half of the original length, correcting the `sameas` and math segment references, and correcting the iteration and plot vectors. [To partition the lengths unequally, it is convenient to use `(s)pecial modes editing` after splitting, to link the first length to the second, then `(m)odify` the first length, then clear (zero) the parameter linking before using the length in the iteration or optimization vector, if that is the intention.] All math segments, vectors, or `sameas` references to the segment specified are incremented by one; that is, the identifying number of the original segment is incremented by one, and the 'clone' segment is effectively inserted before it.

`(F)lip model.` For the same reason that DeltaE is most useful in the first place (i.e., because an adequate set of boundary conditions is almost never known at the most convenient point to start calculations), the number of guesses and targets can sometimes be reduced by starting the integration of a model from what you previously considered the 'bottom.' Orifice pulse tube refrigerators (described in Chapter IV), are a particularly good example because they 'end' with a known impedance, but the 'beginning' driving impedance is less well known. The `(F)lip model` operation automates switching back and forth between these two approaches to a solution, sparing the user from an effort that is otherwise tedious and very error prone. `(F)lip` reverses the order of every segment between the `BEGIN` and the last `HARDEND` or `SOFTEnd`. Segments within `TBRAN`ches are left in their original order, however. `sameas`, math segment and plot references are all adjusted and an attempt is made to reform the guess and target vectors.

Additional menu options are described at the end of Chapter V.

IV. Stirling Systems

Rott's equations implemented in DeltaE are valid for any phase difference between oscillatory pressure and oscillatory velocity, and any degree of thermal contact in the "stack." Hence, DeltaE can be used to model Stirling thermodynamic systems, in which p_1 and U_1 are substantially in phase, as well as thermoacoustic devices in which the phases p_1 and U_1 differ by nearly 90°. The principal additional DeltaE segment needed is one for stacked screen beds, because stacked screen regenerators are more common than parallel-plate, circular, or rectangular pore regenerators. In our opinion, the principal shortcomings of DeltaE for Stirling applications are DeltaE's acoustic approximation (which leads to reduced accuracy at high pressure amplitudes) and its inability to predict end effects and streaming-driven convective heat transport in pulse tubes (a shortcoming shared by many other design programs). Its main virtues are speed and easy integral modeling of some auxiliary components such as ducts, dead volumes, and linear motors.

Harmonic analysis of Stirling systems is discussed by I. Urieli and D. M. Berchowitz in Ref. [8] and by A. J. Organ in Ref. [9].

A. Principles of Computation—Stacked Screens

The full details of the stacked-screen computation method implemented in DeltaE are described in Ref. [10]. As usual in DeltaE, for each pass through DeltaE's integration we adopt the point of view described at the beginning of Chapter III: We regard p_1, U_1, and T_m as the dependent variables of interest. Given their values at one end, we can generate $p_1(x)$, $U_1(x)$, and $T_\mathrm{m}(x)$ throughout the regenerator, using equations of the form

$$dp_1/dx = F_1(p_1, U_1, T_\mathrm{m}, \overline{H_2},\ \text{geometry}), \tag{IV.1}$$

$$dU_1/dx = F_2(p_1, U_1, T_\mathrm{m}, \overline{H_2},\ \text{geometry}), \tag{IV.2}$$

$$dT_\mathrm{m}/dx = F_3(p_1, U_1, T_\mathrm{m}, \overline{H_2},\ \text{geometry}). \tag{IV.3}$$

The exact forms of these equations are displayed in Chapter VI below. Because p_1 and U_1 are complex, Eqs. (IV.1)-(IV.3) actually represent 5 real first-order differential equations. Equation (IV.1) is based largely on the screen friction factor data of Kays and London, Ref. [11]. Equation (IV.2) is based on the continuity equation, and Eq. (IV.3) on the equation for time-averaged energy flux $\overline{H_2}$ through the regenerator; both of the latter use the screen heat transfer coefficient data from Kays and London. The equations are not accurate for hydraulic radius on the order of δ_κ or greater.

The segment type implementing this algorithm is called `STKSCreen`. The corresponding heat exchanger, comprising stacked screens, is called `SX`, in which p_1 and U_1 are computed using Eqs. (IV.1) and (IV.2), with $dT_\mathrm{m}/dx = 0$. As with the parallel-plate heat exchange segments `HX`, an estimated gas-to-metal temperature difference, proportional to the heat exchanger's heat flow, is also incorporated.

B. Stirling Cryocooler

The sample file `Stirling.out` represents a simple 55 Hz, 2 MPa helium Stirling cryocooler with stacked-screen regenerator and heat exchangers. This apparatus is illustrated in Fig. IV.1. First, we examine `Stirling.out`:

```
TITLE     Bare bones Stirling cryocooler
!---------------------------------  0 ---------------------------------
 BEGIN      Initialize things    0
  2.0000E+06 a Mean P    Pa                 300.14    A T-beg  G( 0c)       P
    55.000   b Freq.     Hz               2.8459E+05 B  |p|   G( 0d)       P
   300.14    c T-beg     K       G          -43.119   C Ph(p)  G( 0e)       P
  2.8459E+05 d  |p|      Pa      G          -35.833   D HeatIn G( 1e)       P
   -43.119   e Ph(p)     deg     G
  3.6500E-04 f  |U|     m^3/s
    0.0000   g Ph(U)     deg
 helium     Gas type
 ideal      Solid type
!---------------------------------  1 ---------------------------------
 SX         aftercooler          1
 sameas  2a  a Area      m^2               2.8085E+05 A |p|        Pa
    0.6000   b VolPor                        -43.835   B Ph(p)      deg
  1.0000E-03 c Length     m                3.6265E-04 C |U|      m^3/s
 sameas  2d  d   r_H      m                   -0.3900  D Ph(U)      deg
   -35.833   e HeatIn     W      G             2.0777  E Hdot       W
   300.00    f Est-T      K    = 1H?          36.973   F Edot       W
 sameas  0  Gas type                         -35.833   G Heat        W
 copper     Solid type                        300.00   H MetalT      K
!---------------------------------  2 ---------------------------------
 STKSC      regenerator          2
  1.1670E-04 a Area      m^2               2.2845E+05 A |p|        Pa
    0.6860   b VolPor                        -53.178   B Ph(p)      deg
  5.0000E-02 c Length     m                6.2225E-05 C |U|      m^3/s
  1.3900E-05 d   r_H      m                  -49.487   D Ph(U)      deg
    0.3000   e KsFrac                          2.0777  E Hdot       W
                                               7.0928  F Edot       W
                                              300.14   G T-beg       K
 sameas  0  Gas type                           79.960  H T-end       K
 stainless  Solid type                       -29.880   I StkEdt      W
!---------------------------------  3 ---------------------------------
 SX         cold heat exch       3
 sameas  2a  a Area      m^2               2.2804E+05 A |p|        Pa
    0.6000   b VolPor                        -53.184   B Ph(p)      deg
  1.0000E-03 c Length     m                6.2000E-05 C |U|      m^3/s
 sameas  2d  d   r_H      m                  -52.000   D Ph(U)      deg
```

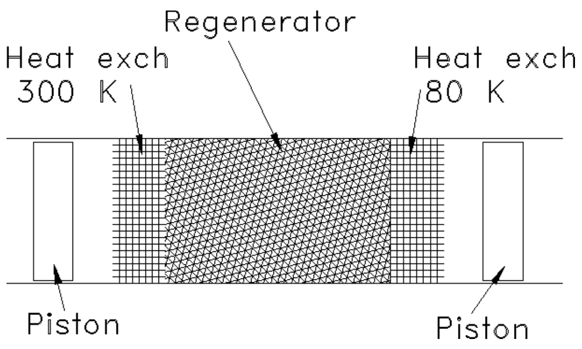


Figure IV.1: The Stirling cryocooler.

```
     0.0000  e HeatIn     W                    7.0677  E Hdot       W
    80.000   f Est-T      K    = 3H?           7.0677  F Edot       W
 sameas  0  Gas type                           4.9900  G Heat        W
 copper     Solid type                        80.000   H MetalT     K
!---------------------------------  4 ---------------------------------
 RPNTARG    U sub 1 at cold end  4
  6.2000E-05 a Target          = 4A?     6.2000E-05 A RPNval
  3C
!---------------------------------  5 ---------------------------------
 RPNTARG    phase(U) at cold end 5
   -52.000   a Target          = 5A?      -52.000   A RPNval
  3D

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      4   0  3   0  4   0  5   1  5
TARGS       4   1  6   3  6   4  1   5  1
SPECIALS    0
```

The real segments consist of a first heat exchanger, at 300 K, the regenerator, and a second heat exchanger at 80 K. All three are stacked screens. The other segments—`BEGIN` and the `RPNTARGET`s—simply define the boundary conditions.

We obtained the hydraulic radius r_h and volumetric porosity ϕ for the screens by hand, using expressions from Organ's book:

$$\phi = 1 - \frac{\pi m d}{4}\sqrt{1 + (md)^2}$$

$$r_h = \frac{d}{4}\frac{\phi}{1-\phi}$$

where d is wire diameter and m is mesh number (i.e., number of wires per unit length). The regenerator is a little over 1 cm in diameter and is 5 cm long. The heat exchangers are the same diameter but only 1 mm long.

DeltaE estimates the temperature difference between the helium gas and the copper screen wires in the heat exchangers, but it has no provision for estimating the temperature difference between the screen wires and the “housing” in which they are mounted (due to the finite thermal conductance of the screen wires themselves). This is not a serious concern for small machines, but should be checked by hand on a case-by-case basis.

Line e in the regenerator segment, “KsFrac,” is the fudge factor by which longitudinal conduction through the regenerator is adjusted due to the spatially intermittent thermal contact between adjacent screens and due to the conduction of the pressure-vessel wall. Following Ref. [12], we often set KsFrac somewhere between 0.1 and 0.3.

Our point of view with respect to boundary conditions in this example is most easily displayed by running DeltaE on this file and examining the vector summary

```
Iteration Vectors Summary:
 GUESS      0c          0d          0e          1e
 name   BEGIN:T-beg BEGIN:|p|   BEGIN:Ph(p)    SX:HeatI
 units      K           Pa          deg          W
 value    3.00E+02    2.93E+05    -43.         -37.
TARGET      1f          3f          4a           5a
 name     SX:Est-T    SX:Est-T  RPNT:Targe  RPNT:Targe
 units      K           K
 value    3.00E+02     80.        6.20E-05    -52.
result     .00         .00          .00         .00
```

and the `BEGIN` segment above. Here, we are considering the volume flow rates (both magnitudes and phases) at the two ends to be given, as if we have in mind an “alpha” Stirling machine, with two pistons determining the volumes of the compression and expansion spaces, respectively. The volume flow rate at the hot end is set by lines f and g in the `BEGIN` segment. The 0° phase of line `0g` essentially determines the zero of phase for the entire system. The volume flow rate 3.65×10^{-4} m^3/s of line `0f`, (together with the frequency set in line `0b`), implies a volumetric stroke of 2.1 cm^3 peak-to-peak at the hot end. The `RPNTARGET`s at the cold end ensure that DeltaE’s shooting method arrives there with the desired cold piston stroke and phase. To arrive at these two targets, DeltaE adjusts two guesses: the pressure amplitude and phase in the `BEGIN` segment (and hence throughout the cooler). We also insist that the metal temperatures in the two heat exchangers be 300 K and 80 K; DeltaE achieves these two targets by adjusting two more guesses: the heat extracted at the hot heat exchanger, and the temperature in the `BEGIN` segment.

DeltaE predicts that, under these circumstances, the cooler will reject 36 W at the hot heat exchanger and will have a cooling power of 5 W. This cooling power accounts for heat conduction and enthalpy flow through the regenerator, but does not account for any heat load imposed by frictional irreversibilities in the cold piston, nor any heat load imposed by the regenerator *case* conduction unless it is included in KsFrac.

We now make or suggest a few simple modifications to this file to illustrate additional features of DeltaE.

To discover what temperature the cooler would maintain with a heat load of 10 W instead of 5 W, we `(c)lear` 3f—the cold heat exchanger temperature—from the target list. Instead, we `(u)se` 3e—the cooling power—as a target, and `(m)odify` it to 10 W. Running DeltaE shows that under these circumstances the cold temperature will be 232 K. Using 3e as an independent plot variable running from 10 W to 2 W with steps of, say, 0.5 W, and using 3H (cold metal temperature) as dependent plot variable will generate a table of cold temperature (and other defaults) vs heat load:

```
gross cooling                                                  metal temp
   power                                                       @ cold hx
   SX:HeatI BEGIN:T-beg BEGIN:|p|   BEGIN:Ph(p)    SX:HeatI    SX:Metal
      W          K          Pa          deg            W          K
     3e         0A          0B          0C            0D          3H
    10.00       300.1      4.3154E+05  -71.57       -24.00       232.0
    9.500       300.1      4.1420E+05  -70.09       -24.73       212.4
    9.000       300.1      3.9704E+05  -68.40       -25.56       193.6
    8.500       300.1      3.8016E+05  -66.45       -26.48       175.7
    8.000       300.1      3.6368E+05  -64.22       -27.51       158.8
    7.500       300.1      3.4778E+05  -61.67       -28.66       142.9
    7.000       300.1      3.3265E+05  -58.75       -29.91       128.1
    6.500       300.1      3.1853E+05  -55.43       -31.28       114.4
    6.000       300.1      3.0571E+05  -51.69       -32.76       101.9
    5.500       300.1      2.9449E+05  -47.52       -34.35       90.41
    5.000       300.1      2.8518E+05  -42.94       -36.03       79.98
    4.500       300.2      2.7809E+05  -37.99       -37.79       70.53
    4.000       300.2      2.7348E+05  -32.74       -39.63       61.99
    3.500       300.2      2.7154E+05  -27.31       -41.54       54.28
    3.000       300.2      2.7240E+05  -21.81       -43.51       47.32
    2.500       300.2      2.7609E+05  -16.38       -45.54       41.02
    2.000       300.2      2.8252E+05  -11.14       -47.62       35.32
```

Insertion of two `'SPEAker` segments before the aftercooler and after the cold heat exchanger would model use of linear motors driving pistons there.

Finally, in the next chapter "Advanced Features" we will use `TBRANCH` and `UNION` to change this model from an "alpha" Stirling machine to a "beta" or "gamma," with one power piston on the hot end and a displacer piston in parallel with the heat exchange elements.

C. Pulse Tube Refrigerator

Changing a Stirling cryocooler into an orifice pulse tube refrigerator (OPTR) is a simple matter of replacing the cold piston with a pulse tube, heat exchanger, orifice, and reservoir volume in series. Figure IV.2 represents such a cooler. The sample file `optr.in` represents a 300 Hz, 3 MPa helium orifice pulse tube refrigerator. After running DeltaE on `optr.in`, we find the following `.out` and `.dat` files:

```
TITLE     an early cooler design, not optimal
!--------------------------------  0 --------------------------------
 BEGIN      Start with 8% p osc
```

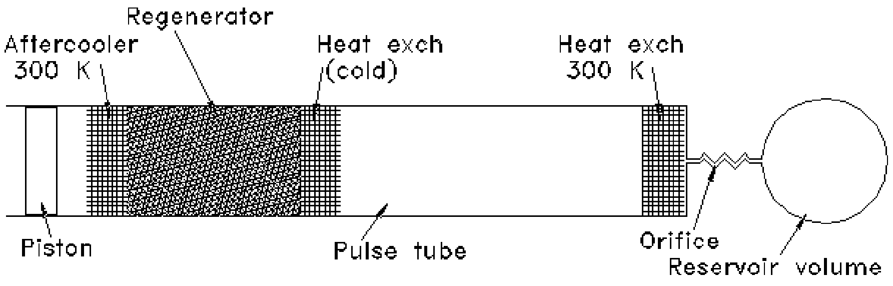


Figure IV.2: An Orifice Pulse Tube Refrigerator (OPTR).

```
 3.0000E+06 a Mean P    Pa               300.10     A T-beg  G( 0c)       P
  300.00    b Freq.     Hz             7.1461E-03 B  |U|    G( 0f)       P
  300.10    c T-beg     K       G        50.447   C Ph(U)  G( 0g)       P
 2.4000E+05 d  |p|      Pa             -491.47    D HeatIn G( 1e)       P
    0.0000  e Ph(p)     deg                5.4026 E HeatIn G( 3e)       P
 7.1461E-03 f  |U|     m^3/s    G        -60.005  F HeatIn G( 6e)       P
   50.447   g Ph(U)     deg     G
helium      Gas type
ideal       Solid type
!--------------------------------  1 --------------------------------
SX          Aftercooler
 1.0290E-03 a Area      m^2            2.2897E+05 A |p|       Pa
    0.6900  b VolPor                     -3.5853  B Ph(p)     deg
 1.2500E-02 c Length     m             6.0475E-03 C |U|      m^3/s
 6.4500E-05 d   r_H      m                44.215  D Ph(U)     deg
 -491.47    e HeatIn    W       G         54.602  E Hdot      W
  300.00    f Est-T     K     = 1H?      465.06   F Edot      W
helium      Gas type                   -491.47    G Heat       W
copper      Solid type                   300.00   H MetalT     K
!--------------------------------  2 --------------------------------
STKSCRN     Regenerator
sameas  1a  a Area      m^2            1.6394E+05 A |p|       Pa
    0.7300  b VolPor                     -21.105  B Ph(p)     deg
 5.5000E-02 c Length     m             1.3356E-03 C |U|      m^3/s
 2.4000E-05 d   r_H      m               -24.425  D Ph(U)     deg
    0.3000  e KsFrac                      54.602  E Hdot      W
                                         109.29   F Edot      W
                                         300.10   G T-beg      K
helium      Gas type                     149.97   H T-end      K
stainless   Solid type                  -355.76   I StkEdt     W
!--------------------------------  3 --------------------------------
SX          Cold heat exchanger
sameas  4a  a Area      m^2            9.4430E+04 A |p|       Pa
    0.6900  b VolPor                     -19.397  B Ph(p)     deg
 2.0000E-03 c Length     m             1.3349E-03 C |U|      m^3/s
 6.4500E-05 d   r_H      m               -24.686  D Ph(U)     deg
    5.4026  e HeatIn    W       G         60.005  E Hdot      W
  150.00    f Est-T     K     = 3H?       62.761  F Edot      W
helium      Gas type                       5.4026 G Heat       W
copper      Solid type                   150.00   H MetalT     K
!--------------------------------  4 --------------------------------
STKDU       Pulse tube
 5.6870E-05 a Area      m^2     S=-2   9.8911E+04 A |p|       Pa
 2.6740E-02 b Perim      m    Fnc( 4a)   -55.206  B Ph(p)     deg
    0.2000  c Length     m             1.2941E-03 C |U|      m^3/s
 1.0000E-05 d WallA     m^2              -42.669  D Ph(U)     deg
                                          60.005  E Hdot      W
                                          62.472  F Edot      W
                                         149.97   G T-beg      K
helium      Gas type                     300.19   H T-end      K
stainless   Solid type                    -0.2886 I StkEdt     W
```

```
!---------------------------------  5 +++++++++ therm insul mode +++++++++
 INSULATE   assume the impedance and compliance are thermally insulated
                                         9.8911E+04 A |p|       Pa
                                          -55.206   B Ph(p)     deg
                                         1.2941E-03 C |U|     m^3/s
                                          -42.669   D Ph(U)     deg
                                           60.005   E Hdot      W
                                           62.472   F Edot      W
                                           0.0000   G HeatIn    W
!---------------------------------  6 +++++++++ therm insul mode +++++++++
 SX         Hot heat exchanger
 sameas  4a  a Area      m^2             2.6246E+04 A |p|       Pa
     0.6900  b VolPor                    -103.37    B Ph(p)     deg
  5.0000E-03 c Length     m              1.2907E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m               -42.956   D Ph(U)     deg
   -60.005   e HeatIn     W      G       1.7588E-08 E Hdot      W
   300.00    f Est-T      K    = 6H?        8.3624  F Edot      W
 helium     Gas type                      -60.005   G Heat       W
 copper     Solid type                     300.00   H MetalT     K
!---------------------------------  7 +++++++++ therm insul mode +++++++++
 IMPEDANCE  The orifice
  1.0000E+07 a Re(Zs) Pa-s/m^3           2.2824E+04 A |p|       Pa
     0.0000  b Im(Zs) Pa-s/m^3           -132.83    B Ph(p)     deg
                                         1.2907E-03 C |U|     m^3/s
                                          -42.956   D Ph(U)     deg
                                         1.7588E-08 E Hdot      W
 sameas  0  Gas type                     3.3401E-02 F Edot      W
 ideal      Solid type                     -8.3290  G HeatIn    W
!---------------------------------  8 +++++++++ therm insul mode +++++++++
 COMPLIANCE Reservoir volume
  1.2680E-02 a SurfAr    m^2             2.2824E+04 A |p|       Pa
  1.5000E-04 b Volum     m^3             -132.83    B Ph(p)     deg
                                         3.4260E-10 C |U|     m^3/s
                                           -7.7865  D Ph(U)     deg
                                         1.7588E-08 E Hdot      W
 sameas  0  Gas type                    -2.2447E-06 F Edot      W
 ideal      Solid type                     0.0000   G HeatIn    W
!---------------------------------  9 +++++++++ therm insul mode +++++++++
 HARDEND    The end
     0.0000  a R(1/z)          = 9G?     2.2824E+04 A |p|       Pa
     0.0000  b I(1/z)          = 9H?     -132.83    B Ph(p)     deg
     0.0000  c  Hdot    W      = 9E?     3.4260E-10 C |U|     m^3/s
                                           -7.7865  D Ph(U)     deg
                                         1.7588E-08 E Hdot      W
                                        -2.2447E-06 F Edot      W
                                        -3.3335E-09 G R(1/z)
 helium     Gas type                     4.7538E-09 H I(1/z)
 ideal      Solid type                     300.19   I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      6   0  3   0  6   0  7   1  5   3  5   6  5
TARGS       6   1  6   3  6   6  6   9  1   9  2   9  3
SPECIALS    2   4 -2   7 -5



-= an early cooler design, not optimal                                    =-
 frequency=     300.000Hz     mean pressure=   3.000E+06Pa

   Tm (K)      Re & Im p1 (Pa)      Re & Im U1(m3/s)        Hdot(W)   Edot(W)
   300.1   240000.0         0.0      0.00455  0.00551        546.07    546.07
!-----------------------------------  1 -----------------------------------
SX         Aftercooler
 Heat =    -491.469 (W)    metal temp=     300.000 Kelvin
```

```
   300.1   228521.3    -14318.5      0.00433  0.00422          54.60    465.06
!------------------------------------  2 ------------------------------------
STKSCRN   Regenerator
   150.0   152943.7    -59030.9      0.00122 -0.00055          54.60    109.29
!------------------------------------  3 ------------------------------------
SX        Cold heat exchanger
 Heat =        5.403 (W)    metal temp=     150.000 Kelvin
   150.0    89070.2    -31361.3      0.00121 -0.00056          60.00     62.76
!------------------------------------  4 ------------------------------------
STKDU     Pulse tube
   300.2    56442.0    -81226.7      0.00095 -0.00088          60.00     62.47
!------------------------------------  5 ------------------------------------
INSULATE  assume the impedance and compliance are thermally insulated
   300.2    56442.0    -81226.7      0.00095 -0.00088          60.00     62.47
!------------------------------------  6 ------------   ++ therm insul mode ++
SX        Hot heat exchanger
 Heat =      -60.005 (W)    metal temp=     300.000 Kelvin
   300.2    -6069.1    -25534.5      0.00094 -0.00088           0.00      8.36
!------------------------------------  7 ------------   ++ therm insul mode ++
IMPEDANCE The orifice
 Imped. work =    8.33     Watts
   300.2   -15515.1    -16739.5      0.00094 -0.00088           0.00      0.03
!------------------------------------  8 ------------   ++ therm insul mode ++
COMPLIANCEReservoir volume
   300.2   -15515.1    -16739.5      0.00000  0.00000           0.00      0.00
!------------------------------------  9 ------------   ++ therm insul mode ++
HARDEND   The end
 inverse impedance (rho a U/p A)=(  -3.334E-09,   4.754E-09)

   300.2   -15515.1    -16739.5      0.00000  0.00000           0.00      0.00
```

The Stirling part of the system is modeled as a stacked-screen regenerator STKSCREEN and two stacked-screen heat exchangers SX. We model the pulse tube itself as a STKDUCT, using Rott's wave equation and enthalpy flux equation in boundary-layer approximation, because the tube diameter is $\gg \delta_\kappa$. (We will discuss this approximation shortly.) The heat exchanger at the hot end of the pulse tube is the HX. The orifice and reservoir volume are easily modeled as a DeltaE IMPEDance and COMPLiance, respectively. Our use of zero for the imaginary part of the IMPEDance reflects our intention that this orifice will truly be resistive, with pressure drop in phase with mass flux.

For purposes of illustration here, we regard the geometry of the apparatus as given, and will explore its performance. The vector summary indicates our point of view:

```
Iteration Vectors Summary:
 GUESS      0c          0f          0g          1e          3e        6e
 name  BEGIN:T-beg BEGIN: |U|  BEGIN:Ph(U)    SX:HeatI    SX:HeatI  HX:HeatI
 units     K          m^3/s        deg           W           W         W
 value   300.10     7.1461E-03    50.477    -491.47       5.4025   -60.005
TARGET      1f          3f          5f          9a          9b        9c
 name     SX:Est-T    SX:Est-T    SX:Est-T HARDE:R(1/z HARDE:I(1/z HARDE:Hdot
 units       K           K          K                                  W
 value    3.00E+02    1.50E+02    3.00E+02    0.00        0.00      0.00
```

Three of the 5 targets fix the hot and cold temperatures at 300 K and 150 K. We leave the amplitude of the oscillatory pressure at the BEGINning at 8% of mean pressure, and leave the frequency fixed at 300 Hz. Hence, we are asking: What is the cooling power at 150 K, and how much input power, volumetric flow rate, etc. are required, for fixed frequency and pressure amplitude? The result, given in the file listings above: 5.4 W of cooling power, requiring 546 W of input power from the compressor.

Our choice of $\mathrm{Re}(Z_s) = 1 \times 10^7$ for the orifice impedance above was random. To find a better orifice setting, we can use $\mathrm{Re}(Z_s)$ as an independent plot variable, letting it range from 1×10^7 to 1×10^8. The cooling power peaks at 7.6 W for $\mathrm{Re}(Z_s) = 4.7 \times 10^7$.

As with most OPTR models we have worked with in DeltaE, this one is not particularly "robust." A change of a typical variable by 20% or 30% will likely cause DeltaE to get hopelessly lost. Hence the steps we used in the plotting of $\mathrm{Re}(Z_s)$ were small: 1×10^6. Part of the "'fragility" of OPTR models (as compared to thermoacoustic models) in DeltaE is due to the fact that small changes in variables near the `BEGINning`, such as p_1, U_1, and the heat removed at the aftercooler, have a large effect on temperatures at the end of the pulse tube. Some of the fragility is due to the fact that OPTR models typically have a large number of guesses and targets. When you encounter a fragile DeltaE model, try to reduce the number of guesses and targets as much as possible (particularly in initial design explorations when you are more lost than DeltaE) and, once you have a convergent model, make only small changes in variables. Tighten up DeltaE's convergence tolerance if you have to use more than 5 guesses and 5 targets. To accomplish a large change in a variable, use `(p)lot` to break it up into many small steps. A fast computer and frequent hard-disk saving of satisfactory converged models will minimize frustration.

Examination of the pulse tube segment in the `.dat` file above shows a possible problem: The pulse tube figure of merit, which Radebaugh defines as $\dot{H}/\dot{W}$, is high: $\dot{H}/\dot{W} \simeq$ 60.0 W/62.5 W$\simeq$ 0.95. A more common experimental value of pulse tube figure of merit is 0.7. DeltaE knows nothing about jet- or streaming-driven convection, and pulse-tube experimentalists are only beginning to learn how to reliably avoid such convection. For a discussion of streaming-driven convection, see Refs. [13] or [14], and [15].

To force DeltaE to accomodate a reduced pulse tube figure of merit, you can introduce an `RPNTARGET` and an additional guess/target pair. The `RPNTARGET` instruction line can compute the ratio of $\dot{H}_2$ to $\dot{E}_2$ in the pulse tube, and the target value in line "a" can be set to something like 0.7. You can simulate the thermal loading of streaming-driven convection, etc. by letting DeltaE guess an unphysically large value for the cross section of the pulse tube metal wall (line 4d), which then conducts significant heat from hot to cold, allowing DeltaE to meet its target of 0.7. We will not do so here.

We can improve the overall performance of this refrigerator by a simple means, similar in principle to the second orifice of a double-inlet pulse tube refrigerator: adding a small duct between the orifice and reservoir volume adds inertance to the impedance of the end of the system; proper choice of the length/area of this duct can phase shift the mass flow through the orifice significantly. This is entirely analogous to putting an inductor in series with an RC circuit, and is well known in the pulse-tube community. Adding an inertance to our model, and adjusting its area/length and Re(Z_s) of the orifice for maximum cooling power brings the cooling power up to 11.8 W in the `.out` file below.

```
TITLE     an early cooler design, not optimal
!---------------------------------  0 ---------------------------------
 BEGIN      Start with 8% p osc
  3.0000E+06 a Mean P    Pa              300.10    A T-beg  G( 0c)      P
   300.00    b Freq.     Hz            7.1857E-03 B  |U|   G( 0f)      P
   300.10    c T-beg     K       G       52.308    C Ph(U)  G( 0g)      P
  2.4000E+05 d  |p|      Pa            -473.81    D HeatIn G( 1e)      P
     0.0000  e Ph(p)     deg             11.820    E HeatIn G( 3e)      P
  7.1857E-03 f  |U|     m^3/s    G      -65.216    F HeatIn G( 6e)      P
    52.308   g Ph(U)     deg     G
 helium     Gas type
 ideal      Solid type
!---------------------------------  1 ---------------------------------
 SX         Aftercooler
  1.0290E-03 a Area      m^2            2.2942E+05 A |p|       Pa
     0.6900  b VolPor                    -3.7072   B Ph(p)     deg
  1.2500E-02 c Length     m             6.0644E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m               46.419   D Ph(U)     deg
  -473.81    e HeatIn     W      G        53.395   E Hdot      W
   300.00    f Est-T      K    = 1H?     445.98    F Edot      W
 helium     Gas type                    -473.81    G Heat       W
 copper     Solid type                   300.00    H MetalT     K
!---------------------------------  2 ---------------------------------
 STKSCRN    Regenerator
 sameas  1a  a Area      m^2            1.6913E+05 A |p|       Pa
     0.7300  b VolPor                    -21.847   B Ph(p)     deg
  5.5000E-02 c Length     m             1.1990E-03 C |U|     m^3/s
  2.4000E-05 d   r_H      m              -23.612   D Ph(U)     deg
     0.3000  e KsFrac                     53.395   E Hdot      W
                                         101.34    F Edot      W
                                         300.10    G T-beg      K
 helium     Gas type                     149.92    H T-end      K
 stainless  Solid type                  -344.63    I StkEdt     W
!---------------------------------  3 ---------------------------------
 SX         Cold heat exchanger
 sameas  4a  a Area      m^2            1.1295E+05 A |p|       Pa
     0.6900  b VolPor                    -21.520   B Ph(p)     deg
  2.0000E-03 c Length     m             1.1980E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m              -23.928   D Ph(U)     deg
    11.820   e HeatIn     W      G        65.216   E Hdot      W
   150.00    f Est-T      K    = 3H?      67.600   F Edot      W
 helium     Gas type                      11.820   G Heat       W
 copper     Solid type                   150.00    H MetalT     K
!---------------------------------  4 ---------------------------------
 STKDU      Pulse tube
  5.6870E-05 a Area      m^2     S=-2   1.1299E+05 A |p|       Pa
  2.6740E-02 b Perim      m    Fnc( 4a)  -48.674   B Ph(p)     deg
     0.2000  c Length     m             1.1917E-03 C |U|     m^3/s
  1.0000E-05 d WallA     m^2             -47.588   D Ph(U)     deg
                                          65.216   E Hdot      W
                                          67.318   F Edot      W
                                         149.92    G T-beg      K
 helium     Gas type                     300.21    H T-end      K
 stainless  Solid type                   -0.2825   I StkEdt     W
!---------------------------------  5 +++++++++ therm insul mode +++++++++
 INSULATE   assume the orifice and compliance are insulated
                                        1.1299E+05 A |p|       Pa
                                         -48.674   B Ph(p)     deg
                                        1.1917E-03 C |U|     m^3/s
                                         -47.588   D Ph(U)     deg
                                          65.216   E Hdot      W
                                          67.318   F Edot      W
```

```
                                            0.0000  G HeatIn    W
!---------------------------------  6 +++++++++ therm insul mode +++++++++
 SX         Hot heat exchanger
 sameas  4a  a Area      m^2            4.1777E+04 A |p|       Pa
     0.6900  b VolPor                    -52.215   B Ph(p)     deg
  5.0000E-03 c Length     m             1.1904E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m              -48.036   D Ph(U)     deg
   -65.216   e HeatIn     W      G      -1.7318E-08 E Hdot      W
   300.00    f Est-T      K    = 6H?      24.799   F Edot      W
 helium     Gas type                     -65.216   G Heat       W
 copper     Solid type                   300.00    H MetalT     K
!---------------------------------  7 +++++++++ therm insul mode +++++++++
 IMPEDANCE  The orifice
  3.5000E+07 a Re(Zs) Pa-s/m^3          3044.3     A |p|       Pa
     0.0000  b Im(Zs) Pa-s/m^3          -137.98    B Ph(p)     deg
                                        1.1904E-03 C |U|     m^3/s
                                         -48.036   D Ph(U)     deg
                                       -1.7318E-08 E Hdot      W
 sameas  0  Gas type                    1.7313E-03 F Edot      W
 ideal      Solid type                   -24.797   G HeatIn    W
!---------------------------------  8 +++++++++ therm insul mode +++++++++
 DUCT       inertance
  2.8000E-02 a Area      m^2            3051.3     A |p|       Pa
     0.5932  b Perim      m             -137.98    B Ph(p)     deg
  3.1620E-02 c Length     m             1.7223E-04 C |U|     m^3/s
  3.0000E-04 d Srough                    -48.166   D Ph(U)     deg
                                       -1.7318E-08 E Hdot      W
 sameas  0  Gas type                    8.4762E-04 F Edot      W
 ideal      Solid type                     0.0000  G HeatIn    W
!---------------------------------  9 +++++++++ therm insul mode +++++++++
 COMPLIANCE Reservoir volume
  1.2680E-02 a SurfAr    m^2   Fnc( 9b)  3051.3     A |p|       Pa
  1.5000E-04 b Volum     m^3     S=-5    -137.98    B Ph(p)     deg
                                        3.6063E-07 C |U|     m^3/s
                                         159.11    D Ph(U)     deg
                                       -1.7318E-08 E Hdot      W
 sameas  0  Gas type                    2.5056E-04 F Edot      W
 ideal      Solid type                     0.0000  G HeatIn    W
!--------------------------------- 10 +++++++++ therm insul mode +++++++++
 HARDEND    The end
     0.0000  a R(1/z)          =10G?     3051.3     A |p|       Pa
     0.0000  b I(1/z)          =10H?     -137.98    B Ph(p)     deg
     0.0000  c  Hdot    W      =10E?     3.6063E-07 C |U|     m^3/s
                                         159.11    D Ph(U)     deg
                                       -1.7318E-08 E Hdot      W
                                        2.5056E-04 F Edot      W
                                        2.0818E-05 G R(1/z)
 helium     Gas type                   -4.0698E-05 H I(1/z)
 ideal      Solid type                   300.21    I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      6   0  3   0  6   0  7   1  5   3  5   6  5
TARGS       6   1  6   3  6   6  6  10  1  10  2  10  3
SPECIALS    2   4 -2   9 -5
```

Beginning with DeltaE version 4.10, the `JOIN` segment can be used at the ends of a pulse tube to model the temperature overshoots and adiabatic-isothermal interface losses there. See Chapter VI, Section B.

Tektronix researchers used DeltaE to model a 350 Hz orifice pulse tube refrigerator, as described in Ref. [16].

D. Etched Foil Regenerators

The segment `STKPOwerlaw` is intended to model regenerators for which the friction factor and heat transfer coefficient are power laws in Reynold's number. This includes etched foil regenerators, as described in [17]. For input syntax and mathematical details, see the end of section VI.B.5.

V. Advanced Features

This chapter introduces additional features of DeltaE that expand its power and convenience for the user who is already comfortable with the basics. Here we explain "math segments" that allow the increased control over the endpoints of DeltaE's iterations and incorporate some basic math functions; active branches that permit simultaneous calculation of side branches and main ducts for complicated models; additional fluid options, including binary gas mixtures; parameter linking so that iterations can be performed while maintaining certain geometric relationships in the model; and several other useful features and tunable parameters.

A. RPNTArgets (Math Segments)

DeltaE reserves a place for a special input parameter to hold a target value in the segment types that have outputs commonly used in targets. These parameters are: heat exchanger temperatures and heat flows, and complex impedances in `HARD`- and `SOFTEnd` segments (`UNION`s, introduced in the next section, are a special case). The code knows, internally, to pair these input values with the appropriate output results of the segment for comparison. The experienced user, however, will soon hunger for more possibilities once a model is defined and converging to meet these basic targets. An application may call for acoustic power, pressure, or velocity (magnitude or phase) to be specified at a certain location, or some derived function of outputs may be desired for targeting or plotting. Math segments are used for these purposes. We most often use them to generate a new type of output based on other outputs in the model; in this case, the first 'target' parameter is simply ignored. Math segments have one real input and one real output which DeltaE recognizes as a potential target/result pair. The other input parameters to these segments are one or more addresses, or, in the case of `RPNTArget`'s, a symbolic equation that can include addresses.

There are 8 types of math segments: `RPNTArget`, `FREETarget`, `QUOTArget`, `COPRTarget`, `EFFRTarget`, `PRODTarget`, `DIFFTarget`, and `VOLMTarget`. We highly recommend `RPNTArget` because it is much more versatile than the other math segments; it can be written to act like any of the other math segments (except `VOLMTarget`). For complicated functions, `RPNTArget` targets can be cascaded. The other, obsolete math segments are still included in DeltaE for backwards compatibility. Their use is described only in Chapter VI.

The "target" parameter of math segments, like that of any other targets, can be used as the independent variable in a plotting loop. Math segments should be placed after all the results that they reference in the model, so that, during each pass of DeltaE through the segments (which are always treated sequentially), the math segments will be updated with the most recent results. Math segments do *not* end with fluid and solid names like other segments.

`RPNTArget` is a new feature (beginning in version 3.9), sufficiently versatile that it effectively makes all other math segment types obsolete. `RPNTArget` allows complicated mathematics, and can perform such mathematics on inputs (including guesses) from other segments and on constants, as well as on results from other segments.

The first line of an `RPNTArget` segment (line "a") is the target parameter. It may also be used as a convenient location for a frequently changed constant or a guess when it is referenced within the formula of the segment. The second line (line "b") is an instruction line expressing the desired algebraic procedure in Reverse Polish Notation, the parenthesis-free algebra encoding technique used by Hewlett-Packard pocket calculators. For example, to generate the particle displacement amplitude

$$|\xi_1| = \frac{|U_1|}{\omega A} \tag{V.1}$$

at the end of, say, segment 5, an `RPNTArget` after segment 5 could be written

```
!-------------------------- 6 ------------------
RPNTARGET   magU1 over omega A (units: meters)
0.01
5C 2 / PI / 0b / 5a /
```

Those familiar with Reverse Polish Notation will recognize that the instruction line could just as well be written

```
5C 2 PI * 0b * 5a * /
```

or even

```
5C 2 PI 0b 5a * * * /
```

Extensions to the code as of version 4.3 also enable us to write the instruction line like this:

```
U1 mag w / 5a /
```

Here we exploit the internal state variables `w` (ω) and `U1` (U_1). In the case of the latter, it is important that no physical segments (ducts, stacks, etc.) are present between the point of interest and the `RPNTARGET`, as these would alter the volume flow rate and other state variables. A complete list of available state variables and also local thermophysical parameters (density, sound speed, etc.) is given Table 6.2.

A complete lesson in Reverse Polish Notation can be found in instruction manuals for most HP pocket calculators. Briefly, as numbers are encountered in the instruction line, they are pushed onto a stack; when a unary operator such as `cos`, `log`, or `sqrt` is encountered, it pops a single number off the stack, acts on it, and pushes the result onto the stack; when a binary operation such as + or * is encountered, it pops two numbers off the stack, combines them appropriately, and pushes the result back onto the stack. When a simple instruction line has been processed, there is only one number remaining on the stack; this number is the "result" of the `RPNTArget` segment. (It is sometimes desirable to intentionally leave intermediate results on the stack to be accessed elswhere. This is perfectly permissible, and will be demonstrated in an example, below.)

To further illustrate the use of `RPNTArget`s, consider how one could compute engine efficiency relative to Carnot's efficiency. (This can also be accomplished with the now-obsolete `EFFRTarget`, described in Sec. VI B.7.) If W =`5F`, Q_h =`3G`, T_h =`3H`, and T_c =`5H`, $\frac{W}{Q_h}\frac{T_h}{T_h - T_c}$ becomes

```
!-------------------------------- 10 --------------------------
 RPNTARG      Efficiency/Carnot Efficiency
   0.000     a Target
 5F 3G / 3H 3H 5H - / *
```

We could also have written

```
 5F 3G / 3H # 5H - / *
```

using the stack push operator, `#`.

For a more complicated use, suppose we are interested in COPR$^2 Q_c$ as an overall figure of merit for a refrigerator, where Q_c is cooling power and COPR is the coefficient of performance relative to Carnot's coefficient of performance. We can generate both COPR and the product COPR$^2 * Q_c$ as follows, utilizing the addresses used in the refrigerator example in Sec. III.C:

```
!-------------------------------- 10 --------------------------
 RPNTARG      COP/COP_Carnot and itself squared times cooling power
   0.000     a Target           (t)       8.45       A RPNval
                                          0.0130     B RPNval
 5G 1F / 3H 5H - 5H / * # SQRD 5G *
```

This example demonstrates how the stack, composed of the segment's output parameters, grows when the number of "pushes" exceedes the number of stack popping operations. In this case '#', the push command, causes the COPR value to be duplicated and pushed to parameter `B`. The copy in parameter `A` (also equal to 0.0130) is overwritten by the `SQRD` operation, then multiplied by `5G` (Q_c).

For another example, illustrating some interesting links between and within DeltaE segments, consider a turbulent flow restriction, which has a pressure drop proportional to the square of velocity. To model this, we would like an acoustic impedance whose real part is proportional to the velocity through it. To cause DeltaE to accomplish this, use

```
!--------------------------------  3 --------------------------
 RPNTARG  Line (a) depends on rho and the geom of the restriction
  2.2500E+08 a Target          (t)      2.2500E+08 A RPNval
 3a  2C *
!--------------------------------  4 --------------------------
 IMPEDANCE    this mimics an orifice with turbulent pressure drop
 sameas 3A  a Re(Zs) Pa-s/m^3
    0.0000  b Im(Zs) Pa-s/m^3
 sameas  0  Gas type
 ideal      Solid type
```

DeltaE allows for complex arithmetic within `RPNTARGET`s, though all other input and output paramters are real. It is possible to implement a calculation such as finding f_k for a parallel-plate stack (compare with the first of Eqs. VI.22). We make the input parameter, `3a`, equivalent to the plate half-spacing y_0 here:

```
!--------------------------------  3 --------------------------
 RPNTARG  Thermal function for parallel plate geom
  2.0000E-04 a Target          (t)     (   0.9244 ,  -0.2392 )  A
 3a dk / (1,1) * tanh lstx /
```

Here `(1,1)` illustrates the entry of a complex constant, $(1+i)$. δ_κ is another of the local state variables that can be accessed, through `dk`).

DeltaE's `RPNTArget` segment recognizes the common trigonometric and other special functions present on most pocket calculators, as well as hyperbolic and Bessel functions that are not readily available. Most functions accept complex arguments. Other operators are given to convert a complex output to a real number in the usual coordinates (real, imaginary, magnitude, phase). For a complete listing, see Tables 2–4 in Chapter VI.

Whenever the instruction line of an `RPNTArget` is entered during an `(I)nsert` operation or edited during a `(m)odify` operation, entering a response of '?' will print a summary similar to the following:

```
Valid RPN commands are:
Basic:   +     -     *     /     ^     ~     inv
```

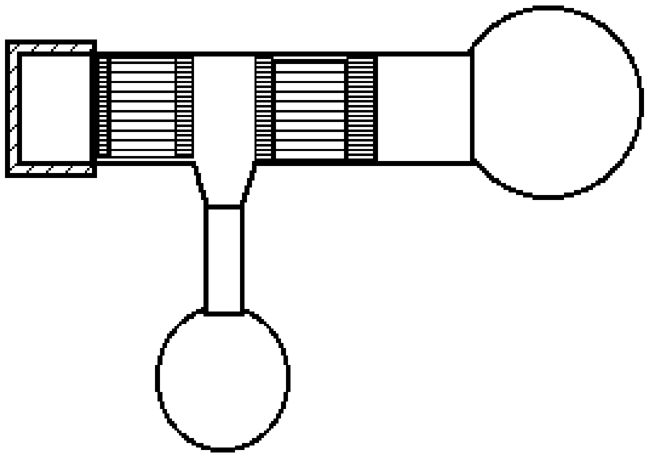


Figure V.1: Modified “beer cooler.”

```
          log   exp   exp   tenx  sqrt  sqrd  abs
Trig:     cos   acos  sin   asin  tan   atan  pi
          cosr  acosr sinr  asinr tanr  atanr sinh  cosh  tanh
Bessel:   i     J0    J1    Y0    Y1
Stack:    #     lstx  a<>b  sto   rcl   min   max
Complex:  mag   real  imag  arg   argr  conj  i
State:    tm    w     f     pm    n1    p1    U1
Thermo:   gamma a     rho   cp    k0    mu    beta  dk    dn    rhos  cs    ks
INVALID:  @@Z   ====  NoOp   (error codes)
```

and then repeat the input prompt. The error codes in the last line are keys to what DeltaE writes when its parser is unable to intepret instruction strings.

B. Active Branches

Although `BRANCh` and `OPNBRanch` have their uses, they are often inadequate for describing the variations in branch impedance with operating conditions. For example, the branch might be a Helmholtz resonator whose impedance changes significantly with frequency. Further, `BRANCh` and `OPNBRanch` are wholly inadequate when branches involve thermoacoustic components. The `TBRANch` segment addresses these inadequacies by allowing DeltaE to integrate its way down a side branch through a series of segments and then return to the trunk and continue integrating there as before.

As an example, consider the modification shown in Fig. V.1 to the basic “beer cooler” (heat-driven thermoacoustic refrigerator) shown in Fig. I.7. We might want to investigate whether performance would improve by adding the side branch so that the entire volume flow rate required by the prime-mover stack would no longer have to flow through the refrigerator stack and much of the resonator dissipation would show up at ambient temperature instead of at the cold heat exchanger.

DeltaE uses the TBRANch segment for cases like this. When it encounters a TBRANch, DeltaE treats subsequent segments as the sequential members of the branch, until it reaches a HARDEnd or SOFTEnd. It then "returns to the trunk," treating the rest of the segments as trunk members. So the sequence of segments for the example of Fig. V.1 might be as follows:

```
TITLE
BEGIN      0
ENDCAP     1
DUCT       2
HX         3
STKSLAB    4
HX         5
TBRANCH    6
  CONE       7
  DUCT       8
  COMPLIANCE 9
  HARDEND   10
HX        11
STKSLAB   12
HX        13
DUCT      14
COMPLIANC 15
HARDEND   16
```

Segments 6 through 10 comprise the side branch; the others comprise the trunk.

The method of computation is as follows. At a branch, the branch impedance determines how the (complex) volume flow rate splits up at the branch. Often, we use the branch impedance as a pair of guesses that DeltaE adjusts in its usual way to get the complex impedance at the next 'END to come out right. (If asked to do so, DeltaE should select both of these guess and target pairs as part of a default set. If not, you should enable them.) TBRANCHed models tend to have guess and target vectors of high dimension, since every 'END contributes two targets (and a few more targets are almost always needed for temperatures, heats, etc.). Stacks and heat exchangers can also be used in branches, and, of course, branches can have subbranches of their own.

TBRANch has a companion segment type, TEE, that takes the filename of another valid DeltaE input file as its only parameter. When DeltaE encounters a TEE, it loads the named file into the model, and replaces the BEGIN segment of the branch file with a TBRANch segment. It tries to guess starting values for the complex branch impedance, and then adjusts the addresses in any sameas declarations and math segments occurring in the branch (or after the branch point) by the number of segments in the branch. Once the file has been read in, the TEE segment disappears—the .out file and (d)isplayed segments will be the composite model.

When rewriting our previous example to use a TEE segment, the model has the form

```
TITLE
```

```
BEGIN       0
ENDCAP      1
DUCT        2
HX          3
STKSLAB     4
HX          5
TEE         6
 branch.in
HX          7
STKSLAB     8
HX          9
DUCT       10
COMPLIANC  11
HARDEND    12
```

where we have omitted the parameters of all but the `TEE` segment. The file `branch.in` is a valid DeltaE input file, which we have run and debugged separately, and which looks like this:

```
TITLE
BEGIN       0
CONE        1
DUCT        2
COMPLIANCE  3
HARDEND     4
```

The file may have any name (*e.g.*, `branch.in, branch.out, branch.tee`), but it must be specified with the complete suffix.

The two models above will combine to produce the same model as our first example. This approach is recommended, especially for nontrivial branches containing stacks, etc., so that the two simpler submodels can be evaluated first. The impedance that DeltaE chooses for the `TBRANch` may need immediate attention; guess and target vectors, math segments, and `sameas` references should also be checked carefully. Special modes (see below) that link length parameters across the branch point will not be handled properly, and must be redone with new segment numbers.

The multiply-connected duct network of Fig. I.3 can also be handled by DeltaE, through use of `TBRANch` and `UNION`. The `UNION` segment is used to tell DeltaE to "connect" a `TBRANch`'s `SOFTEnd` (or `HARDEnd`) back to the trunk at the location of the `UNION` segment. The branch's `SOFTEnd` impedance targets are no longer used; instead, the two real input variables (`b` and `c`) of the `UNION` segment should always be active targets. It does not matter what the initial values of these parameters are; as soon as DeltaE processes the segment, it copies in the current values of the complex pressure at the `SOFTEnd` referenced by the number in parameter `a` of the `UNION` segment. These values are compared to the local complex pressure result, at this `UNION`, in the trunk, and iteration should drive the model until their difference is zero. In other words, when a branch and trunk meet at a `UNION`, they must share the same complex p_1. As before, a guessed branch impedance usually determines how the (complex) volume flow rate splits up at the `TBRANch`. Volume flow rates

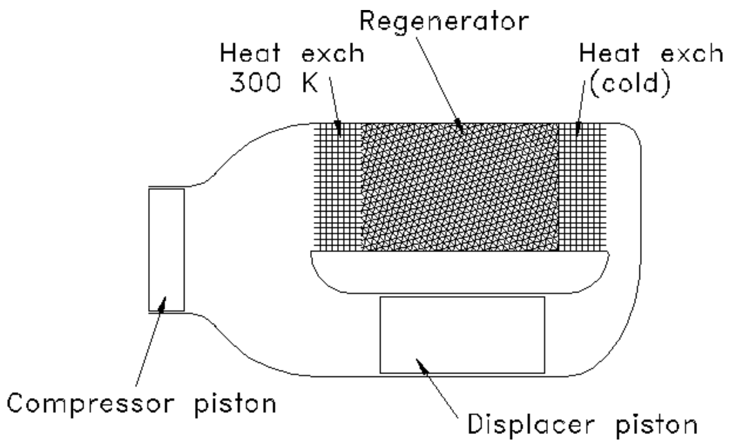


Figure V.2: “Gamma”-style Stirling machine.

are summed at the `UNION`. (The `UNION` segment is somewhat similar to the math segments of the previous section, except that it grabs two results simultaneously, from fixed locations within the referenced segment. Also, the ‘target’ values are not specified by the user, since they vary dynamically depending on what is happening at the attached ’`End` segment.)

As an example of use of `TBRANch` and `UNION`, we return to the Stirling cryocooler example of the previous chapter, and convert it to a “gamma” style Stirling machine, with a compressor piston at the hot end and a displacer piston connecting the hot and cold ends. In the previous example, *PU* power flowed in at the `BEGIN` and out at the `...END`; with a displacer piston, the cold-end *PU* power is returned automatically to the hot end, reducing the hot-end *PU* power requirement.

The apparatus layout is illustrated in Fig. V.2; the corresponding DeltaE file is gamma.in, its layout is

```
TITLE
BEGIN
TBRANCH
   IESPEAKER (the displacer)
   SOFTEND
SX
STKSCREEN
SX
UNION  ('connects' to softend above)
HARDEND
```

and the corresponding `.out` file is

```
TITLE     Stirling cooler w displacer piston, illustrating TBRANCH--UNION
!---------------------------------  0 ---------------------------------
 BEGIN     Initialize things
  2.0000E+06 a Mean P    Pa              300.14    A T-beg  G( 0c)      P
    55.000   b Freq.     Hz            2.8533E+05 B  |p|    G( 0d)      P
   300.14    c T-beg     K       G       -42.649   C Ph(p)  G( 0e)      P
  2.8533E+05 d  |p|       Pa      G      -4.5438E+09 D Re(Zb) G( 1a)      P
   -42.649   e Ph(p)     deg     G      -7.4853E+08 E Im(Zb) G( 1b)      P
  3.3000E-04 f  |U|     m^3/s             1255.7     F   K    G( 2f)      P
     9.0000  g Ph(U)     deg              4.2171E-02 G   Rm   G( 2g)      P
 helium     Gas type                      -35.859   H HeatIn G( 4e)      P
 ideal      Solid type
!---------------------------------  1 ---------------------------------
 TBRANCH    branch to displacer
 -4.5438E+09 a Re(Zb) Pa-s/m^3   G       2.8533E+05 A |p|       Pa
 -7.4853E+08 b Im(Zb) Pa-s/m^3   G        -42.649   B Ph(p)     deg
                                         6.1959E-05 C |U|     m^3/s
                                          128.00    D Ph(U)     deg
                                           -8.7218  E Hdot      W
 sameas  0  Gas type                       -8.7218  F Edot      W
 ideal      Solid type                     37.933   G Edot_T    W
!---------------------------------  2 ---------------------------------
 IESPEAKer  a spring-mounted, driven moving mass
  1.0000E-05 a Area      m^2             2.2888E+05 A |p|       Pa
     1.0000  b   R      ohms              -52.607   B Ph(p)     deg
     0.0000  c L          H              6.2000E-05 C |U|     m^3/s
     1.0000  d B x L    T-m               128.00    D Ph(U)     deg
  1.0000E-02 e   M      kg                 -7.0950  E Hdot      W
  1255.7     f   K      N/m      G         -7.0950  F Edot      W
  4.2171E-02 g   Rm   N-s/m      G          2.9421  G WorkIn    W
     1.0000  h  |I|     A                   7.0133  H Volts     V
   -90.000   i Ph(I)   deg                  1.0000  I Amps      A
                                           32.964   J Ph(Ze)   deg
                                         7.1788E+04 K  |Px|    Pa
 sameas  0  Gas type                      170.81    L Ph(Px)   deg
 ideal      Solid type                     -1.3159  M HeatIn    W
!---------------------------------  3 ---------------------------------
 SOFTEND    reconnect at UNION
     0.0000  a Re(z)              (t)    2.2888E+05 A |p|       Pa
     0.0000  b Im(z)              (t)     -52.607   B Ph(p)     deg
                                         6.2000E-05 C |U|     m^3/s
                                          128.00    D Ph(U)     deg
                                           -7.0950  E Hdot      W
                                           -7.0950  F Edot      W
                                          -11.289   G Re(z)
 sameas  0  Gas type                        0.1196  H Im(z)
 ideal      Solid type                    300.14    I   T       K
!---------------------------------  4 ---------------------------------
 SX         aftercooler
 sameas  5a  a Area      m^2             2.8160E+05 A |p|       Pa
     0.6000  b VolPor                     -43.360   B Ph(p)     deg
  1.0000E-03 c Length     m              3.6173E-04 C |U|     m^3/s
 sameas  5d  d   r_H      m              4.7835E-02 D Ph(U)     deg
   -35.859   e HeatIn     W      G          2.0740  E Hdot      W
   300.00    f Est-T      K    = 4H?       37.001   F Edot      W
 sameas  0  Gas type                      -35.859   G Heat       W
 copper     Solid type                    300.00    H MetalT     K
!---------------------------------  5 ---------------------------------
 STKSC      regenerator
  1.1670E-04 a Area      m^2             2.2929E+05 A |p|       Pa
     0.6860  b VolPor                     -52.602   B Ph(p)     deg
  5.0000E-02 c Length     m              6.2199E-05 C |U|     m^3/s
  1.3900E-05 d   r_H      m               -49.476   D Ph(U)     deg
     0.3000  e KsFrac                        2.0740  E Hdot      W
                                            7.1203  F Edot      W
                                          300.14    G T-beg      K
 sameas  0  Gas type                       79.960   H T-end      K
 stainless  Solid type                    -29.881   I StkEdt     W
!---------------------------------  6 ---------------------------------
 SX         cold heat exch
 sameas  5a  a Area      m^2             2.2888E+05 A |p|       Pa
     0.6000  b VolPor                     -52.607   B Ph(p)     deg
   1.0000E-03 c Length     m              6.2000E-05 C |U|     m^3/s
  sameas  5d  d   r_H      m               -52.000   D Ph(U)     deg
     0.0000  e HeatIn     W                 7.0950  E Hdot      W
    80.000   f Est-T      K     = 6H?       7.0950  F Edot      W
  sameas  0  Gas type                       5.0210  G Heat       W
  copper     Solid type                     80.000  H MetalT     K
 !---------------------------------  7 ---------------------------------
  UNION      displacer cold end
     3.0000  a TendSg                     2.2888E+05 A |p|       Pa
   2.2888E+05 b |p|End    Pa    = 7A?      -52.607   B Ph(p)     deg
    -52.607   c Ph(p)E    deg   = 7B?     4.5040E-18 C |U|     m^3/s
                                           -66.598   D Ph(U)     deg
                                          5.0016E-13 E Hdot      W
  sameas  0  Gas type                     5.0016E-13 F Edot      W
  ideal      Solid type                      0.0000  G End-T     K
 !---------------------------------  8 ---------------------------------
  RPNTARGET  displacer U
   6.2000E-05 a Target          = 8A?     6.2000E-05 A RPNval
   6C
 !---------------------------------  9 ---------------------------------
  RPNTARGET  displacer phase
    -52.000   a Target          = 9A?      -52.000   A RPNval
   6D
 !--------------------------------- 10 ---------------------------------
  HARDEND    the end             10
     0.0000  a R(1/z)          =10G?     2.2888E+05 A |p|       Pa
     0.0000  b I(1/z)          =10H?      -52.607   B Ph(p)     deg
                                          4.5040E-18 C |U|     m^3/s
                                           -66.598   D Ph(U)     deg
                                          5.0016E-13 E Hdot      W
                                          5.0016E-13 F Edot      W
                                          1.0366E-15 G R(1/z)
  helium     Gas type                    -2.5827E-16 H I(1/z)
  ideal      Solid type                     79.960   I   T       K

 ! The restart information below was generated by a previous run
 ! You may wish to delete this information before starting a run
 ! where you will (interactively) specify a different iteration
 ! mode.  Edit this table only if you really know your model!
 INVARS      8   0  3   0  4   0  5   1  1   1  2   2  6   2  7   4  5
 TARGS       8   4  6   6  6   7  2   7  3   8  1   9  1  10  1  10  2
 SPECIALS    0
```

We are using an IESPEAker segment as the displacer piston, because a linear motor and a loudspeaker share the same physical transduction mechanism.

Our guess/target vector summary is the largest we have yet encountered in this user's guide—eight each:

```
Iteration Vectors Summary:
 GUESS      0c          0d          0e          1a          1b          2f
 name  BEGIN:T-beg BEGIN: |p|  BEGIN:Ph(p) TBRAN:Re(Zb TBRAN:Im(Zb IESPE:  K
 units     K           Pa          deg      Pa-s/m^3    Pa-s/m^3      N/m
 value    3.00E+02    2.86E+05    -43.       -4.55E+09   -7.62E+08   1255.7
 GUESS      2g          4e
 name  IESPE: Rm   SXFRS:HeatI
 units   N-s/m          W
 value   -7.87E+08    -36.
TARGET      4f          6f          7b          7c          8a          9a
 name     SX:Est-T    SX:Est-T UNION:|p|En UNION:Ph(p) RPNTA:Targe RPNTA:Targe
 units      K           K          Pa          deg
 value    3.00E+02     80.        2.29E+05    -52.         6.20E-05    -52.
result     .00         .00         .00         .00          .00         .00
TARGET     10a         10b
 name  HARDE:R(1/z HARDE:I(1/z
 units
 value     .00         .00
result     .00         .00
```

One can think of these guesses and targets as paired up in the following way. The `T-begin` guess lets DeltaE hit the T-hot target; these two are so nearly equal, and so trivially related, that they could easily be dropped from the vectors if necessary. The two branch-impedance guesses and the two `IESPEAKER` guesses let DeltaE reach four targets: the two p targets at the `UNION` and the two U `FREETarget`s that essentially determine the displacer piston's motion. The heat removed at the hot heat exchanger determines the temperature at the cold heat exchanger. Finally, the two p guesses in the `BEGIN` segment allow DeltaE to achieve $U = 0$ at the `HARDENd` at the end of the apparatus.

Running this file produced the `.out` file listed above, and the `.dat` file below:

```
-= Stirling cooler w displacer piston, illustrating TBRANCH--UNION          =-
 frequency=      55.000Hz     mean pressure=   2.000E+06Pa

   Tm (K)      Re & Im p1 (Pa)      Re & Im U1(m3/s)        Hdot(W)   Edot(W)
   300.1   209863.6   -193308.3      0.00033  0.00005          29.21     29.21
!----------------------------------- 1 ------------------------------------
TBRANCH   branch to displacer
 TTTTTTTTTTTTTTTTTTTT Branching into Tee Level= 1 TTTTTTTTTTTTTTTTTTT
   300.1   209863.6   -193308.3     -0.00004  0.00005          -8.72     -8.72
!----------------------------------- 2 ------------------------------------
IESPEAKer a spring-mounted moving mass
 (    3.82     ,   -5.88     ) Volts,(  -4.371E-08,   -1.00    ) Amps
 Heat extracted: I^2 R= 0.500    , u^2 Rm= 0.811    , B-Layer= 5.322E-03 Watts.
   300.1   138996.6   -181846.5     -0.00004  0.00005          -7.10     -7.10
!----------------------------------- 3 ------------------------------------
SOFTEND   reconnect at UNION
 impedance (p A/rho a U)=(   -11.3     ,    0.120    )
   300.1   138996.6   -181846.5     -0.00004  0.00005          -7.10     -7.10
 TTTTTTTTTTTTTTTTTTTT Returning to Trunk Level= 0 TTTTTTTTTTTTTTTTTTT
   300.1   209863.6   -193308.3      0.00036  0.00000          37.93     37.93
!----------------------------------- 4 ------------------------------------
SX        aftercooler
 Heat =      -35.859 (W)    metal temp=     300.000 Kelvin
   300.1   204736.2   -193338.2      0.00036  0.00000           2.07     37.00
!----------------------------------- 5 ------------------------------------
STKSC     regenerator
    80.0   139260.2   -182160.8      0.00004 -0.00005           2.07      7.12
!----------------------------------- 6 ------------------------------------
SX        cold heat exch
 Heat =        5.021 (W)    metal temp=      80.000 Kelvin
    80.0   138996.6   -181846.5      0.00004 -0.00005           7.10      7.10
!----------------------------------- 7 ------------------------------------
UNION     displacer cold end
 Union P match difference=   9.022E-10 Pa;   1.421E-13 deg.
    80.0   138996.6   -181846.5      0.00000  0.00000           0.00      0.00
!----------------------------------- 8 ------------------------------------
RPNTARGET displacer U
RPN stack: = 6.2000E-05,
Opstring=  6C
    80.0   138996.6   -181846.5      0.00000  0.00000           0.00      0.00
```

```
!----------------------------------  9 ----------------------------------
RPNTARGET displacer phase
RPN stack: =  -52.000  ,
Opstring=  6D
    80.0   138996.6   -181846.5      0.00000  0.00000          0.00      0.00
!---------------------------------- 10 ----------------------------------
HARDEND   the end            10
 inverse impedance (rho a U/p A)=(   1.037E-15,  -2.583E-16)
    80.0   138996.6   -181846.5      0.00000  0.00000          0.00      0.00
```

The user might next generate cooling power curves by using the cold temperature target as an independent plot variable and the cooling power as dependent plot variable; or the user might explore the frequency dependence of the cooler, by using frequency as an independent plot variable; or the user might want to add more realism to the model by including the large dead volumes shown in the figure near the pistons. If inertial and viscous effects are presumed negligible in those volumes, they can be modeled as COMPLIANCEs:

```
TITLE
BEGIN
COMPLIANCE
   TBRANCH
   IESPEAKER (the displacer)
   SOFTEND
SX
STKSCREEN
SX
COMPLIANCE
UNION  ('connects' to softend above)
HARDEND
```

The user will soon discover that this is a surprisingly robust model, considering the large number of guesses and targets: the model tolerates steps in independent variables of several percent without getting lost.

TBRANCH and UNION are intended for duct networks, where temperature is constant and hence p_1 and U_1 are the variables of interest. For more complex systems, such as this one, the segments HBRANCH and HUNION are energy-conserving versions of TBRANCH and UNION. Definitely use them if you are branching at locations where $\dot{H} \neq \dot{W}$, such as at a branch to a second stage regenerator within a two-stage pulse tube refrigerator. HBRANCH incorporates a potential guess Hbran, giving the fraction of the incoming energy that goes into the branch. Use Hbran as a guess to hit a target down the branch, such as a temperature. HUNION incorporates an additional potential target, that the temperature in the trunk at the union be equal to that at the associated branch end.

C. Turbulence

A turbulence algorithm can be enabled in DUCTs and CONEs, by use of an otherwise hidden input parameter: parameter **d** in DUCTs and parameter **f** in 'CONEs, the relative roughness

(roughness height divided by pipe diameter). Set the roughness equal to zero for smooth walls, or to some small value greater than zero for rough walls. To ensure a laminar calculation, set the roughness equal to -1 (which will cause the parameter to be hidden once again).

The turbulence algorithm follows the quasi-steady approximation, the spirit of the assumptions of Iguchi *et al.*, Ref. [18]. It assumes that oscillatory-flow losses can be calculated by using the Moody friction factor (valid for steady flow) at each instant of time during the oscillatory flow. This assumption has little experimental validation in the range of Reynolds number and R/δ_ν of interest in thermoacoustics, but we believe it provides a useful estimate, better than no estimate at all. For more details, see Sec. VI B.1.

D. Variable Gas Mixtures

Several binary mixtures of gases have proven useful in thermoacoustic devices because of their improved Prandtl numbers and the option to adjust the resonance by changing the sound speed. DeltaE's fluid library contains three such mixtures: He-Xe, He-Ar, and He-Ne. These fluids are specified by a string on a line after a segment's numerical parameters, as are other fluids, but the string contains a 5-character field that represents the fraction of helium in the mixture (for example, `0.981hexe` or `0.889hear`, containing 98.1% and 88.9% helium, respectively).

If all but the first of the fluids (in the `BEGIN` segment) are specified using `sameas 0` statements, it is possible to use the helium fraction of the mixture as an iteration variable for resonance tuning. Simply select `0i` (Version 5) or `0j` (Version 6) from the `(u)se` menu option (it may instead be a plot variable, if you choose). In the `.out` file, the fluid written out will reflect the final concentration used.

Our equations for He-Xe properties are not valid for Xe fractions between 0.5 and 0.999.

E. User-Defined Fluid/Solid

DeltaE has a provision that allows users to specify 'external' fluids or solids that are not part of its internal library of thermophysical properties. Properties are derived, according to current operating conditions, from Eqs. V.2, V.3 and V.4 (below) using coefficients read from a user-written text file. Up to five distinct external fluids and five external solids can be used at one time.

The file can have any name valid under the operating system under which DeltaE is running, and it should end with the extension `.tpf`. If the root filename is the same as any predefined fluids, DeltaE will replace its internal calculations for that fluid with those given in the user file. To request a user-defined fluid, simply use the root file name as you would any other fluid. The `.tpf` file should be in the same directory or folder as the model file. The name of the fluid is set to the root filename of the external fluid file.

The file format is similar to the segment definitions we have used in models described in previous chapters in that comment lines can be added with an initial '!' and blank lines are ignored. Each property is specified by a line containing 1–10 real coefficients which are read in as C_{0-9}, where unused trailing parameters are set to zero. It is critical that the properties be arranged in this order: ρ, c_p, K, a^2, and μ. We also need the ratio of specific heats, γ, and the expansion coefficient β, but these are calculated internally from

$$\beta = -\frac{1}{\rho}\frac{\partial \rho}{\partial T} \quad \text{and} \quad \gamma - 1 = \frac{T\beta^2 a^2}{c_p}. \tag{V.2}$$

Each of the five properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1 \frac{p_m}{T + p_m C_2} + C_3 T + C_4 T^2 + C_5 T^{C_6} + C_7 p_m^2 T^{C_8} + p_m C_9, \tag{V.3}$$

where T and p_m are the absolute temperature (K) and mean pressure (Pa) for each point at which a segment using the fluid is evaluated.

Equation V.3 is a compromise between simplicity and flexibility; it is intended for use in a variety of simple expressions for gases and liquids and has a uniform syntax for specifying all 5 properties. There is only a limited mean pressure dependence, suitable for nearly ideal gases; for more complicated mean pressure dependence, multiple `.tpf` files should be written for each mean pressure range used.

To illustrate the use of these coefficients, consider the example below. To replace the (ideal) helium gas in a model with a more accurate representation that calculates density and sound speed using the first coefficient of the virial expansion for helium, we can write the following file, call it `helium.tpf`, and put it in the same directory as our model:

```
! external fluid; He with first virial coeff for (B=12cc/mole)
! Equation is:
! C0 + C1*pm/(T+C2*pm) + C3*T + C4*T^2 + C5*T^C6 + pm^2 *C7*T^C8 + pm*C9
! Density, rho (m^3):
  0.  4.814e-4 1.44e-6
! isobaric heat capacity, cp (J/kg/K):
  5192.
! Thermal conductivity, k0 (W/m/K):
 0. 0. 0. 0. 0. 0.0025672  0.716
! Square of sound speed, a^2 (m^2/s^2):
  0.  0.  0.  3461.92  0. 0. 0. 0.  0. .0100
! Viscosity, mu (kg/s/m):
 0. 0. 0. 0. 0. 0.412e-6  0.68014
```

The coefficients for density were determined using

$$\rho = \frac{p_m M}{R(T + Bp_m/R)},$$

where $R = 8.314$ J/mole-K, $M = .0040026$ kg/mole, and the first virial coefficient $B = 1.2 \times 10^{-5}\text{m}^3$/mole. We set $C_1 = M/R$ and $C_2 = B/R$. For squared sound speed, we need to satisfy

$$a^2 = \frac{\gamma RT}{M}\left(1 + 2\frac{Bp_m}{RT}\right),$$

so we set $C_3 = \gamma R/M$, and $C_9 = 2B\gamma/M$, where $\gamma = \frac{5}{3}$. See Sec. C.1 in Chapter VI to compare this with how helium properties are calculated in DeltaE's internal routine.

For equations that cannot be manipulated to fit the format of Eq. V.3, we suggest generating a table of data near the expected operating conditions and using curve-fitting tools to generate appropriate coefficients.

User-defined solids follow an identical format, except that only the first three lines are required to specify ρ_s, c_s, and K_s. The meaning of coefficients C_1 and C_2 are also redefined to provide an exponential capability, so the equation for solids is

$$\text{property} = C_0 + C_1 \exp(-TC_2) + C_3T + C_4T^2 + C_5T^{C_6} + C_7p_m^2T^{C_8} + p_mC_9. \qquad \text{(V.4)}$$

It is a good idea to check each new external fluid or solid by using the `(t)hermophysical` command available in the main menu (external fluids or solids show up first and are selected with negative integers). Users can also insert the `THERMophysical` segment using the fluid/solid to display the properties in the `.out` and `.dat` files, or to plot them (see below).

F. Parameter Linking (Special Modes)

DeltaE is versatile in the way it uses different model parameters as guesses to meet its targets: length or volume (to achieve resonance at fixed frequency), stack length and position (to meet an efficiency and amplitude), or stack diameter (to get adequate power), for example. When such geometric variables are released to the solver for manipulation (or when they are made to change in a plot loop), there are often certain geometric relationships to other parameters that we would like to see maintained. For example, if the area of a duct increases, we must increase the associated perimeter as well. Another common wish is to lengthen a segment while simultaneously shortening another segment to keep overall length constant. Also, in a stack made of a constant thickness material in a duct of fixed diameter, we cannot blindly vary the pore size and expect the porosity to remain the same—this could lead to a misleading optimization if we are faced with these constraints. If we go to the trouble to calculate a porosity for our initial segment, we want DeltaE to respect it for the values it chooses as we run the model. 'Special modes' were introduced to link parameters for just these purposes.

A special modes dialog appears automatically whenever a parameter linking capability is possible for a variable that is chosen as a guess vector member:

```
MAIN: (rpwPncTCgudvomfst e?)> u
Guess/Target Address=? ( 0a) 4d
Selection: STKCIR:r0

Add to the guess vector (y/n)? y
Special modes can be  enabled as this parameter is varied
(Only one mode per segment possible):
Mode       Description
     0   Normal mode (no inter-related parameters)
    -1   Adjust porosity while y0(r0) varies (const. Area, L0)
Mode=? ( 0) -1
```

By selecting `-1` for the special mode, we have asked DeltaE to remember the following constant before it begins iterating:

$$\texttt{const} = \texttt{r0}/\texttt{poros} - \texttt{r0}$$

where `r0, poros` are the pore size and porosity of our initial stack. We assume that the effective plate material thickness 2L0 is given by (2L0) $=$ $\texttt{r0}(1/\texttt{poros} - 1)$. During the iteration, as `r0` is changed, DeltaE assumes porosity changes as an ideal porosity would and calculates it from the following:

$$\texttt{poros} = \texttt{r0}/(\texttt{r0} + \texttt{const}),$$

and the effective plate thickness is maintained.

If we create a plot varying the area of our first `INSDUct` (parameter 2a, in most of the examples of the previous chapters), the dialog looks like

```
MAIN: (rpwPncTCgudvomfst e?)> p
define plot variables.  One or two inputs (a-j)
and up to 10 outputs (A-J) can be plotted)
Plot Parameter Address=? ( 0A)2a

use for outer or inner (2d) plot loop (o/i)? o
Outer (or 1-D) Plot Loop:
Independent variable is  DUCT:Area
Plot begins at DUCT:Area  =  1.2920E-02    m^2
New value (<CR> to keep)=?
Plot ends at DUCT:Area  =  1.2920E-02    m^2
New value (<CR> to keep)=? 2e-2
with a step of:    1.00
New value (<CR> to keep)=? 1e-3
Special modes can be  enabled as this parameter is varied
(Only one mode per segment possible):
Mode       Description
     0   Normal mode (no inter-related parameters)
    -2   Maintain consistent Perimeter as initial Area varies
Mode(n)=? ( 0)-2
```

By selecting `-2` for the special mode, we have asked DELTAE to remember the constant:

$$\texttt{const} = \texttt{perim}^2/\texttt{area}$$

and, later, to calculate the perimeter from

$$\texttt{perim} = \sqrt{\texttt{area} * \texttt{const}}.$$

This relationship keeps circular ducts circular and maintains the aspect ratio of rectangular ducts.

A very complicated example, even if somewhat confusing, can give some idea of the power of parameter linking. Interesting iterations can be done by using `sameas` parameters in combination with length parameter linking. For example, if segments 2 and 7 are `DUCT`s, and segments 4 and 5 are `STKSLab`s of equal length (but different material or porosity, perhaps), we can iterate using stack length, keep these lengths equal, *and* keep the overall length and stack center position constant by doing the following:

1. For the length (`c`) of segment 5, specify `sameas 4c`.
2. `(u)se` parameter `4c` as a guess (you will have to clear another guess, or add a suitable target, to keep your guess and target vectors balanced).
3. When prompted to select a special mode for segment 4, choose '2' to keep the sum of segment 2 and 4's lengths constant.
4. Using the `(s)pecial modes editing` option, select parameter `5c` and set its mode to '7'.

If `4c` were an independent plot loop variable instead of a guess vector member, the procedure above would be identical, except that item (2) would be a `(p)lot` selection option instead of a `(u)se` dialog. The following is a list of all parameter linking modes and the segment types for which they are available:

`n`	Keep Length + Length in segment (n) constant: All segments with length.
`0`	Normal mode (no inter-related parameters): All segments
`-1`	Adjust porosity while y0(r0) varies (const. Area, L0): Most stacks and heat exchangers.
`-2`	Maintain consistent Perimeter as initial Area varies: Ducts and cones.
`-3`	Maintain consistent Perimeter as final Area varies: Cones.
`-4`	Adjust porosity as L0 varies (const. Area, y0): `STKSLab`.

`-5` Maintain consistent surface area as volume varies: `COMPLiance`.

`-6` Maintain constant V & valid perim., area as length varies: `STKDUct`

`-7` Vary imaginary part to preserve magnitude (where possible): `IMPED`, `BRANCh`, `TBRANch`, and `HBRANch`.

`-8` Vary imaginary part to preserve phase angle: same as `-7`.

`-9` Maintain consistent Perimeters in cones as both initial and final Areas vary.

Special modes can be tracked by status fields in the center of the `.out` file, between the input and output colums. The master parameter—the one through which the mode is controlled—has the form `S:` *n*, where *n* is the mode number. A slave parameter, one which can not be modified independently when it is controlled by a special mode, has a status indicator of the form `Fnc(`*nn*`p)`, where *nn*`p` is the parameter address (*e.g.* `2a`).

G. Thermophysical Properties

The `(t)hermophysical` menu selection (see Chapter VI for further details) allows the user to have keyboard access to the library of fluid and solid properties for a given state (which defaults to the current temperature, pressure, acoustic frequency, and fluid or solid). This feature has proven so convenient that we often start DeltaE simply to look up the transport properties of gases. (For this purpose, it is often useful to have a dummy file present (e.g., `nil.in`) that contains only a `TITLE` line. If you respond to the input file prompt with this filename, DeltaE will quickly go to the menu line and allow you to access the options.)

A companion to the `(t)hermophysical` menu selection is `THERMophysical` segment type, which takes no input parameters except for the fluid and solid type (again, see Chapter VI for a summary). This segment can be inserted anywhere in a model where the user wants to know the fluid and solid properties at the local temperature and pressure, whatever they may be at the time. Both the `.out` and `.dat` files contain outputs for these properties where the segment is inserted. By using the plotting loops, tables of properties can be generated over ranges of temperature, pressure, or frequency by varying these values in a `BEGIN` segment, ending the model with a `THERMophysical` segment, and plotting as many of the outputs as are required.

H. State Variable Plots

State variable plots allow you to view the distribution of temperature, pressure, volume flow rate, and enthalpy along the entire length of a model. The format is somewhat similar to that of the `*.dat` file, but with more detail. Selecting `(G)enerate state variable plot` from the `(E)xtras` submenu will cause a `*.spl` file to be written. The output below was generated from the `5inch.in` example file (Sec. III.B) *before* guess and targets were added, and before iterations were performed:

```
 ->5INCH.spl
!Created@15:32:59  8-Jan-98 with DeltaE Vers. 4.0b5 for the IBM/PC-Compatible
-= Five-Inch Thermoacoustic Engine                                          =-
Seg# x(m) GasA(m^2) T(K)    Re[p](Pa)  Im[p](Pa) Re[U]   Im[U](m^3/s) Hdot(W)
  1 0.000 0.012920  500.0    80000.0        0.0  0.00000   0.00000      0.00
  1 0.000 0.012920  500.0    80000.0        0.0 -0.00003   0.00000     -1.20
  2 0.000 0.012920  500.0    80000.0        0.0 -0.00003   0.00000     -1.20
  2 0.056 0.012920  500.0    79992.8        0.1 -0.00006  -0.00790     -1.20
  2 0.112 0.012920  500.0    79935.6        0.6 -0.00011  -0.02371     -1.20
  2 0.167 0.012920  500.0    79821.2        1.6 -0.00016  -0.03949     -1.20
  2 0.223 0.012920  500.0    79649.7        3.0 -0.00021  -0.05525     -1.20
  2 0.279 0.012920  500.0    79421.3        4.7 -0.00026  -0.07096     -1.20
  2 0.279 0.012920  500.0    79285.7        5.8 -0.00029  -0.07880    -11.65
  3 0.279 0.005078  500.0    79285.7        5.8 -0.00029  -0.07880    -11.65
  3 0.339 0.005078  500.0    78318.3      403.1 -0.00184  -0.08697   2198.55
  4 0.339 0.010465  500.0    78318.3      403.1 -0.00184  -0.08697   2198.55
  4 0.395 0.010465  428.8    77731.9      847.5  0.00166  -0.09752   2198.55
  4 0.451 0.010465  375.9    76966.1     1289.3  0.00455  -0.10886   2198.55
  4 0.506 0.010465  334.5    76002.7     1733.3  0.00703  -0.12056   2198.55
  4 0.562 0.010465  300.9    74823.1     2182.0  0.00920  -0.13235   2198.55
  4 0.618 0.010465  272.8    73408.4     2636.2  0.01117  -0.14404   2198.55
  4 0.618 0.010465  272.8    73408.4     2636.2  0.01117  -0.14404   2198.55
  5 0.618 0.006158  272.8    73408.4     2636.2  0.01117  -0.14404   2198.55
  5 0.669 0.006158  272.8    71194.2     2996.0  0.01046  -0.15121    145.66
  6 0.669 0.012670  272.8    71194.2     2996.0  0.01046  -0.15121    145.66
  6 1.400 0.012670  272.8    62578.7     2479.1  0.01377  -0.23637    145.66
  6 2.130 0.012670  272.8    35545.6     1109.2  0.01814  -0.36310    145.66
  6 2.861 0.012670  272.8      688.1     -470.6  0.01887  -0.40990    145.66
  6 3.592 0.012670  272.8   -34320.3    -1946.2  0.01584  -0.36645    145.66
  6 4.323 0.012670  272.8   -61772.8    -3026.4  0.00967  -0.24234    145.66
  6 4.323 0.012670  272.8   -70671.8    -3347.8  0.00578  -0.15802     60.13
  7 4.323 0.012670  272.8   -70671.8    -3347.8  0.00578  -0.15802     60.13
  7 4.323 0.012670  272.8   -70671.8    -3347.8  0.00580  -0.15802     59.58
```

The following features of state variable plots are noted:

- When generating a state variable plot, DeltaE does not iterate; it simply takes one pass through the model using the current guess variable values.
- During the pass, DeltaE prints `Nint`/2 + 1 (`Nint` is the number of Runge-Kutta steps—see Sec. J for details) lines of data for each segment that it knows how to integrate (stacks, ducts, and cones).
- Two lines are printed for elements which do direct calculations (heat exchangers, end-caps, etc.): one before, and one after the segment is computed.
- Segments that do not have any physical effect, such as math segments, `BEGIN`, and `'END` segments, generate no output in the plot.
- The third column, `GasA`, is the current cross-sectional area times the porosity of the segment.

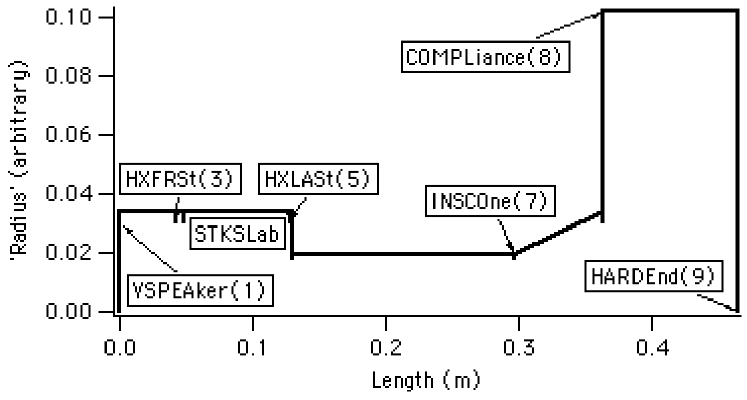


Figure V.3: Geometry of Hofler refrigerator example.

- Acoustic power is not an output, but, in a spreadsheet, it can be derived from the plotted variables using $\dot{E} = \Re[p_1\tilde{U_1}]/2$

- When a model contains a branch segment, a blank line will be left before and after the branch in the output. Also, the x distance counter begins at zero again in the branch.

I. Geometry

When sizes are changing dynamically, it is often desirable to know something about the physical size and layout of a device under design, no matter how abstract the available information may be. DeltaE has an option to write a 'geometry' file for this purpose. Selecting it causes a `*.geo` file to be written that contains x, y pairs suitable for plotting with your favorite graphics software. When this file is given to graphing software, the resulting plot is representative of a half cross-section of a cylindrical device similar to the model. The figure below is an annotated plot made from the geometry file for our final example of the Hofler refrigerator.

We have labeled all of the segments, except for the ducts, with their numbers and types. DeltaE generates little 'tick' marks to identify the breaks between segments. The lines down to zero on either end are generated by the `VSPEAker` and `HARDEnd` segments, respectively. The height at most points is proportional to the square root of the area. The `COMPLiance` is the exception; it looks nothing like Hofler's sphere. It is a symbolic cylinder that has length equal to radius (sort of—factors of π are ignored) proportional to the specified volume. (Some models, such as those with active branches, are not supported properly by geometry files yet.)

J. Tuning and Debugging

The `(T)olerances/debugging` menu selection gives the user access to a number of internal parameters that control the quantity of output and diagnostic information generated and the way that the solver approaches the iterations it will perform. An explanation of these parameters is given below:

`Nprint` If `Nprint` ≤ 0, the `.dat` file will contain only the final converged iteration of the model. Otherwise, DeltaE saves every `Nprint`th intermediate iteration. If `Nprint` ≥ 0, intermediate steps in every stack integration are included in the data file. For `Nprint` > 0, every segment is displayed to the screen (equivalent to typing the `.out` file). This can be useful in finding model errors that cause DeltaE to crash before the first converged data point is ever stored. If `Nprint` < 0, a concise iteration summary line is printed every `-Nprint+1` intermediate iterations. By setting `Nprint` to a larger negative integer, time-consuming screen output can be reduced, which will make calculations run several times faster on machines with good floating point performance. The summary line contains only the iteration number and the root-mean-square sum of the errors (targets $-$ results), and the line will overstrike itself. If `Nprint` $= 0$, the iteration count and the complete guess and (target$-$result) vectors are displayed on sequential lines. Default: -1.

`PlotDat` This variable controls output generated during plots, where multiple solutions are processed sequentially. If `PlotDat` ≥ 0, all error messages that occur when DeltaE has doubts about the convergence of a datapoint are announced (on a MacIntosh, 'OK' must be clicked in the alert box before calculations will continue. For other values of `PlotDat`, DeltaE will continue silently, but will still write the messages to the `.dat` file and mark the lines in the `.plt` file with a '$*$.' If `PlotDat` ≥ 1, all converged endpoints are written to the .dat file (it can become quite large!). For `PlotDat` $= 0$, only the most recent is kept. Default: 0.

`tolerance` Recommended range: 10^{-9}–10^{-2}. This value governs the point at which DeltaE considers its iterations finished. The default value is close to the limit that can be reached using single-precision arithmetic (all DeltaE calculations are double precision). This value does not relate directly to errors between any particular result and its target value; it concerns changes in the norm of the error vector. Default: 10^{-5}.

`Runge-Kutta steps` This is an even integer that determines the number of integration steps used to span each stack-type segment, turbulent duct, or cone. It does not affect other segment types. It also determines the resolution with which state variable plots (the `(G)enerate` option described in the preceding section) are printed: `Nint`/2 lines per segment. Larger values will cause a slower, more accurate computation. Small values will increase speed at the price of integration accuracy, but may cause convergence problems if the specified tolerance is too small. Output from every other step can be enabled with the `Nprint` parameter. Default: 10.

`Normalization mode` In a numerical problem where all of the input variables in the guess vector and all of the output variables used in the target vector are of wildly different magnitudes, a difficulty arises in choosing how much to change each variable and how much to weigh the errors between the target and the result values. Particularly, this affects `HARD`- and `SOFTEnd` segments. A 0.01 K error in a heat exchanger temperature is fairly benign to us, but in the complex end impedance, an error of 0.01 could leave us with hundreds of watts of power flow where there must be zero. In the standard mode (1), DeltaE uses the solver's internal method to normalize the solution vector, which usually does a reasonable job. For pathological cases, DeltaE has a special mode (2) that tries to normalize all input variables and output variable differences to unity. This can present its own problems, however, since we do not know how to normalize zero input variables (phases are a special case, automatically normalized by 360°). In normalizing outputs, problems can occur if the model is very far from being converged, giving large initial error values; if it is very close to being converged, the errors could be near zero, presenting the other problem. Use mode 1 whenever possible, and mode 2 when you must. It may sometimes help to specify a zero input (target or guess) variable as some tolerably small nonzero number when using this mode. Default: 1.

`Step bound factor` recommended range: 0.01–100. This value regulates the size of initial excursions DeltaE makes from initial guesses to find the directions in which it must iterate. Some difficult cases can benefit by reducing this value. Default: 100.

`FCNerr` There is a limit to the accuracy with which a computer can calculate the 'function' that represents one complete pass through a model. The assumed value of this error affects the increments between iterations that the solver will choose; if the increments are too small, the effect on the result will be unpredictable. Larger values of `FCNerr` can speed iterations, with a less accurate endpoint. Too small a value can cause the solver to lose its way completely. This quantity is system-dependent, and it may be necessary to increase it slightly for very complex models. Recommended range: $> 5 \times 10^{-15}$. Default: 10^{-10}.

`Minimum Temperature` There is a temperature floor, 10 K by default, to prevent DeltaE 's solver from exploring unphysical temperatures such as negative temperatures. Brave users with special needs at lower temperatures (and generally with their own, external thermophysical properties files!) can set this floor to a lower value. (Some of DeltaE's internal fluids use a higher temperature floor. He-Ar mix, for example, does not calculate properties below 70 K, in order to prevent unreasonable values from being generated).

`Display exergy` The last two columns in `.dat` files are normally enthalpy and work flux. Beginning with version 5.0, DeltaE will optionally append a column `Xdot` to keep track of the *exergy* flux. When this mode is set, an environment temperature is also requested to which the exergy change is referenced. Also in this mode, a line stating the change in exergy flux at each segment is added. The latter is particularly useful for tracking effects such as acoustic dissipation in the `INSULATE` mode, where energy does not leave the system except through segments such as heat exchangers.

`Plot field delimiter` Normally, DELTAE generates plots and state variable plot files with the numbers in fixed width columns. Some spreadsheets and plotting packages do not process these files as well as they do delimited text. If `plot_f_sep` is set to `1`, a comma will be placed between columns in each of these plot types.

J.1. Initialization files.

If any of the above parameters are modified from their default values, you will generally want to keep the new values for every new run on the current model, and reuse them every time you execute the program. Therefore, whenever the `(T)olerances/debugging` option is used to change the default settings, all of the tunable parameters are written to a special file when the model is saved. This file has the same base filename as the model, with the extension `ini`. Whenever a new model is loaded, DELTAE checks for a `.ini` file in the same directory with a matching base filename and loads these settings if it is found. This file is written in `NAMELIST` format which makes it easy to examine and modify using any text editor.

Frequently, a collection of similar models will reside in a single subdirectory, and these files will share identical custom settings. For these situations, any `.ini` file can be copied (or renamed) to `default.ini` and DELTAE will use the settings within it for any model run from the same directory. If a model has its own individual `.ini`, its settings will take precedence.

VI. Reference

A. General

DeltaE solves the one-dimensional wave equation, with temperature evolution, in the usual low-amplitude, "acoustic" approximation.

In each pass, DeltaE integrates from `BEGINning` to `HARDEnd` or `SOFTEnd`, with respect to 5 real variables: real $T_m(x)$, complex $p_1(x)$, and complex $U_1(x)$. It uses the differential (or simpler) equations appropriate for each segment, with the evolution of these variables in each segment controlled by local parameters, such as geometry and energy flow, and global parameters, such as frequency and mean pressure. Continuity of T_m, p_1, and U_1 are used at the junctions between segments.

In general, a single pass of DeltaE's integration does not result in desired values of all variables. A shooting method is used to adjust chosen initial variables, called "guesses," in order to hit desired results, called "targets." Initial guesses are provided by the user, or (more commonly) by a previous run of DeltaE.

The table below serves as a guide to choice of guess and target variables.

DeltaE has two modes for handling total energy flow $\dot{H}_2$ in segments other than stacks and heat exchangers, i.e., in ducts, cones, compliances, impedances, and the like. Segments `INSULate` and `CONDUct` control this mode in subsequent segments; conduction mode is the default if neither `INSULate` nor `CONDUct` is present. In insulated mode, $\dot{H}_{\text{out}} = \dot{H}_{\text{in}}$ in ducts, cones, etc. In conduction mode, $\dot{H}_{\text{out}} = \dot{E}_{\text{out}}$ in these segments, independent of the value of $\dot{H}_{\text{in}}$. Use insulated mode when you want to place ducts, compliances, or impedances between stacks and heat exchangers, and when you want to model insulated acoustic networks attached to thermoacoustic devices. Use conduction mode when you don't care about $\dot{H}$. (The mode also has an effect on the calculation of $\dot{Q}$ in `HX`, `TX`, etc. when they appear after a stack; see `HX` subsection below for details.)

Table VI.1: Choosing guesses and targets.

	Variables *we* think of as fixed	Variables *we* think of as results
Inputs for *each pass* of DELTAE. Includes T-begin, p-begin, U-begin, p-mean, freq, all dimensions, 'ducer coefficients, volts @`VDUCEr`, heat at most `HX`, gas concentration.	simply fixed in input file	guess
Results from *each pass* of DELTAE. Includes all T, p, U except in `BEGIN`; heat @ `HXLASt` in `CONDUCT` mode; current in `VDUCErs`; combinations of above such as `RPNTArg` or z at ends.	target	simply results in `.out` and `.dat` files

B. Segments

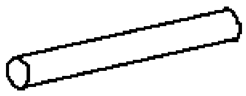
All of DELTAE's segment types are listed by functional grouping in this section. An alphabetical listing and cross-reference is presented at the end of the section.

B.1. Ducts, cones

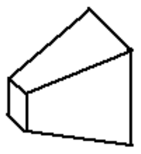
Segment types: `DUCT, ISODUct, INSDUct, CONE, ISOCOne, INSCOne`

Sample input-file segments:

```
DUCT    comments typed here are retained in output
3.14e-4  m2 area
0.0628   m perim
0.1      m length
helium     gas
copper     solid
```



```
ISOCONE    this one is square
1.0   m2   Initial Area
4.0   m    In Perim
2.0   m    Length
0.25  m2   Final area
2.0   m    Final perim
air        gas
ideal      solid
```



Use:

Use for ducts and cones of any cross-sectional shape (*e.g.*, square, circular) by giving suitable area and perimeter. `DUCT` and `CONE` are preferred. The `INS` and `ISO` varieties are leftovers from earlier versions of DeltaE, before segments `INSULATE` and `CONDUCT` were created to control thermal contact at the side walls. Mean temperature is independent of x.

Computation algorithm:

T_m is not affected by ducts.

In laminar ducts of length L, p_1 and U_1 are calculated by

$$\begin{aligned} p_{\text{out}} &= p_{\text{in}} \cos kL - \frac{i\omega\rho_m}{(1-f_\nu)kA} U_{\text{in}} \sin kL, \\ U_{\text{out}} &= U_{\text{in}} \cos kL - \frac{i(1-f_\nu)kA}{\omega\rho_m} p_{\text{in}} \sin kL, \end{aligned} \tag{VI.1}$$

with complex wavevector k given by

$$k = \frac{\omega}{a}\sqrt{\frac{1+(\gamma-1)f_\kappa/(1+\epsilon_s)}{1-f_\nu}}. \tag{VI.2}$$

In cones, p_1 and U_1 evolve according to

$$\begin{aligned} \frac{dp_1}{dx} &= -\frac{i\omega\rho_m}{(1-f_\nu)A} U_1, \\ \frac{dU_1}{dx} &= -\frac{iA\omega}{\rho_m a^2}\left[1+\frac{\gamma-1}{1+\epsilon_s}f_\kappa\right] p_1, \end{aligned} \tag{VI.3}$$

which are equivalent to the lossy Webster horn equation

$$\left[1+\frac{\gamma-1}{1+\epsilon_s}f_\kappa\right] p_1 + \frac{a^2}{\omega^2}\frac{1}{A}\frac{d}{dx}\left[(1-f_\nu)\,A\frac{dp_1}{dx}\right] = 0. \tag{VI.4}$$

The perimeter varies linearly from its initial value to its final value; area varies quadratically. Hence, circular cones have circular cross-sections everywhere, with diameter varying linearly with axial position.

In very narrow ducts and cones, for $R/\delta < 25$, f_κ and f_ν are calculated using complex Bessel functions

$$f_\kappa = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\nu]}{(i-1)(r_0/\delta_\nu)J_0[(i-1)r_0/\delta_\nu]}. \tag{VI.5}$$

Where $R/\delta > 30$, the boundary-layer approximation is used:

$$f_\kappa = (1-i)\Pi\delta_\kappa/2A, \quad f_\nu = (1-i)\Pi\delta_\nu/2A. \tag{VI.6}$$

For intermediate values, linear interpolation is used to make a smooth match between the two regimes. While the narrow duct solution assumes a circular cross-section, the shape of the duct is irrelevant in the boundary-layer approximation. A square duct with dimensions much larger than the penetration depth can be modeled simply by choosing perimeter $= 4\sqrt{\text{area}}$, for example.

In all cases, boundary-layer approximation is used for the effect of the solid:

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2}. \tag{VI.7}$$

The exiting total energy flow $\dot{H}_{\text{out}}$ is computed as $\dot{H}_{\text{out}} = \dot{E}_{\text{out}}$ for `DUCT` and `CONE` in conduction mode, and for `ISODuct` and `ISOCone` in either mode. This essentially assumes that the duct wall is thermally anchored, so heat generated by acoustic power dissipation is carried away externally. Thermoacoustic heat transport along the perimeter, which in fact contributes a small difference between $\dot{H}$ and $\dot{E}$ in ducts, is neglected.

The exiting total energy flow $\dot{H}_{\text{out}}$ is computed as $\dot{H}_{\text{out}} = \dot{H}_{\text{in}}$ for `DUCT` and `CONE` in insulated mode, and for `INSDuct` and `INSCone` in either mode. This essentially assumes that the side walls are thermally insulated, so that the heat generated by acoustic power dissipation is deposited in an adjacent heat exchanger. If several `INSDUct`s and/or `INSCOne`s are strung together, the power dissipated in all of them should show up in the nearest heat exchanger.

(See also `STKDUct`, which allows a temperature gradient along a duct. It is described under Stacks below.)

Turbulence extensions:

DELTAE's turbulence algorithm assumes that turbulent oscillatory flow is described by the Moody friction factor at each instant of time during the oscillatory flow. [See any engineering fluid mechanics textbook to review the Moody friction factor as a function of Reynolds number and relative roughness of the pipe wall.] This assumption must be excellent in the low frequency limit, in which $R/\delta_\nu \longrightarrow 0$. This limit is approached in many inertances for pulse tube refrigerators. We do not know how good the assumption is for large R/δ_ν, which is of interest in the resonators of standing-wave thermoacoustic systems. For experimental validation of the assumption for intermediate R/δ_ν, see Ref. [18].

DeltaE's turbulence algorithm can be enabled in `DUCT`s, `CONE`s, `ISODUCT`s, `INSDUCT`s, `ISOCONE`s, and `INSCONE`s. T_m is unchanged, and p_1 and U_1 are numerically integrated according to Eqs. VI.3 above, with f_ν and f_κ modified as described below to account for the turbulence. To enable the turbulence algorithm, include (or `(m)odify`, from within the program) parameter **d** in the '`DUCT` segment in the input/output file (use parameter **f** for '`CONE`s). Parameter **d** is the relative roughness ε, whose definition can be found in fluid-mechanics textbooks: roughness height divided by pipe diameter. A typical value might be 10^{-3}. Setting this parameter equal to minus one makes that line of the input/output file disappear, returning the `'CONE` calculation to laminar.

A sample of a modified duct segment, with turbulence enabled, is given below.

```
!--------------------------------- 6 --------------------------------
 ISODUCT    Cold Duct
  4.0000E-03 a Area      m^2  S:-2       1.1129E+05 A |p|       Pa
  0.2220     b Perim      m   Fnc( 2a)   -176.6     B Ph(p)     deg
   3.650     c Length     m              2.7085E-05 C |U|     m^3/s
  1.0000E-04 d Srough                    -176.7     D Ph(U)     deg
                                          1.507     E Hdot      W
 helium     Gas type                      1.507     F Edot      W
```

The portion of the `.dat` file corresponding to the above duct segment is as follows:

```
!------------------------------------  6 -----------------------------------
 ISODUCT   Cold Duct
 Re=0.29E+06, r/dn= 210.7, m=  1.1574, m-prime=0.9987 at start;
 Re=0.39E+06, r/dn= 210.7, m=  1.4568, m-prime=0.9970 peak @x= 1.2775
  End of this segment is laminar.
 Heat extracted:    137.    Watts
   306.6   -111099.   -6592.1     -0.00003  0.00000           1.51      1.51
```

There are three new lines in the listing. The parameters given in the first two lines are Reynolds number (based on diameter), the ratio of radius to viscous penetration depth δ_ν, the dissipation multiplier m, the inertial multiplier m' (see below), and the location along the duct. The third line says that velocities return to a laminar regime before the end of the duct. If the peak Reynolds number occurs at either end of the duct, or if the entire duct is laminar, the middle line detailing the peak location will be omitted.

The volume flow rate and hence Reynolds number N_R vary sinusoidally in time; hence, the instantaneous Moody friction factor f_M has a complicated time dependence. We simplify this time dependence by essentially using a Taylor-series expansion around the peak Reynolds number:

$$f_M(t) \simeq f_M + \frac{df_M}{dN_R}\frac{N_R}{|U_1|}\left(\mathrm{Re}\left[U_1 e^{i\omega t}\right] - |U_1|\right), \tag{VI.8}$$

where f_M and the derivative on the right-hand side are evaluated at the peak Reynolds number. It is then straightforward to integrate the instantaneous power dissipation over a full cycle, obtaining for the time-averaged power dissipation per unit length

$$\frac{d\overline{\dot{E}}}{dx} = \frac{\rho\,|U_1|^3}{3\pi^3 R^5}\left[f_M - (1 - 9\pi/32)\,N_R\frac{df_M}{dN_R}\right], \tag{VI.9}$$

where the quantities in the square bracket are evaluated at the peak Reynolds number.

When this is compared to the equivalent result for laminar flow

$$\frac{d\overline{\dot{E}}}{dx} = \frac{\rho\,|U_1|^2\,\omega}{2\pi R^2}\mathrm{Re}\left[\frac{i}{1-f_\nu}\right], \tag{VI.10}$$

it is apparent that turbulence multiplies the dissipation by a factor m given by the ratio of the two expressions above:

$$m = \frac{\delta_\nu^2 N_R}{6\pi R^2}\frac{\left[f_M - (1 - 9\pi/32)\,N_R\,df_M/dN_R\right]}{\mathrm{Re}\left[i/\left(1-f_\nu\right)\right]}. \tag{VI.11}$$

DeltaE evaluates f_M and df_M/dN_R as a function of Reynolds number and ε using the iterative expression

$$\frac{1}{\sqrt{f_M}} = 1.74 - 2\,\log_{10}\left(2\varepsilon + \frac{18.7}{N_R\sqrt{f_M}}\right), \tag{VI.12}$$

which is a remarkably good approximation to the Moody friction factor [R. M. Olson, Essentials of Engineering Fluid Mechanics]. To account for turbulence, DeltaE increases the resistive component of the pressure gradient, and hence the viscous power dissipation, by m. It decreases the inertial pressure gradient by

$$m' = \left(\frac{1-\delta_\nu/R}{1-\delta_\nu/mR}\right)^2 \tag{VI.13}$$

to correct approximately for the steeper velocity gradient at the wall, which increases the effective area open to gas contributing to inertial effects. It also multiplies the thermal penetration depth by m, in an attempt to account very approximately for changes in thermal relaxation losses due to increased heat transfer. Both m and m' are displayed in the `*.dat` file.

At low enough velocities, $m \longrightarrow 1$ and DeltaE reverts to a laminar calculation. The $m = 1$ boundary between laminar and turbulent zones in DeltaE occurs roughly at

$$N_R \simeq 2000 \text{ for } R/\delta_\nu < 2, \tag{VI.14}$$

$$\frac{N_R}{R/\delta_\nu} \simeq 1000 \text{ for } R/\delta_\nu > 2. \tag{VI.15}$$

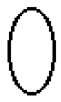
DeltaE versions prior to 3.3 had a simpler turbulence algorithm, which was adequate for standing-wave resonators but not for the Reynolds numbers and R/δ_ν's found in inertances for pulse-tube refrigerators. That algorithm was enabled by setting perimeters negative instead of positive in 'CONES. If you still have old DeltaE output files with negative perimeters, the current version of DeltaE should be able to read and interpret them; it will save them in the new format.

B.2. Lumped elements: compliance, endcap, impedance

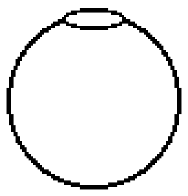
Segment types: `COMPLiance, ENDCAp, IMPEDance`

Sample input-file segments:

```
ENDCAP    a surface with thermal dissipation
1.134e-3 m2  Area
SAMEAS 0     Gas
ideal        solid
```



```
COMPLIANCE  this one is a sphere
0.1257  m2  Area
4.19e-3 m3  Volume
0.859hexe   Gas
nickel      solid
```



```
IMPEDANCE    just a lumped series impedance
1.0 Pa-s/m3 Re(Z)
-0.2 Pa-s/m3 Im(Z)
helium
```

```
! Blank lines at ''solid'' location are interpreted as ''ideal'' solid
```

Use:

An endcap is a surface area with thermal dissipation. It always absorbs acoustic power. A compliance is exactly that: a lumped acoustic volume element with surface thermal dissipation. An impedance is a lumped series complex impedance.

Computation algorithms:

An endcap does not affect temperature or pressure amplitude; volume flow changes according to

$$U_{\text{out}} = U_{\text{in}} - \frac{\omega p_1}{\rho a^2}\frac{\gamma - 1}{1+\epsilon_s}A\frac{\delta_\kappa}{2}. \qquad \text{(VI.16)}$$

Pressure amplitude and temperature are unchanged by a compliance; volume flow changes according to

$$U_{\mathrm{out}} = U_{\mathrm{in}} - i\frac{\omega p_1}{\rho a^2}\left[V - i\frac{\gamma - 1}{1+\epsilon_s}A\frac{\delta_\kappa}{2}\right]. \tag{VI.17}$$

In both cases,

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s \rho_s c_s}\right)^{1/2}. \tag{VI.18}$$

At an impedance, volume flow rate and temperature are unchanged; pressure changes according to $p_{\mathrm{out}} = p_{\mathrm{in}} - ZU_1$.

The exiting total energy flow $\dot{H}_{\mathrm{out}}$ is computed as $\dot{H}_{\mathrm{out}} = \dot{E}_{\mathrm{out}}$ for conduction mode. This essentially assumes that the component is thermally anchored, so heat generated by acoustic power dissipation is carried away externally.

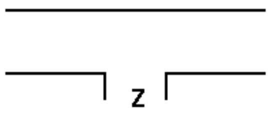
The exiting total energy flow $\dot{H}_{\mathrm{out}}$ is computed as $\dot{H}_{\mathrm{out}} = \dot{H}_{\mathrm{in}}$ in insulated mode. This essentially assumes that the component is thermally insulated, so that the heat generated by acoustic power dissipation must somehow be deposited elsewhere, either upstream or downstream, such as in an adjacent heat exchanger (or flow out through a `BEGIN` or `'END` segment).

B.3. Transducers, branches

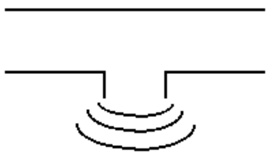
Segment types: `BRANCh, OPNBRanch, PISTBranch, VDUCEr, IDUCEr, VSPEAker, ISPEAker, VEDUCer, IEDUCer, VESPEaker, IESPEaker`

Sample input-file segments:

```
BRANCH
1  Pa-s/m3  Re(Z)
1.  Pa-s/m3  Im(Z)
0.500hear
ideal
```



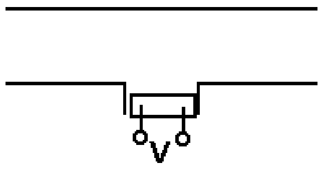
```
OPNBRANCH
.05  Pa-s/m  Re(Z)/k^2
.2   Pa-s/m2 Im(Z)/k
air
```



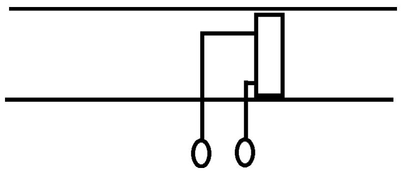
```
PISTBRAN     Baffled Piston
.05  Radius   m
air
```

```
 VDUCER
   1.000E-09 a Re(Ze)   ohms
    .000     b Im(Ze)   ohms
```

```
  1.000E+04 c Re(T1) V-s/m^3
   .000     d Im(T1) V-s/m^3
 -1.000E+04 e Re(T2)   Pa/A
   .000     f Im(T2)   Pa/A
  1.000E-09 g Re(Zm) Pa-s/m^3
  1.000E-09 h Im(Zm) Pa-s/m^3
  10.0      i AplVol    V
SAMEAS  0  Gas type
ideal      Solid type
```



```
VEDUCER     Enclosed driver
  1.000E-09 a Re(Ze)   ohms
   .000     b Im(Ze)   ohms
  1.000E+04 c Re(T1) V-s/m^3
   .000     d Im(T1) V-s/m^3
 -1.000E+04 e Re(T2)   Pa/A
   .000     f Im(T2)   Pa/A
  1.000E-09 g Re(Zm) Pa-s/m^3
  1.000E-09 h Im(Zm) Pa-s/m^3
  10.0      i  Vin      V
  45.0      j Ph(Vin)  deg
SAMEAS  0  Gas type
ideal      Solid type
```

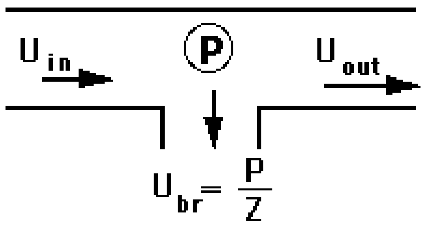


`IDUCEr` or `IEDUCer`: same as `VDUCEr` or `VEDUCer`, except that current appears in line `i` (and phase of it on line `j` for enclosed units) instead of voltage.

```
VSPEAKER
  6.000E-04 a Area      m^2
   6.00     b   R      ohms
   .000     c L         H
   8.00     d B x L    T-m
  5.000E-03 e   M      kg
   .000     f   K      N/m
   .000     g   Rm    N-s/m
  -22.5     h AplVol    V
SAMEAS  0  Gas type
ideal      Solid type
```

```
VESPEAKER
  6.000E-04 a Area      m^2
   6.00     b   R      ohms
   .000     c L         H
   8.00     d B x L    T-m
  5.000E-03 e   M      kg
   .000     f   K      N/m
   .000     g   Rm    N-s/m
  62.0      h  Vin      V
 -37.2      i Ph(Vin)  deg
SAMEAS  0  Gas type
ideal      Solid type
```

`ISPEAker` or `IESPEaker`: same as `VSPEAker` or `VESPEaker`, except that current appears in line `h` (and phase of it on line `i` for enclosed units) instead of voltage.



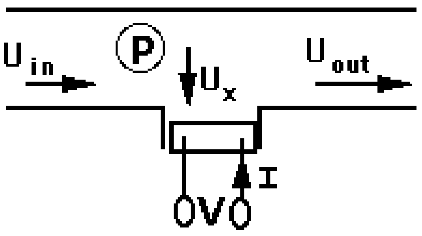


Figure VI.1: BRANCH (left) and branched 'DUCER or 'SPEAKER (right).

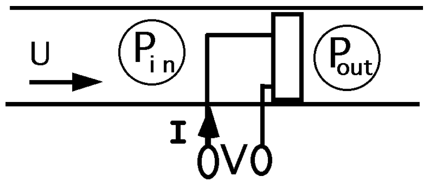


Figure VI.2: Enclosed 'EDUCer or 'ESPEaker.

Use:

BRANCh, OPNBRanch, and PISTBranch are side branches with fixed impedances. With BRANCh, the user specifies the real and imaginary parts of the impedance, assumed independent of frequency. OPNBRanch and PISTBranch incorporate the frequency dependence of radiation impedance. Thus radiation impedance at the end of an open tube radiating to 4π solid angle can be modeled as an OPNBRanch followed immediately by a HARDEnd. PISTBran approximates the radiation impedance of a baffled piston of the given radius in radiating into the specified fluid.

The 'DUCErs and 'SPEAkers are electroacoustic transducers. 'DUCErs have frequency-independent parameters; 'SPEAkers let the user specify mass, spring constant, force constant, resistance, and inductance, so that frequency-dependent (even resonant) transducers can be modeled. With IDUCEr and ISPEAker, the user specifies the (real) current, and each pass of DeltaE calculates the (complex) voltage; with VDUCEr and VSPEAker, the user specifies voltage, and DeltaE computes current. IDUCEr and ISPEAker cannot be used with zero mechanical impedance because this would lead to a division of zero by zero (see below). Hence, use VDUCEr or VSPEAker for resonant or massless-and-springless transducers.

'SPEAker-type segments incorporate dissipation losses over their area as if they included an ENDCAp, but 'DUCEr-type segments, which have no area parameter, do not. Enclosed 'ESPEaker-type segments include dissipation losses for both sides of the driver.

Branched transducer elements, which require only magnitude of voltage or current applied as an input, effectively anchor the phase to zero for that parameter. The phase of pressure and/or velocity, as given in the BEGIN statement, must usually be allowed to vary (i.e. guessed) in accordance with this reference. This effectively limits a model to only one

V or ISPEAker (or 'DUCEr), unless they are wired exactly in phase (or 180° out). Enclosed transducers, however, have a phase input which allows them to be used in oddly phased pairs, or where the phase reference is determined by a BEGIN statement, for example.

Computation algorithms:

None of these segments affects T_m.

A branch is a side branch with complex impedance Z. Pressure is unchanged, but volume flow rate changes according to $U_{\rm out} = U_{\rm in} - p_1/Z$. For an open branch, the numbers in the input file are multiplied by $(\omega/a)^2$ and ω/a respectively to obtain the impedance. For a baffled piston of radius r where the wavenumber is $k = \omega/a$ locally, the PISTBran radiation impedance is given by

$$Z_{\rm rad} = \frac{\rho a}{A}\left((1 - \frac{2J_1(2kr)}{2kr} + i \begin{cases} \left(\frac{4/\pi}{2kr} + \frac{\sqrt{8/\pi}\sin(2kr - 3\pi/4)}{(2kr)^{3/2}}\right) & \text{If } 2kr > 2.68, \\ \frac{(4/\pi)2kr}{3}\left(1 - \frac{(2kr)^2}{15}\right) & \text{otherwise.} \end{cases} \right) \tag{VI.19}$$

Output Edot_B gives the acoustic power flowing into the BRANCH.

A branched transducer **IDUCEr, VDUCEr, ISPEAKer, VSPEAKer** is an object attached as shown in the figure like a branch impedance, but obeying the complex equations $V_1 = Z_e I_1 + \tau U_x$, $p_1 = \tau' I_1 + Z_m U_x$. Pressure is unchanged, but volume flow rate changes according to $U_{\rm out} = U_{\rm in} - U_x$. The volume flow rate of the transducer, U_x, is displayed in the output column.

There are three cases of interest for a branched transducer:

1. If an electrical load impedance $Z_{\rm ext}$ is hung on the transducer, it should be covered using a BRANCH segment, with $Z_{\rm branch} = p_1/U_1 = Z_m - \tau\tau'/(Z_e + Z_{\rm ext})$.
2. If current I_1 is given (here, we take its phase to be real), then $U_x = (p_1 - \tau' I_1)/Z_m$ and $V_1 = Z_e I_1 + \tau U_x$.
3. If voltage V_1 is given (and we take its phase to be real), then $I_1 = (Z_m V_1 - \tau p_1)/(Z_e Z_m - \tau\tau')$ and $U_x = (V_1 - Z_e I_1)/\tau$.

An enclosed transducer **IEDUCer, VEDUCer, IESPEAker, VESPEAker** is an object attached in series with other segments, as shown in Fig. VI.2. Volume flow rate is nearly unchanged (except for surface thermal losses—see below), but pressure is changed by the force exerted by the transducer, obeying the complex equations $V_1 = Z_e I_1 - \tau U_1$, $p_{\rm out} - p_{\rm in} = \tau' I_1 - Z_m U_1$.

There are three cases of interest for an enclosed (series) transducer:

1. If an electrical load impedance Z_{ext} is hung on the transducer, it should be covered using an `IMPEDANCE` segment, with $Z_{\text{imp}} = Z_m - \tau\tau'/(Z_e + Z_{\text{ext}})$.
2. If current I_1 is given, then $p_{\text{out}} = p_{\text{in}} + \tau' I_1 - Z_m U_1$ and $V_1 = Z_e I_1 - \tau U_1$.
3. If voltage V_1 is given, then $I_1 = (V_1 + \tau U_1)/Z_e$ and $p_{\text{out}} = p_{\text{in}} + \tau' I_1 - Z_m U_1$.

In the case of speakers, $Z_e = R + j\omega L$; $\tau = -\tau' = Bl/A$; $Z_m = R_m/A^2 + j(\omega m - k/\omega)/A^2$. Thermal surface losses are computed for area A using the same formula as for an `ENDCAp`. Branch speakers are assumed to have area A exposed to the oscillating pressure. Enclosed speakers have area A exposed to p_{in} and area A exposed to p_{out}, because typically both sides of the speaker experience oscillatory pressure. As described above for `ENDCAps`, thermal surface losses manifest themselves as a small change in volume flow rate.

The `Px` outputs in the enclosed segments give the complex pressure difference across the transducer.

Note that `IDUCEr` and `ISPEAker` will crash if Z_m is zero, so it is best to use `VDUCEr` or `VSPEAker` for mechanically ideal or resonant transducers.

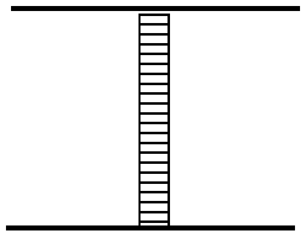
B.4. Heat exchangers

Segment types: `HX, TX, SX, PX; HXFRSt, HXMIDl, HXLASt; TXFRSt, TXMIDl, TXLASt; SXFRSt, SXMIDl, SXLASt, PXFRSt, PXMIDl, PXLASt`

Sample input-file segments:

```
HX        parallel-plate heat exchanger
sameas 1  Area
0.600     GasA/A
6.35e-3 m Length
1.9e-4  m y0 = half of plate spacing
-20.0   W HeatIn
300.    K Est-T
sameas 0  Gas
copper    solid
```

```
TXFRST     tube-in-shell heat exchanger
    0.2     a Area     m^2
   .188     b GasA/A
   .400     c Length    m
  6.350E-03 d radius    m   (radius of each tube)
  1.818E+05 e HeatIn    W
  1.000E+03 f Est-T     K
sameas  0  Gas type
nickel     Solid type
```



```
SX     Hot heat exchanger
  1.029E-03 a Area      m^2  total cross sectional area
  0.690     b VolPor         volumetric porosity
  2.000E-02 c Length     m
  6.450E-05 d   r_H      m   hydraulic radius
  -284.     e HeatIn     W
   300.     f Est-T      K      (t)
helium     Gas type
copper     Solid type
```

```
PX
  1.0E-4    a Area      m^2  total cross-sectional area
  0.70      b VolPor         volumetric porosity
  0.04      c Length     m
  4.000E-05 d   r_H      m   hydraulic radius = gas volume / gas-solid contact area
   900.     e HeatIn     W
  0.07      f f_con
  0.22      g f_exp
  0.035     h h_con
  0.22      i h_exp
   300.0    j Est-T      K
helium     Gas type
copper     Solid type
```

`*XMIDl` and `*XLASt` use same format.

Use:

Heat exchangers are used to inject or remove heat. They necessarily have surface area, so they experience both viscous and thermal dissipation of acoustic power. In `HX...`s the thermoacoustic working fluid is between parallel plates; in `TX...`s it is inside cylindrical tubes; and in `SX...`s the geometry is randomly-stacked screens. The `PX....` segments allow the user to control heat-exchanger parameters for which friction factor and heat transfer coefficient are power laws in Reynold's number. The derivation assumes fairly good thermal contact between fluid and solid.

`HX`, `TX`, `SX`, and `PX` are preferred. The `...FRST`, `...MIDL`, and `LAST` varieties are leftovers from earlier versions of DeltaE, when DeltaE was unable to ascertain the environment of the heat exchanger.

Heat exchangers have a temperature difference between metal temperature and fluid mean temperature that is proportional to the heat flow. The proportionality constant is not well verified experimentally; we believe it to within a factor of 2.

`SX...`s are valid only for hydraulic radius smaller than thermal and viscous penetration depths. There is no warning if this bound is exceeded.

Computation algorithms:

None of these segments affects temperature; i.e., T_m, the mean gas temperature, is unchanged. However, these segments estimate the temperature difference between the gas and the adjacent metal.

In `HX...` and `TX...` heat exchangers, p_{out} and U_{out} are calculated using

$$\begin{aligned} p_{\text{out}} &= p_{\text{in}} \cos kL - \frac{i\omega\rho_m}{(1-f_\nu)kA_{\text{fluid}}} U_{\text{in}} \sin kL, \\ U_{\text{out}} &= U_{\text{in}} \cos kL - \frac{i(1-f_\nu)kA_{\text{fluid}}}{\omega\rho_m} p_{\text{in}} \sin kL, \end{aligned} \tag{VI.20}$$

with complex wavevector k, given by

$$k = \frac{\omega}{a}\sqrt{\frac{1+(\gamma-1)f_\kappa/(1+\epsilon_s)}{1-f_\nu}}. \tag{VI.21}$$

`HX...`s use parallel plate geometry in computing f_κ, f_ν, and ϵ_s:

$$\begin{aligned} f_\kappa &= \frac{\tanh[(1+i)y_0/\delta_\kappa]}{(1+i)y_0/\delta_\kappa}, \quad f_\nu = \frac{\tanh[(1+i)y_0/\delta_\nu]}{(1+i)y_0/\delta_\nu}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{\tanh[(1+i)y_0/\delta_\kappa]}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.22}$$

Similarly, `TX...`s use cylindrical geometry in computing f_κ, f_ν, and ϵ_s: For $R/\delta < 25$, f_κ and f_ν are calculated using complex Bessel functions

$$f_\kappa = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}. \tag{VI.23}$$

Where $R/\delta > 30$, the boundary-layer approximation is used:

$$f_\kappa = (1-i)\Pi\delta_\kappa/2A, \quad f_\nu = (1-i)\Pi\delta_\nu/2A. \tag{VI.24}$$

For intermediate values, linear interpolation is used to make a smooth match between the two regimes. In both cases, ϵ_s is calculated using

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)r_0/2\delta_\kappa}{\tanh[(1+i)\ell/\delta_s]}. \tag{VI.25}$$

In `TX...`, the radius is that of one circular pore, so that for a heat exchanger comprised of N circular pores, the total cross-sectional area available to the working fluid is $N\pi r_0^2 =$ `(Area)(GasA/A)`.

In `SX...`,

$$\frac{dp_1}{dx} = -i\omega\rho_{\text{m}}\left[1+\frac{(1-\phi)^2}{2(2\phi-1)}\right]\langle u_1\rangle - \frac{\mu}{r_{\text{h}}^2}\left(\frac{c_1(\phi)}{8}+\frac{c_2(\phi)R_1}{3\pi}\right)\langle u_1\rangle, \tag{VI.26}$$

$$\frac{d\left\langle u_1\right\rangle}{dx}=-\frac{i\omega\gamma}{\rho_{\mathrm{m}}a^2}p_1+\frac{i\omega T_{\mathrm{m}}\beta^2}{\rho_{\mathrm{m}}c_p}\frac{\epsilon_{\mathrm{s}}+(g_c+e^{2i\theta_p}g_v)\epsilon_h}{1+\epsilon_{\mathrm{s}}+(g_c+e^{2i\theta_T}g_v)\epsilon_h}p_1, \tag{VI.27}$$

using

$$c_1(\phi)=1268-3545\phi+2544\phi^2, \quad c_2(\phi)=-2.82+10.7\phi-8.6\phi^2, \tag{VI.28}$$

$$b(\phi)=3.81-11.29\phi+9.47\phi^2, \tag{VI.29}$$

$$R_1=4\left|\left\langle u_1\right\rangle\right|r_{\mathrm{h}}\rho_{\mathrm{m}}/\mu, \tag{VI.30}$$

$$\epsilon_{\mathrm{s}}=\phi\rho_{\mathrm{m}}c_p/(1-\phi)\rho_{\mathrm{s}}c_{\mathrm{s}}, \quad \epsilon_h=8ir_{\mathrm{h}}^2/b(\phi)\sigma^{1/3}\delta_\kappa^2, \tag{VI.31}$$

$$\delta_\kappa^2=2K/\omega\rho_{\mathrm{m}}c_p, \tag{VI.32}$$

$$\theta_p=\mathrm{phase}(\langle u_1\rangle)-\mathrm{phase}\,(p_1)\,, \quad \theta_T=\mathrm{phase}(\langle u_1\rangle)-\mathrm{phase}\left(\langle T\rangle_{u,1}\right), \tag{VI.33}$$

$$g_c=\frac{2}{\pi}\int_0^{\pi/2}\frac{dz}{1+R_1^{3/5}\cos^{3/5}(z)}, \quad g_v=-\frac{2}{\pi}\int_0^{\pi/2}\frac{\cos(2z)\,dz}{1+R_1^{3/5}\cos^{3/5}(z)}. \tag{VI.34}$$

Here, the spatial average oscillatory velocity $\langle u_1\rangle=\langle U_1\rangle/\phi A$, where ϕ is volumetric porosity and A is regenerator cross sectional area. These expressions were derived with the assumption that the thermal and viscous penetration depths are much larger than r_h.

The `PX...` segments can be used when friction factor and heat-transfer coefficient are power laws in Reynolds number. The derivation assumes fairly good thermal contact between fluid and solid. Area A is the total cross sectional area of the heat exchanger, VolPor ϕ is its volumetric porosity, and L is its length, so that $A\phi L$ is the total volume of gas in the heat exchanger. The value r_H = r_h is the hydraulic radius, defined as the ratio of total gas volume to gas–solid contact area. The steady-state friction factor and heat-transfer coefficients must be known by the user in power-law forms:

$$\begin{aligned} f &= f_{con}\,(R)^{-f_{exp}}\,, \\ St\,Pr^{2/3} &= h_{con}\,(R)^{-h_{exp}}\,, \end{aligned}$$

where Reynolds number R is defined in the usual way as

$$R=\frac{4Ur_h\rho_m}{\phi A\mu}=\frac{4\,\langle u\rangle\,r_h\rho_m}{\mu}.$$

Note: this is Fanning friction factor, the friction factor used by Kays and London, so that instantaneously

$$\frac{dp}{dx}=\frac{f}{r_h}\frac{1}{2}\rho\,\langle u\rangle^2=\frac{\mu}{8r_h^2}\,f\,R\,\langle u\rangle\,. \tag{VI.35}$$

The pressure drop is computed using

$$\frac{dp_1}{dx}=-i\omega\rho_m\,\langle u_1\rangle-I_f\,\frac{\mu_m}{8r_h^2}\,f_{con}\,|R_1|^{1-f_{exp}}\,\langle u_1\rangle$$

where

$$I_f = \frac{2}{\pi}\int_0^\pi \sin^{3-f_{exp}} \omega t \, d(\omega t).$$

Thermal-relaxation effects due to oscillating compressibility are computed using the same equations for $d\langle u_1 \rangle /dx$ as for screen heat exchangers but with

$$\begin{aligned} g_c &= |R_1|^{h_{exp}-1} \frac{2}{\pi}\int_0^{\pi/2} \cos^{h_{exp}-1} \omega t \, d(\omega t), \\ g_v &= -|R_1|^{h_{exp}-1} \frac{2}{\pi}\int_0^{\pi/2} \cos 2\omega t \cos^{h_{exp}-1} \omega t \, d(\omega t), \\ b(\phi) &= h_{con}. \end{aligned}$$

The metal temperature is computed relative to the fluid mean temperature using

$$\Delta T = I_h \frac{\dot{Q}}{K\, h_{con}\, |R_1|^{1-h_{exp}}} \frac{r_h^2}{\phi A x_{\text{eff}}}, \tag{VI.36}$$

where $x_{\text{eff}} = \min\{2\,|\langle u_1\rangle|\,/\omega,\ L\}$, R_1 is the Reynolds number amplitude (based on the *amplitude* of the velocity), and

$$I_h = \frac{4}{\pi}\int_0^{\pi/2} \cos^{h_{exp}+1} \omega t \, d(\omega t) = \frac{h_{exp}}{h_{exp}+1}\frac{4}{\pi}\int_0^{\pi/2} \cos^{h_{exp}-1} \omega t \, d(\omega t). \tag{VI.37}$$

(The second form of I_h, obtained from the first via integration by parts, expresses I_h in terms of g_c above.) To maintain DeltaE 's high speed, the trigonometric integrals are not evaluated by DeltaE; we use simple functional fits to these integrals in DeltaE.

In `HX...`s and `TX...`s, metal temperature is computed relative to fluid mean temperature using

$$\Delta T = \frac{\dot{Q}}{K}\frac{y_{\text{eff}}}{\Pi x_{\text{eff}}} \tag{VI.38}$$

where

$$\begin{aligned} x_{\text{eff}} &= \min\{\text{peak-to-peak displacement amplitude, HX length}\} \\ y_{\text{eff}} &= \min\{\delta_\kappa, r_h\}, \end{aligned}$$

with hydraulic radius r_h equal to y_0 for `HX...`s and equal to half the circular pore radius for `TX...`s. This expression may be quite inaccurate, but we believe it is better than nothing. A little experimental evidence for it is presented in *J. Acoust. Soc. Am.* **92**, 1151 (1992). This expression and the duct turbulence algorithm are the only computations in DeltaE that are not correct in the acoustic approximation. If you dislike it, use the gas temperatures (available as outputs in the stack segment) instead of the metal temperatures for plotting or targeting (using math segments).

In `SX...`s, the metal temperature is computed relative to fluid mean temperature using

$$\Delta T = \frac{\dot{Q}}{K} \frac{r_h^2 (g_c - g_v)}{b(\phi)\phi A x_{\text{eff}}} \tag{VI.39}$$

where again $x_{\text{eff}} = \min\{\text{peak-to-peak displacement amplitude, HX length}\}$.

In `PX...`s, the metal temperature is computed relative to the fluid mean temperature using

$$\Delta T = I_h \frac{\dot{Q}}{K\, h_{\text{con}} R_1^{1-h_{\text{exp}}}} \frac{r_h^2}{\phi A x_{\text{eff}}}, \tag{VI.40}$$

where $x_{\text{eff}} = \min\{2\left|\langle u_1 \rangle\right|/\omega,\ L\}$, R_1 is the Reynolds number based on the amplitude of the velocity, and

$$I_h = \frac{4}{\pi} \int_0^{\pi/2} \cos^{h_{\text{exp}}+1} \omega t \, d(\omega t). \tag{VI.41}$$

Positive heat $\dot{Q}$ (parameter "e") flows into the apparatus. There are two kinds of $\dot{H}$ calculations in heat exchangers. For almost all cases, the heat flow $\dot{Q}$ is an input for each pass of DeltaE. However, for `'..LASt`, and for exchangers located downstream of stacks in conduction mode, the heat is calculated as a result, usually assuming that $\dot{H}_2 = \dot{E}_2$ in the subsequent segment but assuming $\dot{H}_2 = 0$ if the subsequent segment is an `INS...`. (If this seems confusing, you will appreciate why DeltaE is evolving away from the use of `INS...`, `...FRST`, `...MIDL`, and `...LAST` segments.

Most cases: $\dot{H}_{\text{out}} = \dot{H}_{\text{in}} + \dot{Q}$.

`'LASt`: $\dot{Q} = \dot{H}_{\text{out}} - \dot{H}_{\text{in}}$. $\dot{H}_{\text{out}} =$ [0 if next segment is `INSDUct` or `INSCOne`; $\dot{E}_{\text{out}}$ otherwise].

B.5. Stacks

Segment types: `STKSLab, STKREct, STKCIrc, STKDUct, STKCOne, STKPIns, STKSCreen, STKPOwerlaw`

Sample input-file segments:

```
STKSLAB   parallel-plate stack
SAMEAS 1  Area
0.724     GasA/A
7.85e-2 m Length
1.8e-4 m  y0 (half the plate spacing)
4.0e-5 m  Lplate (half the plate thickness)
SAMEAS 0  Gas
kapton    Solid
```

```
STKRECT  rectangular-pore stack
SAMEAS 1  Area
0.694     GasA/A
7.85e-2 m Length
2.0e-4 m   a (half of pore width)
4.0e-5 m  Lplate (half the plate thickness)
4.0e-4 m   b  (pore area is 2a times 2b)
SAMEAS 0  Gas
stainless Solid
```

```
STKCIRC approximates hexagonal honeycomb stack
SAMEAS 1  (m^2) total area
0.81     gas area/total area
0.279    (m) length
0.50e-3  (m) radius of circular pore
0.05e-3  (m) L:half of sht thcknss
helium    gas type
stainless stack material

STKDUCT  boundary-layer approx
0.01 m2 area of gas
0.4  m perimeter (this duct is square)
1. m length
0.001 m2 wall material's cross-sectional area
helium
stainless

STKCONE  boundary-layer w/ taper
0.01  m2 area of gas
0.35  m  perimeter
1. m length
sameas 8a
sameas 8b
0.001    fwall
helium
stainless

STKPINS  Muller/Keolian pinstack invention
sameas 2a a area   m^2
3.2e-4    b 2y0     m   2y0 = nearest-neighbor center-to-center distance
!                             in the hexagonal lattice
0.1       c Length  m
4.e-5     d R pin   m   pin radius
helium
stainless

 STKSCreen      a screen regenerator
 sameas  1a  a Area      m^2  cross section of regenerator
    .673     b VolPor         volumetric porosity
   5.500E-02 c Length     m
   1.830E-05 d   r_H      m   hydraulic radius
    .300     e KsFrac         fudge factor F for solid conduction
 sameas  0  Gas type
 stainless  Solid type

 STKPOwerlaw      an etched foil regenerator
 sameas  1a  a Area      m^2  cross section of regenerator
    .700     b VolPor         volumetric porosity
    0.04     c Length     m
   40.e-6    d   r_H      m   hydraulic radius
    .300     e KsFrac         fudge factor for solid conduction
   36.       f f_con
   1.0       g f_exp
```

```
  24.       h h_con
  0.8       i h_exp
sameas  0  Gas type
stainless  Solid type
```

Use:

Use STK* segments for segments having $dT_m/dx \neq 0$: the stacks of standing-wave thermoacoustic engines and refrigerators, the regenerators of Stirling engines and refrigerators, and the pulse tubes and thermal-buffer tubes of orifice pulse-tube refrigerators, thermoacoustic-Stirling hybrids, etc..

Use STKSLab for parallel-plate or jellyroll stacks (or regenerators). Use STKREct for square or rectangular pores whose aspect ratio is not large (see Ref. [19].). Use STKCIrc for circular or hexagonal pores. Use STKPIn for stacks comprised of pin arrays (see Ref. [20]). If pore size or plate separation is much greater than thermal and viscous penetration depths, use STKDUct or STKCOne. Use STKSCreen for stacked-screen regenerator (see Ref. [10]). Use STKPOwerlaw for etched-foil regenerators, or any other regenerator for which friction factor and heat-transfer coefficients follow power laws in Reynolds number.

In a STKCOne, the variable fwall (parameter 'f') determines the wall thickness. If fwall<1, the wall thickness is constant, but if fwall>1, the wall thickness is proportional to the local cone radius (so that the wall stress can be constant—the minimum-weight, minimum-thermal-conductivity design). Specifically,

For fwall<1, wall thickness =fwall$\times$ initial perimeter

For fwall>1, wall thickness = local perimeter/fwall, so
local wall cross-sectional area = (local perimeter)^2/fwall.

Computation algorithm:

Except in STKSCreen and STKPOwerlaw, pressure propagates according to Rott's wave equation, written in the form

$$\begin{aligned}\frac{dp_1}{dx} &= -\frac{i\omega\rho_m}{(1-f_\nu)A_{\text{fluid}}}U_1,\\ \frac{dU_1}{dx} &= -\frac{i\omega A_{\text{fluid}}}{\rho_m a^2}\left(1+\frac{(\gamma-1)f_\kappa}{1+\epsilon_s}\right)p_1+\frac{\beta(f_\kappa-f_\nu)}{(1-f_\nu)(1-\sigma)(1+\epsilon_s)}\frac{dT_m}{dx}U_1,\end{aligned} \quad \text{(VI.42)}$$

subject to the condition that energy flow $\dot{H}_2$ is independent of x, which imposes the following condition on $T_m(x)$:

$$\frac{dT_m}{dx} = \frac{\dot{H}_2 - \frac{1}{2}\Re\left[p_1\tilde{U_1}\left(1 - \frac{T_m\beta(f_\kappa - \tilde{f}_\nu)}{(1+\epsilon_s)(1+\sigma)(1-\tilde{f}_\nu)}\right)\right]}{\frac{\rho_m c_p |U_1|^2}{2\omega A_{\text{fluid}}(1-\sigma)|1-f_\nu|^2}\Im\left[\tilde{f}_\nu + \frac{(f_\kappa - \tilde{f}_\nu)(1+\epsilon_s f_\nu/f_\kappa)}{(1+\epsilon_s)(1+\sigma)}\right] - A_{\text{fluid}}K - A_{\text{solid}}K_{\text{solid}}}. \tag{VI.43}$$

For `STKSLab`,

$$\begin{aligned} f_\kappa &= \frac{\tanh[(1+i)y_0/\delta_\kappa]}{(1+i)y_0/\delta_\kappa}, \quad f_\nu = \frac{\tanh[(1+i)y_0/\delta_\nu]}{(1+i)y_0/\delta_\nu}, \\ & \epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{\tanh[(1+i)y_0/\delta_\kappa]}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.44}$$

For `STKREct`,

$$\begin{aligned} f_\kappa &= 1 - \frac{64}{\pi^4}\sum_{\substack{m,n\\ \text{odd}}} \frac{1}{m^2n^2Y_{mn}(\delta_\kappa)}, \quad f_\nu = 1 - \frac{64}{\pi^4}\sum_{\substack{m,n\\ \text{odd}}} \frac{1}{m^2n^2Y_{mn}(\delta_\nu)}, \\ & \epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)ab/\delta_\kappa(a+b)}{\tanh[(1+i)\ell/\delta_s]}, \\ \text{where} \qquad Y_{mn}(\delta) &= 1 - i\frac{\pi^2\delta^2}{8a^2b^2}(b^2m^2 + a^2n^2) \end{aligned} \tag{VI.45}$$

For `STKCIrc`,

$$\begin{aligned} f_\kappa &= \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \\ & \epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)r_0/2\delta_\kappa}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.46}$$

For `STKDUct` or `STKCOne`,

$$\begin{aligned} f_\kappa &= (1-i)\Pi\delta_\kappa/2A, \quad f_\nu = (1-i)\Pi\delta_\nu/2A, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{1}{\tanh[(1+i)\ell/\delta_s]}\text{ , where } \ell = \frac{\text{wall x-sect area}}{\text{perimeter}}, \end{aligned} \tag{VI.47}$$

so long as $2A/\Pi\delta_\nu > 30$. Otherwise, for $2A/\Pi\delta_\nu < 25$, the functions are the same as for `STKCIrc`. In between, a linear combination is used.

For `STKPIns`,

$$\begin{aligned} f_\nu &= -\frac{\delta_\nu}{(i-1)}\frac{2r_i}{r_o^2 - r_i^2}\frac{Y_1[(i-1)r_o/\delta_\nu]J_1[(i-1)r_i/\delta_\nu] - J_1[(i-1)r_o/\delta_\nu]Y_1[(i-1)r_i/\delta_\nu]}{Y_1[(i-1)r_o/\delta_\nu]J_0[(i-1)r_i/\delta_\nu] - J_1[(i-1)r_o/\delta_\nu]Y_0[(i-1)r_i/\delta_\nu]}, \\ f_\kappa &= -\frac{\delta_\kappa}{(i-1)}\frac{2r_i}{r_o^2 - r_i^2}\frac{Y_1[(i-1)r_o/\delta_\kappa]J_1[(i-1)r_i/\delta_\kappa] - J_1[(i-1)r_o/\delta_\kappa]Y_1[(i-1)r_i/\delta_\kappa]}{Y_1[(i-1)r_o/\delta_\kappa]J_0[(i-1)r_i/\delta_\kappa] - J_1[(i-1)r_o/\delta_\kappa]Y_0[(i-1)r_i/\delta_\kappa]}, \end{aligned}$$

and

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{J_0(\sqrt{-i\omega/\kappa_s}r_i)}{J_1(\sqrt{-i\omega/\kappa_s}r_i)} f_\kappa\sqrt{-i\omega/\kappa}\frac{r_o^2 - r_i^2}{2r_i}. \tag{VI.48}$$

In STKSLabs, STKRECTs, STKCIrcs, and STKPIns, the “Area” (the first line of the input file) is the total cross sectional area of the stack assembly, including both fluid cross section and solid cross section. In STKSLabs, STKRECTs, and STKCIrcs, $A_{\text{fluid}} = (\texttt{Area}) \times (\texttt{GasA/A})$ and $A_{\text{solid}} = (\texttt{Area}) \times (1 - \texttt{GasA/A})$. Plate half thickness (the 4th line of the input file) is used only for computing ϵ_s, not for computing heat conduction along x or what fraction of the Area is available to the fluid. This allows separate accounting for area blocked by “ideal” fins and by support struts or other structure. In most cases, ϵ_s is near 0, so plate thickness need not be specified with much accuracy; GasA/A is far more important. Because of the need to compute specialized functions, STKCIrcs compute more slowly than STKSLabs or STKDUcts; STKPIns are slower still, and STKREcts are very slow, especially for large aspect ratios. Hence, in the latter case, we recommend that STKSLabs be used until initial guesses and geometry are very close to finalized.

In stacked screen regenerators, pressure, volume flow rate, mean temperature evolve according to

$$\frac{dp_1}{dx} = -i\omega\rho_{\text{m}}\left[1 + \frac{(1-\phi)^2}{2(2\phi-1)}\right]\langle u_1\rangle - \frac{\mu}{r_{\text{h}}^2}\left(\frac{c_1(\phi)}{8} + \frac{c_2(\phi)R_1}{3\pi}\right)\langle u_1\rangle, \tag{VI.49}$$

$$\begin{aligned}\frac{d\langle u_1\rangle}{dx} &= -\frac{i\omega\gamma}{\rho_{\text{m}}a^2}p_1 + \beta\frac{dT_{\text{m}}}{dx}\langle u_1\rangle + \\ &\quad i\omega\beta\left[\frac{T_{\text{m}}\beta}{\rho_{\text{m}}c_p}\frac{\epsilon_{\text{s}} + (g_c + e^{2i\theta_p}g_v)\epsilon_h}{1+\epsilon_{\text{s}} + (g_c + e^{2i\theta_T}g_v)\epsilon_h}p_1 - \frac{1}{i\omega}\frac{dT_{\text{m}}}{dx}\frac{\epsilon_{\text{s}} + (g_c - g_v)\epsilon_h}{1+\epsilon_{\text{s}} + (g_c + e^{2i\theta_T}g_v)\epsilon_h}\langle u_1\rangle\right],\end{aligned} \tag{VI.50}$$

$$\begin{aligned}\frac{dT_{\text{m}}}{dx} &= \left\{\Re\left[\left(T_{\text{m}}\beta\frac{\epsilon_{\text{s}} + \epsilon_h(g_c + e^{2i\theta_p}g_v)}{1+\epsilon_{\text{s}} + \epsilon_h(g_c + e^{2i\theta_T}g_v)} + 1 - T_{\text{m}}\beta\right)p_1\widetilde{\langle u_1\rangle}\right] - \frac{2\overline{H_2}}{\phi A}\right\} \\ &\quad /\left\{\frac{\rho_{\text{m}}c_p}{\omega}\Im\left[\frac{\epsilon_{\text{s}} + \epsilon_h(g_c - g_v)}{1+\epsilon_{\text{s}} + \epsilon_h(g_c + e^{2i\theta_T}g_v)}\right]\langle u_1\rangle\widetilde{\langle u_1\rangle} + 2K_{\text{eff}}\frac{1-\phi}{\phi}\right\},\end{aligned} \tag{VI.51}$$

using

$$c_1(\phi) = 1268 - 3545\phi + 2544\phi^2, \quad c_2(\phi) = -2.82 + 10.7\phi - 8.6\phi^2, \tag{VI.52}$$

$$b(\phi) = 3.81 - 11.29\phi + 9.47\phi^2, \tag{VI.53}$$

$$R_1 = 4\left|\langle u_1\rangle\right| r_{\text{h}}\rho_{\text{m}}/\mu, \tag{VI.54}$$

$$\epsilon_{\text{s}} = \phi\rho_{\text{m}}c_p/(1-\phi)\rho_{\text{s}}c_{\text{s}}, \quad \epsilon_h = 8ir_{\text{h}}^2/b(\phi)\sigma^{1/3}\delta_\kappa^2, \tag{VI.55}$$

$$\delta_\kappa^2 = 2K/\omega\rho_{\text{m}}c_p, \tag{VI.56}$$

$$\theta_p = \text{phase}(\langle u_1\rangle) - \text{phase}\left(p_1\right), \quad \theta_T = \text{phase}(\langle u_1\rangle) - \text{phase}\left(\langle T\rangle_{u,1}\right), \tag{VI.57}$$

$$g_c = \frac{2}{\pi}\int_0^{\pi/2}\frac{dz}{1 + R_1^{3/5}\cos^{3/5}(z)}, \quad g_v = -\frac{2}{\pi}\int_0^{\pi/2}\frac{\cos(2z)\,dz}{1 + R_1^{3/5}\cos^{3/5}(z)}. \tag{VI.58}$$

These expressions were derived with the assumption that viscous and thermal penetration depths are much larger than r_h. Here, the spatial average oscillatory velocity $\langle u_1 \rangle = \langle U_1 \rangle / \phi A$, where ϕ is volumetric porosity and A is regenerator cross sectional area; and $K_{\text{eff}} = FK_s$ where F is a factor to reduce thermal conduction along x due to the poor thermal contact between adjacent screen layers (Radebaugh [12] recommends $F \approx 0.1$). F can also be used to account for conduction in the case surrounding the regenerator.

`STKPOwerlaw` segments are calculated in the same manner as `STKSCRN`'s, with a few exceptions. The friction factor and heat transfer coefficients are given by

$$\begin{aligned} f &= f_{\text{con}} R^{-f_{\text{exp}}}, \\ St\, Pr^{2/3} &= h_{\text{con}} R^{-h_{\text{exp}}}, \end{aligned}$$

where Reynolds number R is defined in the usual way as

$$R = \frac{4Ur_h\rho}{\phi A \mu}.$$

[Note: this is Fanning friction factor, the friction factor used by Kays and London, so that instantaneously $dp/dx = (f/r_h)\,\frac{1}{2}\rho u^2$.] The pressure equation is replaced by

$$\frac{dp_1}{dx} = -i\omega\rho_{\text{m}} \left[1 + \frac{(1-\phi)^2}{2(2\phi-1)}\right] \langle u_1 \rangle - I_f \frac{\mu}{8r_{\text{h}}^2} f_{\text{con}} R_1^{1-f_{\text{exp}}} \langle u_1 \rangle \tag{VI.59}$$

where

$$I_f = \frac{2}{\pi} \int_0^{\pi} \sin^{3-f_{\text{exp}}}(z)dz \tag{VI.60}$$

In the volume flow rate and mean temperature equations, these parameters are redefined for the power law stack:

$$g_c = R_1^{h_{\text{exp}}-1} \frac{2}{\pi} \int_0^{\pi/2} \cos^{h_{\text{exp}}-1}(z)dz \tag{VI.61}$$

$$g_v = -R_1^{h_{\text{exp}}-1} \frac{2}{\pi} \int_0^{\pi/2} \cos 2z \cos^{h_{\text{exp}}-1}(z)dz \tag{VI.62}$$

$$b(\phi) = h_{\text{con}}. \tag{VI.63}$$

Values of $\phi, r_{\text{h}}, F = K_{\text{eff}}/K_s, f_{\text{con}}, f_{\text{exp}}, h_{\text{con}}$, and h_{extp} for particular etched foil regenerators can be obtained from Ran Yaron.

In both `STKSCreen` and `STKPOwerlaw` segments, the trigonometric integrals are not evaluated by DELTAE; these integrals were performed once, off-line. We now use simple functional fits during computation of either segment type.

Computation with mean-flow enabled:

To enable mean-flow capability, put segment `MEANFLOW` in position 1, immediately following the `BEGIN` segment (see Section VI B.6 for details). Its presence establishes a constant mean mass flux through the subsequent segments, and modifies the behavior of `STKSLAB`'s, `STKCIRC`'s, `STKRECT`'s, `STKSCREEN`'s, and `STKPOWRLAW`'s appropriately. (It does not yet have any effect in other segments, so for example don't use a `TBRANCH` expecting to be able to split up the mean flow.) See Refs. [21, 22]. In the affected segments,

- Mean volume flow rate U_m can vary with x, due to x dependence of T_m,
- The dependence of T_m on x is changed by the nonzero U_m.

The display of information is slightly changed. As always, `Hdot` is the "acoustic"-plus-longitudinal-conduction part of the total energy flux—not including the new mean-flow part of the total energy flux, $\dot{H}_m = \rho_m w_m U_m$, where w_m is the enthalpy per unit mass. The `.out` file shows U_m and $\dot{H}_m$ at the end of each of the segments affected by meanflow.

The change in mean velocity is computed using

$$\frac{dU_m}{dx} = U_m \beta \frac{dT_m}{dx},$$

which comes from

$$\frac{dU_m}{dx} = -\frac{U_m}{\rho_m}\frac{d\rho_m}{dx}.$$

The temperature gradient in stacks is computed using constancy of total energy flux $\dot{H}_{\text{tot}} = \dot{H} + \dot{H}_m$, where $\dot{H}$ is the old, original "acoustic"-plus-longitudinal-conduction energy flux and $\dot{H}_m$ is the mean-flow contribution. In other words, we solve this equation for dT_m/dx :

$$\begin{aligned}
\dot{H}_{\text{tot}} &= \frac{1}{2}\Re\left[p_1\widetilde{U_1}\left(1 - \frac{T_m\beta(f_\kappa - \widetilde{f_\nu})}{(1+\epsilon_s)(1+\sigma)(1-\widetilde{f_\nu})}\right)\right] \\
&\quad + \frac{\rho_m c_p}{2A_{\text{fluid}}\omega(1-\sigma)\left|1-f_\nu\right|^2}\frac{dT_m}{dx}U_1\widetilde{U_1}\Im\left[\widetilde{f_\nu} + \frac{(f_\kappa - \widetilde{f_\nu})(1+\epsilon_s f_\nu/f_\kappa)}{(1+\epsilon_s)(1+\sigma)}\right] \\
&\quad - (A_{\text{fluid}}K + A_{\text{solid}}K_s)\frac{dT_m}{dx} + \rho_m w_m U_m
\end{aligned}$$

and use $\dot{H}_{\text{tot}}$=constant to integrate our way through a stack. The only new thing in this equation is the final term, $\rho_m w_m U_m$. As usual, the value of $\dot{H}_{\text{tot}}$ is determined by conditions in segments other than the stack, such as the value of U_m and the heat flows in adjacent heat exchangers.

B.6. Begin, ends, mean-flow mode, insulate/conduct mode

Segment types: `TITLE, BEGIN, INSULate, CONDUct, HARDEnd, SOFTEnd, MEANFLOW`

Sample input-file segments:

```
TITLE   comments here are reproduced in .DAT and .OUT

BEGIN
1.0e6 Pa    Mean P
500. Hz     Freq.
300. K      T-beg
3.0e4 Pa    |p|@0
0.0 deg     Ph(p)0
5.0e-4 m3/s |V|@0
0.000 deg   Ph(V)0
0.00  W     Hdot
helium      Gas type

INSULATE

CONDUCT

MEANFLOW
1.E-04     U_m    m^3/s
sameas  0  Gas type

HARDEND
0.000     R(1/z)
0.000     I(1/z)
SAMEAS 0  Gas type

SOFTEND
0.  Re(z)
0.  Im(z)
water
```

Use:

The initial segments of all input files must be `TITLE` and `BEGIN`. `TITLE` is just used to give a comment field that gets reproduced in all subsequent files, so put a descriptive name in its comment field. `BEGIN` is counted as the zeroth segment of the file. It is used to initialize variables that are shared by subsequent segments (i.e., frequency, mean pressure, and $\dot{H}$), and the five variables required by each pass of DeltaE to get started (i.e., real and imaginary parts of pressure amplitude and volume flow rate, and mean temperature).

Gas type isn't really used here, but you have to give one anyway.

Parameter "h" in `BEGIN` is the initial value of $\dot{H}$. When it is set equal to $\dot{E}$ (enter "Edot" when (m)odifying parameter 0h), it disappears from the display, for backward compatibility with older versions of DELTAE.

`BEGIN` segments can be used anywhere in a DELTAE file, in order to set any or all of its parameters to new values. This can be useful for packing two or more related DELTAE models into a single file.

`INSULATE` causes $\dot{H}$ calculations in subsequent segments to be done in insulated mode; `CONDUCT` causes $\dot{H}$ calculations in subsequent segments to be done in conduction mode. Insulated mode basically enforces conservation of energy, so that the heat cannot leak out of the system via the side walls of ducts, impedances, etc.; such heat must find its way to a nearby heat exchanger.

The insulated/conduction mode does not affect stacks or most heat exchangers. Conduction mode is the default, and is the only mode that existed in DELTAE prior to version 5. We hope that DELTAE will evolve more and more toward an exclusively insulated mode as the years pass. Toward that end, segments such as `TBRANCH` and `UNION` also conserve energy when insulated mode is in effect.

`MEANFlow`, when used, should always be in segment 1, immediately following the `BEGIN` statement. Its presence establishes a constant mean mass flux through the subsequent segments, and modifies the behavior of mean-flow savvy segments (currently, these are: `HXFRST`, `HXMIDL`, `HXLAST`, `STKSLAB`, `STKRECT`, and `STKCIRC`). This feature is still very experimental; its effects on stack computation algorithms are described in the previous subsection.

Often, the final segment (except math segments) will be either `HARDEnd` or `SOFTEnd`. These contain two or three default targets. Use `HARDEnd` if you want the complex volume flow rate at the end of the apparatus to be zero. This is the usual case in a closed system. Use `SOFTEnd` if you want complex pressure amplitude at the end to be zero. We find this useful for symmetrical systems, where `SOFTEnd` indicates that the rest of the apparatus is a mirror image of what is in the input file, and forces a complex pressure node. In both `'END`s, the complex impedances are made dimensionless according to $z = Ap_1/\rho a U_1$, where A is the area of the last segment with an area, and ρ and a are evaluated at the local temperature.

Disable these as targets if you want DELTAE to ignore the impedance. This approach is useful in early stages of debugging a new model that doesn't readily converge—it may let you see what's out of whack. Set these targets nonzero to model a nonzero end impedance—or use `BRANCh` or `OPNBRanch`.

The third default target in the `...END` segments, $\dot{H}$, appears only in insulated mode.

Computation algorithm:

`BEGIN` *sets* values of T_m, p_1, U_1, and $\dot{H}$. `INSULate` and `CONDUct` sets and clears (respectively) a bit that controls computations in most subsequent segments. `MEANFlow` *sets* the value of U_m. The `'ENDS` leave all these variables unchanged.

B.7. Math segments

Segment types: `RPNTArget, VOLMTarget, FREETarget, DIFFTarget, PRODTarget, QUOTArget, EFFRTarget, COPRTarget, CONSTants.`

Sample input-file segments:

```
RPNTARGET   magU1 over omega A
0.01   desired gas displacement amplitude after segment five
5C 2 PI * / 0b / 5a /

VOLMTARGET
0.50    a targeted volume (cubic meters)
1A      b BegAddr
10A     c EndAddr

FREETARGET
500.   Watts of power targeted at driver
3G     Address of computed power at driver

DIFFTARGET
0.00    a targeted difference
1B      b D1Addr
1L      c D2Addr

PRODTARGET similar to DIFFTarget.
0.00    a targeted product
1B      b M1Addr
1L      c M2Addr

QUOTARGET
1.0  desired quotient
1A   numerator address
6A   denominator address

EFFRTARGET
0.2  desired 2nd law efficiency
7F   work (numerator address)
4G   heat (denominator address)
4H   T hot address
6H   T cold address

COPRTARGET
0.2  desired 2nd law efficiency
7G   heat (numerator address)
2F   work (denominator address)
6H   T hot address
4G   T cold address
```

```
CONST       test of CONST
1.00      a    So                          2.250E+03 A So*PLo
2.00      b    Si                          0.000     B Si*PLi
3.00      c   C_1                           3.00     C   C_1
4.00      d   C_2                           4.00     D   C_2
5.00      e   C_3                           5.00     E   C_3
6.00      f   C_4                           6.00     F   C_4
7.00      g   C_5                           7.00     G   C_5
8.00      h   C_6                           8.00     H   C_6
9.00      i   C_7                           9.00     I   C_7
10.0      j   C_8                           10.0     J   C_8
11.0      k   C_9                           11.0     K   C_9
12.0      l   C_10                          12.0     L   C_10
helium      Gas type
ideal       Solid type
```

Use:

Use this class of segments to create targets other than DeltaE default targets (which include only end impedances and heat exchanger heats and temperatures). You may also use them for simple arithmetic operations.

The introduction of `RPNTARGet` with version 3.9 makes the other math segments obsolete (with the possible exception of `VOLMTARget`).

Computation algorithms:

`RPNTArget`: Result is computed by interpreting the instruction line (line “b”) in Reverse Polish Notation. Items in the instruction line must be separated by blanks. All commands (except Bessel functions) can be either upper or lower case (but not a mixture of the two). The command line in a single `RPNTArget` must not be longer than 80 characters. (Note that DeltaE rewrites the string so that each segment number occupies 2 characters regardless of its value. The practical typed length is therefore a little bit shorter.) As with RPN calculators, when operators do not consume all numbers on the ‘stack’, more than one output is generated, a feature that can be exploited to reference multiple results. The stack grows downward, from `A`–`J`. Valid operators and inputs are summarized in Tables VI.1, VI.2, and VI.3.

`VOLMTarget`: result = sum of the volumes in all duct, cone, stack, compliance, and heat exchanger segments beginning with `BegAddr` and ending with `EndAddr` (parameter letters are inconsequential). Porosity is not used in calculating this volume—that is, porosity is always effectively 100%. This segment is intended to give an indication of the overall size of a design for doing tradeoff analysis.

`FREETarget`: This simplest math segment performs no computation. It simply has two input parameters: the target value, and an output address to which the solver can compare the target value.

`DIFFTarget:` result = `[D1Addr]` − `[D2Addr]`, where `[]` signifies value calculated at this address.

`PRODTarget:` result = `[M1Addr]` × `[M2Addr]`.

`QUOTArget:` result = `[NumAdr]` / `[DenAdr]`.

`EFFRTarget:` result $= \frac{W}{Q_h} \frac{T_h}{T_h - T_c}$

`COPRTarget:` result $= \frac{Q_c}{W} \frac{T_h - T_c}{T_c}$

Table VI.2: List of `RPNTArget` constants

Item	**Description**	**comment or examples**
Numeric		
<Output>	(Seg. # and cap letter)	5D
<Input>	(Seg. # and l.c. letter)	5c
<Constant>	real or complex	8.314; 6.02e23; (-0.01, 1.03)
pi i	3.14159265 $\sqrt{-1}$	 complex
Thermophysical (position dependent):		
gamma	γ; c_p/c_v	
a	a; local sound speed	m/s
rho	ρ_m	kg/m^3
cp	c_p; heat capacity	J/kg/K
k0	K_0; conductivity	W/m/K
mu	μ; viscosity	kg/s/m
beta	β; expansion coefficent	1/K
dk	δ_κ; thermal penetration	m
dn	δ_ν; viscous penetration	m
rhos	ρ_s; solid density	kg/m^3
cs	c_s; solid heat capacity	J/kg/K
ks	K_s; solid conductivity	W/m/K
State Variables (position dependent):		
tm	T_m; mean temperature	K
w	ω; circular frequency	rad/s
f	f; frequency	Hz
pm	p_m; mean pressure	Pa
n1	gas fraction; binary mix	molar frac. of light gas
p1	p_1; oscillatory pressure	Pa; complex
U1	U_1; oscil. volume flow rate	m^3/s; complex
Error Codes		
====	ERROR CODE	illegal number place holder
@@Z	ERROR CODE	illegal addr. place holder
NoOp	ERROR CODE	illegal op. place holder

Table VI.3: List of `RPNTArget` operators

Item	Description	example or comment
+	$x = y + x$	
-	$x = y - x$	
*	$x = y * x$	
/	$x = y/x$	
^	$x = y^x$	
sqrt	$x = \sqrt{x}$	
sqrd	$x = x * x$	
~	$x = -x$	change sign
abs	$x = \|x\|$	absolute value
inv	$x = 1/x$	
real	$x =$real(x)	
imag	$x =$imag(x)	
mag	$x = \|x\|$	magnitude of complex x
conj	$x =$conj(x)	complex conjugate of x
arg		phase of complex x (degrees)
argr		phase of complex x (radians)
Stack manipulation:		
#	$y = x$, $x = x$	(lifts higher outputs also)
lstx	$y = x$, $x =$previous x	Lifts the stack and recalls value from before last math operation
a<>b	$x = y$; $y = x$	
sto	$S = x$	Store in register "S"
rcl	$x = S$	Recall from register "S"
'x' refers to parameter '`A`', the base of the stack. 'y' refers to next parameter in the stack ('`B`').		
All functions may be UPPER or lower case, but not MiXeD		

Table VI.4: List of `RPNTArget` functions

Item	Description	example or comment
conj	$x = \tilde{x}$	conjugate
sin; asin	$x = \sin(x)$;$x = \sin^{-1}(x)$	these trig functions in degrees
cos; acos	$x = \cos(x)$;$x = \cos^{-1}(x)$	and require real arguments
tan; atan	$x = \tan(x)$;$x = \tan^{-1}(x)$	
atan2	$x = \tan^{-1}(x/y)$	2 argument, 4 quadrant arctan
sinr	$x = \sin(x)$	these trig functions in radians
asinr	$x = \sin^{-1}(x)$	*real argument only*
cosr	$x = \cos(x)$	
acosr	$x = \cos^{-1}(x)$	*real argument only*
tanr	$x = \tan(x)$	
atanr	$x = \tan^{-1}(x)$	*real argument only*
atan2r	$x = \tan^{-1}(x/y)$	2 argument, 4 quadrant arctan
sinh	$x = \sinh(x)$	
cosh	$x = \cosh(x)$	
tanh	$x = \tanh(x)$	
J0*	$x = J_0(x)$	Bessel function of zero order
J1*	$x = J_1(x)$	Bessel function of first order
Y0*	$x = Y_0(x)$	Neumann function of zero order
Y1*	$x = Y_1(x)$	Neumann function of first order
log	$x = \log_e(x)$	natural log
exp	$x = e^x$	
log10	$x = \log_{10}(x)$	*real argument only*
tenx	$x = 10^x$	
min; max	$x = \min(x, y)$;$x = \max(x, y)$	*real argument only*
All functions accept and return complex arguments unless otherwise noted *Bessel functions are recognized as UPPER CASE only.		

B.8. Joining condition

Segment types: JOIN

Sample input-file segments

```
!---------------------------------  8 ---------------------------------
 SXFRST     p.t. hot h.x.
  1.1675E-04 a Area      m^2              1.3507E+05 A |p|       Pa
     0.6470  b VolPor                      -17.664   B Ph(p)     deg
  3.0000E-03 c Length     m               2.9183E-04 C |U|     m^3/s
  1.2000E-05 d   r_H      m                138.51    D Ph(U)     deg
     2.8164  e HeatIn     W      G         -13.711   E Hdot      W
   300.00    f Est-T      K     (t)        -18.030   F Edot      W
 sameas  0  Gas type                         2.8164  G Heat       W
 copper     Solid type                     300.00    H MetalT     K
!---------------------------------  9 ---------------------------------
 JOIN       first join example
                                          1.3507E+05 A |p|       Pa
                                           -17.664   B Ph(p)     deg
                                          2.9863E-04 C |U|     m^3/s
                                           138.51    D Ph(U)     deg
                                           -13.711   E Hdot      W
                                           -18.450   F Edot      W
                                           300.00    G T-beg      K
                                           324.67    H T-end      K
!--------------------------------- 10 ---------------------------------
 STKDUCT    the pulse tube
  1.1675E-04 a Area      m^2     S=-2     1.3511E+05 A |p|       Pa
  3.8307E-02 b Perim      m    Fnc(10a)    -17.530   B Ph(p)     deg
  7.0000E-02 c Length     m               2.7164E-04 C |U|     m^3/s
  5.8678E-05 d WallA     m^2               166.83    D Ph(U)     deg
                                           -13.711   E Hdot      W
                                           -18.298   F Edot      W
                                           324.67    G T-beg      K
 sameas  0  Gas type                        57.932   H T-end      K
 stainless  Solid type                       0.1518  I StkEdt     W
!--------------------------------- 11 ---------------------------------
 RPNTARGET  ratio of cold end displacement to p.t. length
     0.1000  a Target           (t)     9.2000E-02 A RPNval
 10C 2 / PI /  0b / 10a / 10c /
!--------------------------------- 12 ---------------------------------
 JOIN       second example;  note that RPNTARgs can go between
                                          1.3511E+05 A |p|       Pa
                                           -17.530   B Ph(p)     deg
                                          2.7854E-04 C |U|     m^3/s
                                           166.83    D Ph(U)     deg
                                           -13.711   E Hdot      W
                                           -18.763   F Edot      W
                                            57.932   G T-beg      K
                                            89.997   H T-end      K
!--------------------------------- 13 ---------------------------------
 RPNTARGET  ratio of warm end displacement to p.t. length
     6.0000  a Target           (t)     9.8838E-02 A RPNval
  8C 2 / PI /  0b / 10a / 10c /
!--------------------------------- 14 ---------------------------------
 SXMIDL     cold heat exchanger  adj rh
  1.1675E-04 a Area      m^2              1.5473E+05 A |p|       Pa
     0.6470  b VolPor                      -16.785   B Ph(p)     deg
  5.7150E-03 c Length     m               2.7985E-04 C |U|     m^3/s
  1.2000E-05 d   r_H      m                169.37    D Ph(U)     deg
     7.1278  e HeatIn     W      G          -6.5828  E Hdot      W
    90.000   f Est-T      K    =14H?       -21.525   F Edot      W
```

```
helium     Gas type                        7.1278  G Heat       W
copper     Solid type                     90.000   H MetalT     K
```

Use and Computation algorithm:

The JOIN segment accounts for small discontinuities in thermoacoustic variables at the interface between a heat exchanger or other isothermal segment and an unmixed, stratified adiabatic segment such as a pulse tube. The discontinuity in mean temperatures is first order in the acoustic amplitude, the discontinuity in volume flow rate is second order, and there is no discontinuity in pressure. References: Storch, Radebaugh, and Zimmerman NIST Technical Note 1343 for δT_m, Smith and Romm 27th IECEC 5.529 for δU_1, Swift "Thermoacoustics: A unifying perspective for some engines and refrigerators" and references therein for both δT_m and δU_1.

The discontinuity in temperature is:

$$T_{\text{out}} - T_{\text{in}} = \left(\frac{T_m \beta}{\rho_m c_p} |p_1| \sin\theta - \frac{|U_1|}{\omega A_{\text{fluid}}} \frac{dT_m}{dx} \right) F, \tag{VI.64}$$

where θ is the angle by which p_1 leads U_1 and the factor F is given by

$$F = \frac{A_{\text{gas}} k \; dT_m/dx}{\dot{H} - \dot{E}}. \tag{VI.65}$$

The factor F is a crude attempt to account for the two-dimensional nature of the JOIN problem within DeltaE's inherently 1-dimensional character. The derivation of the T discontinuity joining condition in the references cited above neglects the thermal conductivity of the wall of the tube and the boundary-layer shuttle entropy flux. If these are zero, then $F = 1$ and Eq. (VI.64) represents the joining condition derived in the references cited above. At the other extreme, if the solid wall of the duct or the boundary layer is effectively a thermal short circuit compared to bulk gas conductivity, then $F = 0$ and no temperature discontinuity appears in the JOIN segment.

The discontinuity in $|U_1|$ is

$$\begin{aligned} |U_1|_{\text{out}} &= |U_1|_{\text{in}} - \frac{8}{3\pi} \frac{(\gamma - 1)}{\rho_m a^2} |p_1| \, |U_1| \cos\theta \\ &= |U_1|_{\text{in}} - \frac{16}{3\pi} \frac{\gamma - 1}{\rho_m a^2} \dot{E}_2 \end{aligned} \tag{VI.66}$$

There is no discontinuity in p_1, in the phase of U_1, or in $\dot{H}_2$.

When it encounters a JOIN segment, DeltaE looks both upstream and downstream (ignoring unphysical segments such as RPNTARGETs) to figure out whether the "heat exchanger" is upstream and the “pulse tube” is downstream, or vice versa. Both A and dT_m/dx are obtained from the “pulse tube” or other open duct, with $dT_m/dx = 0$ for ordinary `ISO` and `INS-DUCT`s and `CONE`s, and dT_m/dx nonzero for `STKDUCT`s or `STKCONE`s. If the relevant duct is a `STKDUCT` or `STKCONE` *downstream* of the `JOIN` segment, DeltaE actually jumps ahead momentarily to evaluate dT_m/dx.

The `JOIN` segment seems to us to be most useful at the ends of a pulse tube, but it can also be used in standing-wave systems between a resonator duct and a heat exchanger.

Note: The `JOIN` feature cannot yet be used reliably with meanflow.

B.9. Tees and unions

Segment types: `TEE, TBRANch, UNION, HBRANch, HUNIOn`

Sample input-file segments:

```
TEE        branch file to load
branch.in

TBRAN      the fork
 4.412E+07 a Re(Zb) Pa-s/m^3  G
-3.528E+06 b Im(Zb) Pa-s/m^3  G
sameas  0  Gas type
ideal      Solid type

UNION      below the branch
     4    segment number of SOFTEND of the TBRANCH
    3.e4  |p| @ end (Pa)
     0.   ph(p) @end
sameas 0  Gas type
ideal      Solid type

HBRAN      fork with Hbran
 4.412E+07 a Re(Zb) Pa-s/m^3  G
-3.528E+06 b Im(Zb) Pa-s/m^3  G
 0.49      c  Hbran           G
sameas  0  Gas type
ideal      Solid type

HUNION    H matching joint      5
     10    segment number of SOFTEND of the HBRANCH
    4.e3  |p|End  Pa    =5A?
     0.   Ph(p)E        =5B?
   300.    T-est  K     =5G?
sameas 0  Gas type
ideal      Solid type
```

Use:

Use TBRANch for branched systems too complicated for BRANCh or OPNBRanch.

In insulated mode, TBRANCH acts like HBRANCH and UNION acts like HUNION.

When it encounters a TBRANch or HBRANch, DeltaE treats subsequent segments as the sequential members of a branch until it reaches a HARD- or SOFTEnd; then it "returns to the trunk," treating the rest of the segments as trunk members. If the system is multiply connected, a UNION or HUNION segment in the trunk tells DeltaE where to connect the branch's SOFTEnd back to the trunk.

If UNION is used, the branch's SOFTEnd impedance targets should not be used; instead, enable the UNION's targets to ensure that (complex!) p_1 is equal at the SOFTEnd of the branch and at the UNION in the trunk. The guessed branch impedance determines how the (complex) volume flow rate splits up at the TBRANch or HBRANch; volume flow rates add at the UNION or HUNION. The union targets are a special case in that their input values are dynamically rewritten by DeltaE during iterations, depending on the most recent results at the named SOFTEnd. The real input parameters (magnitude and phase of pressure) can have any value initially. DeltaE will overwrite them during each pass with the current magnitude and phase of pressure at the referenced SOFTEnd.

BRANCH and UNION are intended for duct networks, where temperature is constant and hence p_1 and U_1 are the variables of interest. For more complex systems, the segments HBRANCH and HUNION are energy-conserving versions of BRANCH and UNION. Use them if you are branching at locations where $\dot{H}_2 \neq \dot{E}_2$, such as at a branch to a second stage regenerator within a two-stage pulse tube refrigerator. HBRANCH incorporates a potential guess Hbran, giving the portion of the incoming energy that goes into the branch. Use Hbran as a guess to hit a target down the branch, such as a temperature. (Note: Prior to version 5, this parameter was the *fraction* of the total power going down the branch.) HUNION incorporates an additional potential target, that the temperature in the trunk at the union be equal to that at the associated branch end. Energy flows are added in HUNION.

When DeltaE encounters a TEE, it loads the named file into the model, and replaces the BEGIN segment of the branch file with a TBRANch segment. It tries to guess starting values for the complex branch impedance, and then adjusts the addresses in any sameas declarations and math segments occurring in the branch (or after the branch point) by the number of segments in the branch. Once the file has been read in, the TEE segment disappears—the .out file and (d)isplayed segments will be the composite model. The file may have any name (*e.g.* branch.in, stub.out, branch.tee), but it must be specified with the complete suffix.

Computation algorithm:

These segments leave T_m and p_1 unchanged.

At a TBRANch or HBRANch, $U_{\mathrm{branch}} = p_1/Z_1$ and $U_{\mathrm{trunk}} = U_{\mathrm{in}} - U_{\mathrm{branch}}$. At HBRANCH, $H_{\mathrm{branch}} = H_{\mathrm{bran}}$, $H_{\mathrm{trunk}} = H_{\mathrm{in}} - H_{\mathrm{branch}}$. At a UNION or HUNION, $U_{\mathrm{out}} = U_{\mathrm{in}} + U_{\mathrm{end}}$. At HUNION, $H_{\mathrm{out}} = H_{\mathrm{in}} + H_{\mathrm{end}}$. The "end" location is the branch's SOFTEnd, identified in the first input line of the UNION or HUNION segment. The 'BRANCHs have an output Edot_T to display the acoustic power flowing past the branch in the trunk.

B.10. Acoustical decomposition

Segment type: DECOMpose

Sample input-file segment:

```
DECOMp     Termination
  8.100E-03 a Area      m^2             15.8     A |Pin|     Pa
                                        31.9     B |Pref|    Pa
sameas  0  Gas type                     4.11     C RflCoe   W/W
ideal      Solid type                  -77.9     D PhI-R    deg
```

Use:

Use DECOM to decompose the acoustic field into *incident* and *reflected* pressure waves; that is, solve for $P_{\mathrm{in}}, P_{\mathrm{ref}}$, and $\phi_I - \phi_R$ in the equation

$$p_1 = P_{\mathrm{in}} e^{i(-kx+\phi_I)} + P_{\mathrm{ref}} e^{i(kx+\phi_R)}, \tag{VI.67}$$

where $P_{\mathrm{in}}, P_{\mathrm{ref}}$, and k are considered real for this segment.

Computation algorithm:

Since the segments surrounding the DECOM segment are generally lossy in DELTAE, its results are strictly valid only at that point. The magnitudes are calculated from

$$\begin{aligned} P_{\mathrm{in}} &= \tfrac{|p_1 + U_1 \rho a/A|}{2} \\ P_{\mathrm{ref}} &= \tfrac{|p_1 - U_1 \rho a/A|}{2} \end{aligned} \tag{VI.68}$$

and the phase difference is given by

$$\text{phase}\left(\frac{p_1 + U_1\rho a/A}{p_1 - U_1\rho a/A}\right) \tag{VI.69}$$

The sound power reflection coefficient, $(P_{\text{ref}}/P_{\text{in}})^2$, is also found and given as output `C`.

B.11. Thermophysical properties dump

Segment type: `THERMOphys`

Sample input-file segment:

```
THERMO
sameas 0
```

Use:

Use `THERMO` to provide a record of thermophysical properties and penetration depths at a given location in the apparatus. With plotting features, can be used to generate a table of thermophysical properties.

B.12. External file and program interfaces

Segment types: `BLKDAta, SYSEXec`

Sample input-file segments:

```
BLKDATA    datafile

SYSEXEC    myprog.exe < sysin.dat > sysout.dat
sameas 0b
sameas 2A
sameas 2C
sameas 10c
sameas 11F
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

Neither BLKDAta or SYSEXec use fluid or solid lines.

Use:

BLKDAta is a specialized segment designed for users who have a quantity of tabular data that they want DeltaE to follow. The data may represent experimental results, the results from other computational methods, or simply an irregular parameter space to be explored that is not well served by the simple plot loop facility (*e.g.* logarithmic or irregular step size, or parametric plotting). The data file is text format, delimited with spaces or tabs, and it may contain up to 14 columns and any number of rows, all with the same number of entries. The first one or two rows can optionally contain descriptive strings that are read into the parameters' description and unit fields, respectively. At present, only one BLKDATA segment per model is allowed. The number of inputs displayed in the .out file is equal to the number of columns in the associated data file.

The data file is assumed to have the suffix .blk and its root name is supplied in the title position of the segment. When the file is opened, the number of rows and columns are counted, and initial column titles are read in, if present. The number of input and output parameters for the segment is set to the number of columns in the file. If plotting is not enabled, only the first numeric row of the BLKDATA file is read in to fill the parameters once.

```
!--------------------------------  1 --------------------------------
 BLKDATA    test
   511.40    a Freq.  Hz
  6653.0     b |p|@0  Pa
    10.000   c SkrAmp  A
```

where the first lines of the file test.blk are as follows:

```
Freq.       |p|@0       SkrAmp
  Hz          Pa          A
 511.4       6653.        0.03
 511.0      14401.        0.07
```

If the file test.blk is a log of experimental data, we can ask DeltaE to to emulate the experimental conditions by inserting appropriate sameas statements, as in the following fragment that enforces the experimental applied speaker voltage, frequency, and pressure amplitude:

```
!-------------------------------  0 --------------------------------
```

```
 BEGIN      the setup
  7.8000E+04 a Mean P    Pa
 sameas  1a  b Freq.     Hz
   300.00    c T-beg     K
 sameas  1b  d |p|@0     Pa
     0.0000  e Ph(p)0    deg
     0.0000  f |U|@0    m^3/s
     0.0000  g Ph(U)0    deg
 air        Gas type
 ideal      Solid type
!--------------------------------  1 ---------------------------------
 BLKDATA    test
   511.40    a Freq.  Hz
  6653.0     b |p|@0  Pa
  3.0000E-02 c SkrAmp  A
!--------------------------------  2 ---------------------------------
 ENDCAP     speaker back
 sameas  4a  a Area      m^2             100.00    A |p|       Pa
                                          0.0000   B Ph(p)     deg
                                        1.1824E-08 C |U|     m^3/s
                                         180.00    D Ph(U)     deg
                                       -5.9121E-07 E Hdot      W
 sameas  0  Gas type                   -5.9121E-07 F Edot      W
 ideal      Solid type                 -5.9121E-07 G HeatIn    W
!--------------------------------  3 ---------------------------------
 ISODUCT    back duct
 sameas  4a  a Area      m^2              99.377   A |p|       Pa
  2.0000E-02 b Perim      m             5.5864E-03 B Ph(p)     deg
  1.2000E-02 c Length     m             5.3045E-06 C |U|     m^3/s
                                         -90.330   D Ph(U)     deg
                                       -1.5415E-06 E Hdot      W
 air        Gas type                   -1.5415E-06 F Edot      W
 ideal      Solid type                 -9.5034E-07 G HeatIn    W
!--------------------------------  4 ---------------------------------
 VESPEAKER  open back speaker
  1.5000E-04 a Area      m^2              68.286   A |p|    Hz
     4.6000  b   R      ohms             -91.445   B Ph(p)     deg
  5.5000E-05 c L         H              5.2965E-06 C |U|     m^3/s
     3.3000  d B x L    T-m              -90.455   D Ph(U)     deg
  4.0000E-04 e   M      kg              1.8081E-04 E Hdot      W
  8560.0     f   K      N/m             1.8081E-04 F Edot      W
     0.8000  g   Rm   N-s/m             1.8597E-03 G EdotIn    W       P
     0.1945  h  |V|     V                 0.1945   H Volts     V
   243.00    i Ph(V)   deg              2.2615E-02 I Amps      V
                                          32.263   J Ph(Ze)   deg
                                         121.99    K  |Px|    Pa
 sameas  0  Gas type                    -145.97    L Ph(Px)   deg
 ideal      Solid type                 -1.6759E-03 M HeatIn    W
!--------------------------------  5 ---------------------------------
 RPNTARGET  use to match speaker current output to measured
 sameas  1c  a                = 5A?     2.2615E-02 A
 4I
```

The above provides a means to tie the value of any DeltaE *input* parameter to the tabulated data. Enforcing values that normally appear as DeltaE *outputs* is a little more subtle. This requires a `RPNTArget` or `FREETarget` to be set so that the output value can be forced to match the input value placeholder (parameter “a” in the `RPNTArget`). This target must be added to the list of target vectors, and of course, an appropriate parameter must be added to the guess vector for balance. A `sameas` reference links the `BLKDATA` parameter (`1c`) into the target location of segment 5.

When any of the input parameters within a BLKDAta segment is selected as the independent variable of a plot loop, there is no additional dialogue to perform; each numeric line of the block data file will be read in order. Each row of the file is read into the input parameters of the segment, one row per data point. DeltaE will run as many solutions as there are data points in the file.

The SYSEXec segment allows DeltaE to execute an external program that can take its inputs from DeltaE and can in turn generate data that will be read in by DeltaE. The first action taken by SYSEXec is to write all of its 14 input parameters to a text file called sysin.dat in single column format. It then calls the the external program using the entire command string specified in the title field of the segment. This program may or may not make use of the numbers in sysin.dat. DeltaE will then read up to 14 output parameters from the file sysout.dat, if it exists. If data needs to be read back in to DeltaE, the program executed should generate these values in the sysin.dat file in the same ASCII, single column format. If more than 14 values are required, more can be read in through a BLKDAta segment.

The following is an trivial example showing how a set of integers assigned to DeltaE input parameters can be sorted by the MS-DOS sort command. The output of the sort command is directed into a file using the '>' character.

```
! Sort input parameters into output parameters (DOS version)
!--------------------------------  2 ---------------------------------
 SYSEXEC    sort sysin.dat > sysout.dat
     3.0000  a In  1                      1.0000  A Out  1
     4.0000  b In  2                      2.0000  B Out  2
     6.0000  c In  3                      2.0000  C Out  3
     7.0000  d In  4                      3.0000  D Out  4
     2.0000  e In  5                      4.0000  E Out  5
    87.000   f In  6                      4.0000  F Out  6
     2.0000  g In  7                      5.0000  G Out  7
     1.0000  h In  8                      6.0000  H Out  8
    10.000   i In  9                      7.0000  I Out  9
    13.000   j In 10                      9.0000  J Out 10
     5.0000  k In 11                     10.000   K Out 11
     4.0000  l In 12                     11.000   L Out 12
    11.000   m In 13                     13.000   M Out 13
     9.0000  n In 14                     87.000   N Out 14
```

After a (r)un, the file sysin.dat will contain the numbers on the left; the sysout.dat will resemble the column on the right.

The number of outputs displayed in a SYSEXec display is determined by the number of lines in the sysout.dat file.

B.13. ALPHABETICAL LISTING AND CROSS-REFERENCE

BEGIN: **(VI B.6)** Initializes $p_{1,}$ U_1, and T_m at the beginning, and sets global f, p_m.

`BLKDATA`: **(VI B.12)** Allows DeltaE to sequentially read rows of a spreadsheet-like text file, through the plotting function.

`BRANCH`: **(VI B.3)** A side-branch with frequency-independent complex impedance.

`COMPLIANCE`: **(VI B.2)** A lumped acoustic compliance (with surface losses).

`CONDUCT`: **(VI B.6)** Sets computation in subsequent segments to conduction mode.

`CONE`: **(VI B.1)** A cone with viscous and thermal dissipation.

`CONSTANTS`: **(VI B.7)** Allows constants and plot independent variables to be used in math segments.

`COPRTARGET`: **(VI B.7)** Allows targeting of ratio of refrigerator COP to Carnot's COP.

`DECOMPOSE`: **(VI B.10)** Decomposes wave into forward and backward traveling components.

`DIFFTARGET`: **(VI B.7)** Allows targeting of difference of two results.

`DUCT`: **(VI B.1)** A duct with viscous and thermal dissipation.

`ENDCAP`: **(VI B.2)** A surface area with $|p_1|^2\,\delta_\kappa$ loss.

`EFFRTARGET`: **(VI B.7)** Allows targeting of ratio of engine efficiency to Carnot's efficiency.

`FREETARGET`: **(VI B.7)** Allows use of non-default target.

`HARDEND`: **(VI B.6)** Default inverse-impedance targets, for hard model termination.

`HBRANCH`: **(VI B.9)** An energy-conserving `BRANCH` for multi-stage refrigerators.

`HX`: **(VI B.4)** A parallel-plate heat exchanger.

`HXFRST`: **(VI B.4)** A parallel-plate heat exchanger before a stack.

`HXLAST`: **(VI B.4)** A parallel-plate heat exchanger after a stack.

`HXMIDL`: **(VI B.4)** A parallel-plate heat exchanger between two stacks.

`HUNION`: **(VI B.9)** An energy-summing, temperature-matching `UNION`.

`IDUCER`: **(VI B.3)** A current-driven transducer attached as a side branch (and independent of frequency).

`IEDUCER`: **(VI B.3)** An enclosed (i.e. series) current-driven transducer (and independent of frequency).

`IESPEAKER`: **(VI B.3)** An enclosed (i.e. series) current-driven electrodynamic transducer.

`IMPEDANCE`: **(VI B.2)** A lumped-parameter series acoustic impedance.

`INSCONE`: **(VI B.1)** An insulated cone, with viscous and thermal dissipation.

`INSDUCT`: **(VI B.1)** An insulated duct, with viscous and thermal dissipation.

`ISOCONE`: **(VI B.1)** An isothermal cone, with viscous and thermal dissipation.

`ISODUCT`: **(VI B.1)** An isothermal duct, with viscous and thermal dissipation.

`INSULATE`: **(VI B.6)** Sets computation in subsequent segments to insulated mode.

`ISPEAKER`: **(VI B.3)** A current-driven electrodynamic transducer, attached as a side branch.

`JOIN`: **(VI B.8)** Joining condition between isothermal and adiabatic segments.

`MEANFLOW`: **(VI B.6)** Enables nonzero mean flow superimposed on the acoustics.

`OPNBRANCH`: **(VI B.3)** A side-branch impedance with frequency dependence of 4π open radiation impedance.

`PISTBRANCH`: **(VI B.3)** A side-branch impedance with frequency dependence of baffled piston.

`PXFRST`: **(VI B.4)** A power-law heat exchanger before a stack.

`PXMIDL`: **(VI B.4)** A power-law heat exchanger between two stacks.

`PXLAST`: **(VI B.4)** A power-law heat exchanger afer a stack.

`PRODTARGET`: **(VI B.7)** Allows targeting of product of two results.

`PX`: **(VI B.4)** A tubular heat exchanger.

`PXFRST`: **(VI B.4)** A tubular heat exchanger before a stack.

`PXLAST`: **(VI B.4)** A tubular heat exchanger after a stack.

`PXMIDL`: **(VI B.4)** A tubular heat exchanger between two stacks.

`RPNTARGET`: **(VI B.7)** Versatile math segment that uses a Reverse Polish Notation containing constants, addresses, and operators.

`QUOTARGET`: **(VI B.7)** Allows targeting of quotient of two results.

`SOFTEND`: **(VI B.6)** Default impedance targets, for mirror-image model termination; also for connection of sidebranch to `UNION`.

`STKCIRCLE`: **(VI B.5)** Thermoacoustic stack (or regenerator) comprised of array of circular pores.

`STKCONE`: **(VI B.5)** Thermoacoustic element comprised of a single, conical pore.

`STKDUCT`: **(VI B.5)** Thermoacoustic element comprised of a single, straight pore.

`STKPINS`: **(VI B.5)** Thermoacoustic stack (or regenerator) comprised of array of pins.

`STKPOWERLAW`: **(VI B.5)** Regenerator with friction factor and heat transfer as power laws in Reynolds number.

`STKRECT`: **(VI B.5)** Thermoacoustic stack (or regenerator) comprised of array of rectangular pores.

`STKSCREEN`: **(VI B.5)** Regenerator comprised of stacked screens.

`STKSLAB`: **(VI B.5)** Slab-geometry stack or regenerator, comprised of parallel plates.

`SX`: **(VI B.4)** A screen heat exchanger.

`SXFRST`: **(VI B.4)** A screen heat exchanger before a stack.

`SXMIDL`: **(VI B.4)** A screen heat exchanger after a stack.

`SXMIDL`: **(VI B.4)** A screen heat exchanger between two stacks.

`SYSEXEC`: **(VI B.12)** Outputs data to an external program, runs that program, and can then read in data generated by that program.

`TBRANCH`: **(VI B.9)** The beginning of a side-branch series of segments.

`TEE`: **(VI B.9)** A temporary segment that inserts a complete file into the model. The file's `BEGIN` segment becomes a `TBRANCH`.

`THERMOPHYSICAL`: **(VI B.11)** Displays properties of gas and solid at the local temperature.

`TITLE`: **(VI B.6)** Comment field required at start of every file.

`TX`: **(VI B.4)** A tubular heat exchanger.

`TXFRST`: **(VI B.4)** A tubular heat exchanger before a stack.

`TXLAST`: **(VI B.4)** A tubular heat exchanger after a stack.

`TXMIDL`: **(VI B.4)** A tubular heat exchanger between two stacks.

`UNION`: **(VI B.9)** Matches p_1 and adds U_1 at union between end of side branch and trunk.

`VDUCER`: **(VI B.3)** A voltage-driven transducer attached as a side branch (and independent of frequency).

`VEDUCER`: **(VI B.3)** An enclosed (i.e. series) voltage-driven transducer (and independent of frequency).

`VESPEAKER`: **(VI B.3)** An enclosed (i.e. series) voltage-driven electrodynamic transducer.

`VOLMTARGET`: **(VI B.7)** Allows targeting of total volume of a series of segments.

`VSPEAKER`: **(VI B.3)** A current-driven electrodynamic transducer, attached as a side branch.

C. Fluids

We provide an artificial temperature floor of 10 Kelvin to prevent DeltaE from trying to use negative temperatures when it is really lost. Consequently, no temperature below 10 Kelvin can be used. In any case, most of the equations for the fluids are inaccurate when this limit is reached. This floor can be modified within the `(T)olerances/debugging` menu.

In what follows, `ta` is temperature in Kelvin, `t1` is temperature in Celsius.

Unless otherwise specified, properties are computed using fits to the data compiled in Touloukian's TPRC series.

DeltaE looks for a 10-character field to determine fluid type. Be sure to use plenty of trailing spaces after short fluid names like "air" to get comments like "gas-type" out of the field.

C.1. helium

Ideal gas approximation for equation of state (including sound speed and expansion coefficient) and specific heat. Transport from Touloukian:

```
k0=0.0025672*ta**0.716
mu=0.412e-6*ta**0.68014
```

C.2. #.###hear (helium-argon mixtures)

Number in the fluid name is helium fraction. Ideal gas approximation for equation of state and specific heat. Transport from Touloukian.[1]

```
k0he=0.0025672*ta**0.716
amuhe=0.412e-6*ta**0.68014
k0ar=(1.39e-4*ta**0.852-1.5e-8*(ta-300.)*(ta-300.))*(1.+2.e-8*pm)
amuar=(1.77e-7*ta**0.852-25.e-12*(ta-300.)*(ta-300.))*(1.+2.e-8*pm)
k0=x1*k0ar+x2*k0he-(k0ar+k0he)*x1*x2**1.5
mu=x1*amuar+x2*amuhe+0.2*(amuar+amuhe)*x1*x2
```

[1] As we learn more about the importance of oscillating thermal diffusion in the thermoacoustics of gas mixtures, we will probably have to revise DeltaE's gas-mixture algorithms dramatically. In the meantime, we simply use the expressions presented here.

C.3. #.###hexe (helium-xenon mixtures)

Number in the fluid name is helium fraction. Ideal gas approximation for equation of state and specific heat. Our fits to Touloukian's transport data are only accurate for `frxe` < 0.5 or for `frxe` $= 1.000$:[1]

```
k0he=0.0025672*ta**0.716
amuhe=0.412e-6*ta**0.68014
k0xe=4.75e-5*ta**0.84*(1.+1.e-7*pm)
amuxe=0.187e-6*ta**0.85*(1.+25.e-9*pm)
frxe=1.-fhe
k0=k0he*fhe+k0xe*frxe-2.*(k0he+k0xe)*frxe*fhe*fhe
mu=amuhe*fhe+amuxe*frxe+(amuhe+amuxe)*frxe*fhe*fhe*(0.8+3.7*fhe*fhe*(0.25-f
rxe))
```

C.4. neon

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.001149*ta**0.65907
mu=0.735e-6*ta**0.66065
```

C.5. air

Ideal gas approximation for equation of state and specific heat. Transport from Pierce, Acoustics:

```
parameter (tps=110.4,tpa=245.4,tpb=27.6,tp0=300.,tpexp=223.8306)
k0=2.624e-2*(ta/tp0)**1.5*(tp0+tpexp)/(ta+tpa*exp (-tpb/ta))
mu=1.846e-5*(ta/tp0)**1.5*(tp0+tps)/(ta+tps)
```

C.6. #.######w (humid air, wet air)

The number in the fluid name is the water mole fraction in a mixture of air and water. As with other gas mixtures, the water mole fraction can be modified or used as an independent plot variable (see "Variable gas mixtures" in Chapter V for details). Even more than with the other gas mixtures, we worry that future improvements to understanding of oscillating thermal diffusion will bring dramatic change to this part of DeltaE. Nevertheless, for the present:

In DeltaE, we think of humid and wet air as a sort of single fluid, having two or three interpenetrating components: dry air, water vapor, and sometimes liquid water.

The molar volume v and molar enthalpy h are DeltaE's primary dependent variables, as functions of three independent variables: mean pressure p_m, mean temperature T_m, and mole fraction of water x_{wat}. To determine whether a given mixture is wet or merely humid, DeltaE compares x_{wat} with p_{sat}/p_m, where $p_{sat}(T_m)$ is the saturated vapor pressure at temperature T_m.

DeltaE treats humid air like the other supported gas mixtures, using the ideal-gas equation of state for v and a slightly nonlinear temperature dependence for h (due to the T dependence of c_p for water vapor in the ASHRAE tables). Also in accordance with ASHRAE recommendations, we set μ and K equal to their dry-air values.

In wet air, in accordance with Bob Hiller's measurements ("Condensation in a steady-flow thermoacoustic refrigerator," J. Acoust. Soc. Am. **108**, 1521–1527 (2000)), the *oscillating* thermodynamics is that of humid air at saturation, while the *meanflow* thermodynamics includes the enthalpy of the liquid when segment `MEANFLOW` is enabled. In other words, if there is no mean flow, the calculation proceeds exactly as for an ideal-gas mixture and ignores the condensate, but if mean flow is nonzero DeltaE performs stack integrations with ideal-gas-mixture properties in the momentum and continuity equations but with the heat of condensation/evaporation (plus the small enthalpy of the condensate itself) included in the energy equation as it is integrated to find dT_m/dx. In all cases, we ignore any dynamic effects of oscillating diffusion of the water vapor through the air.

Although the enthalpy of wet air calculated in DeltaE's thermo algorithm includes the latent heat of freezing and melting as the condensate passes through 0°C, we have not yet incorporated this latent heat, of the liquid-to-solid phase transition, into the numerical integrations in `STK` segments, so integrating through $T_m = 0$°C in wet air is of dubious value. (Anyway, we have not thought of any good reason to use wet air in DeltaE below 0°C, because we believe the stack would simply plug up with ice.)

The saturated vapor pressure is accurate from −100°C to 370°C; other properties are reasonably accurate from −50°C to 150°C.

C.7. nitrogen

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
    k0=0.0003609*ta**0.7512
    mu=0.3577e-6*ta**0.6885
```

C.8. hydrogen

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.002627*ta**0.744
mu=0.19361e-6*ta**0.6723
```

C.9. deuterium

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.002795*ta**0.686
mu=0.2726e-6*ta**0.6721
```

C.10. co2 (carbon dioxide)

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k10=2.8646E-5*ta**1.1318
k20=3.692E-5*ta**1.0940
k0=k10+(pm-1.01e6)/(1.01e6)*(k20-k10)
u10=1.4187E-7*ta**.8216
u20=1.5416E-7*ta**.8094
mu=u10+(pm-1.01e6)/(1.01e6)*(u20-u10)
```

C.11. #.###nexe (neon-xenon mixtures)

Ideal gas approximation for equation of state and specific heat. Transport from ??: (Thermal conductivity not very accurate for high xenon concentrations.)[2]

```
k0he=0.001149*ta**0.65907
amuhe=0.735e-6*ta**0.66065
k0xe=4.75e-5*ta**0.84*(1.+1.e-7*pm)
amuxe=0.187e-6*ta**0.85*(1.+25.e-9*pm)
frxe=1.-fhe(ns)
k0=k0he*fhe(ns)+k0xe*frxe-1.3*(k0he+k0xe)*frxe*fhe(ns)**2.5
mu=amuhe*fhe(ns)+amuxe*frxe+0.12*(amuhe+amuxe)*frxe*fhe(ns)**4
```

[2] As we learn more about the importance of oscillating thermal diffusion in the thermoacoustics of gas mixtures, we will probably have to revise DeltaE's gas-mixture algorithms dramatically. In the meantime, we simply use the expressions presented here.

C.12. NGcbProd (natural-gas combustion products)

Natural Gas combustion products with 5% excess air. Use around 1 atm only. Data supplied by British Gas, with references to Pritchard. Molar weight is a gaussian curve fit taken from Pritchard's data between 288 K and 4000 K.[2]

```
       gamma=1.4
       cp=gasprop(ta,1392.02d0,39.3769d0,-3.89819d0,-0.0317961d0,
     &    0.0327554d0,-1.44149d-3)
       if (ta.gt.2000.) then
         mass=27.9495-7.81175*dexp(-((ta-4151.85)/1047.42)**2)
       else
         mass=27.84
       endif
       r=8314.
       a=dsqrt(gamma*r*ta/mass)
       rho=pm*mass/(r*ta)
       beta=1./ta
       k0=gasprop(ta,0.0997279d0,0.0125516d0,6.73728d-5,4.22761d-4,
     &    1.43198d-4,1.35508d-5)
       mu=gasprop(ta,50.2973d0,4.68523d0,-0.12061d0,0.0140082d0,
     &    -0.001488951d0,4.97968d-5)*1.d-6
       goto 900
       real*8 function gasprop(ta,a,b,c,d,e,f)
       real*8 z,ta,a,b,c,d,e,f
       z=(ta-1400)/200
       gasprop=a+z*(b+ z*(c + z*(d +z*(e+f*z))))
       return
       end
```

C.13. sodium

Data for sodium from Foust, Sodium-NaK Engineering Handbook.

```
a0=2578.
at1=-.52
ap=6.1e-7
r0=950.1
rt1=-2.2976e-1
rt2=-1.46e-5
rt3=5.638e-9
c0=1.4361e3
ct1=-5.8024e-1
ct2=4.6208e-4
k0=.918e2-4.9e-2*t1
if(t1.le.500.) then
 e1=.697
 e2=1.235e-5
else
 e1=1.04
 e2=8.51e-6
endif
a=a0+at1*t1
rho=r0+rt1*t1+rt2*t1**2+rt3*t1**3
beta=(-rt1-2.*rt2*t1-3.*rt3*t1**2)/rho
bt=beta**2-(2.*rt2+6.*rt3*t1)/rho
```

```
cp=c0+ct1*t1+ct2*t1**2
rp=1./a/a+ta*beta**2/cp
bp=-beta/(rho*a**2)+2.*at1/(rho*a**3)-beta**2/rho/cp
bp=bp-2.*ta*beta*bt/rho/cp-ta*beta**3/rho/cp
bp=bp+ta*beta**2*(ct1+2.*ct2*t1)/rho/cp/cp
cpp=-ta*(beta**2+bt)/rho
c So far, everything is evaluated at p=0.
a=a+ap*pm
rho=rho+rp*pm
beta=beta+bp*pm
cp=cp+cpp*pm
gamma=1.+ta*beta**2*a**2/cp
mu=e2*rho**(1./3.)*exp (e1*rho/ta)
```

C.14. nak-78

This is for eutectic NaK-78. Data for sodium-potassium from Foust, Sodium-NaK Engineering Handbook.

```
a0=2051.
at1=-.53
ap=0.
r0=876.4
rt1=-2.183e-1
rt2=-2.982e-5
rt3=0.
c0=970.69
ct1=-.36903
ct2=3.4309e-4
k0=21.4+2.07e-2*t1-2.2e-5*t1**2
if(t1.le.400.) then
 e1=.688
 e2=1.16e-5
else
 e1=.979
 e2=8.2e-6
endif
a=a0+at1*t1
rho=r0+rt1*t1+rt2*t1**2+rt3*t1**3
beta=(-rt1-2.*rt2*t1-3.*rt3*t1**2)/rho
bt=beta**2-(2.*rt2+6.*rt3*t1)/rho
cp=c0+ct1*t1+ct2*t1**2
rp=1./a/a+ta*beta**2/cp
bp=-beta/(rho*a**2)+2.*at1/(rho*a**3)-beta**2/rho/cp
bp=bp-2.*ta*beta*bt/rho/cp-ta*beta**3/rho/cp
bp=bp+ta*beta**2*(ct1+2.*ct2*t1)/rho/cp/cp
cpp=-ta*(beta**2+bt)/rho
c So far, everything is evaluated at p=0.
a=a+ap*pm
rho=rho+rp*pm
beta=beta+bp*pm
cp=cp+cpp*pm
gamma=1.+ta*beta**2*a**2/cp
mu=e2*rho**(1./3.)*exp (e1*rho/ta)
```

C.15. External—provided by user's file.

Files can have any name valid under the operating system under which DeltaE is running, and should end with the extension `.tpf`. If the root filename is the same as any pre-defined fluids, DeltaE will replace its internal calculations for that fluid with those given in the user file. To request a user-defined fluid, simply use the root file name as you would any other fluid. The `.tpf` file should be in the same directory or folder as the model file. The name of the fluid is set to the root filename of the external fluid file. Up to five distinct external fluids can be used at one time.

Each property is specified by a line containing 1–10 real coefficients to be read in as C_{0-9}, where unused parameters are set to zero. The order of the property lines is ρ, c_p, K, a^2, and μ. Comment lines can be added with an initial '!', and blank lines are ignored.

Each of the five properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1 \frac{p_m}{T + p_m C_2} + C_3 T + C_4 T^2 + C_5 T^{C_6} + C_7 p_m^2 T^{C_8} + p_m C_9, \qquad \text{(VI.70)}$$

where T and p_m are the absolute temperature (K) and mean pressure (Pa) for each point at which a segment using the fluid is evaluated.

D. Solids

We provide an artificial temperature floor of 10 Kelvin to prevent DeltaE from trying to use negative temperatures when it is really lost. Consequently no temperature below 10 Kelvin can be used.

In what follows, `ta` is temperature in Kelvin, `t1` is temperature in Celsius.

DeltaE looks for a 10-character field to determine solid type. Be sure to use plenty of trailing spaces after short solid names like "`mylar`" to get comments like "`solid-type`" out of the field.

D.1. ideal

`ks, rhos`, and `cs` are effectively infinite, so $\epsilon_s = 0$.

D.2. copper

```
ks=398.-.0567*(ta-300.)
```

```
rhos=9000.
cs=420.
```

D.3. nickel

```
if (ta.lt.631) then
 ks=63.8+.08066*(631.-ta)
else
 ks=63.8+.02156*(ta-631.)
endif
rhos=8700.
cs=530.
```

D.4. stainless (stainless steel)

```
rhos=8274.55 -1055.23 *dexp(-((T1-2171.05)/2058.08)**2)
ks=(266800*ta**(-5.2)+0.21416*ta**(-1.6))**(-0.25)
cs=(1.7054e-6*ta**(-0.88962)+23324/ta**6)**(-1/3) + 15/ta
```

Prior to version 3.5b2, DeltaE's stainless steel properties were very inaccurate at cryogenic temperatures.

D.5. molybdenum

```
rhos= 10868.6 -2637.52 * exp (-((T1-11383.7)/9701.36)**2)
cs= 253.791 +0.0583812 *T1-2.73919e-06*T1**2
ks= (33.9616 -0.00947953 *T1-4.12809e-08*T1**2)*4.186
```

D.6. tungsten

```
cs=.13576e3*(1.-4805./ta**2)+.0091159*ta+2.31341e-9*ta**3
ks=135.5+1.05e4/ta-.023*ta
rhos=19254*(1.-3.*(-8.69e-5+3.83e-6*t1+7.92e-10*t1**2))
```

D.7. kapton

```
ks=0.2*(1.-exp(-ta/100.))
rhos=1445.-0.085*ta
cs=3.64*ta
```

D.8. mylar

```
ks=0.11+1.7e-4*ta
rhos=1400.-0.175*ta
cs=3.7*ta
```

D.9. External-provided by user's file.

External solids, like external fluids, are derived from coefficients in user-written text files. Up to five external solids can be used at once. Each property is specified by a line containing 1–10 real coefficients to be read in as C_{0-9}, where unused parameters are set to zero. The order of the property lines is ρ_s, c_s, and K_s. Comment lines can be added with an initial '!', and blank lines are ignored.

Each of the three properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1 \exp(-TC_2) + C_3T + C_4T^2 + C_5T^{C_6} + C_7p_m^2T^{C_8} + p_mC_9. \qquad \text{(VI.71)}$$

To request a user-defined solid, simply use the root file name as you would any other solid. The `.tpf` file should be in the same directory or folder as the model file. If the name matches any pre-defined solid name, the (constant) user-defined properties will replace DeltaE's internal calculations. External solids are similar in most respects to external fluids; see Section V E for more relevant information.

E. Menu Options

We list DeltaE's menu options in the order in which they appear.

`(r)un model` instructs DeltaE to begin its computation, adjusting the elements of the guess vector until either all targets are met or an error condition is reached. If one or two plot-independent variables are set, DeltaE will step through them.

`(w)rite current model state` saves the current state in a `.out` file. If the file already exists, you will be given the option of overwriting or renaming it.

`(n)ew model input file` brings a new `.in` (or `.out` file from the disk. If changes have been made in the current model, the user will be prompted to save it first.

`(R)estore vectors.` Use this option to restore all the parameters that were changed to their starting point after an unsuccesful iteration, then modify some value(s) and try again. Do not use this option if the vector table has since been edited.

If you do not respond '`y`'es to the prompt about vector restoration and you have one or both plot loops enabled, you will be give an additional option:

```
Restore to state before last (B)egin or (r)un (y|n)? n
```

```
Restore from a recently plotted point? y
```

DELTAE will now proceed to display the `.plt` file one line at a time. After each line this prompt appears:

```
Return to this state (y|n|Q)? y
```

Typing '`y`' at this point causes the independent plot variable(s) and all members of the guess vector to be returned to those values displayed in the file. Typing '`n`' (or simply <CR>) causes the next line to be displayed. '`Q`' skips to the end of the file and makes no changes. No outputs are changed when this option is executed, so the model must be `(r)un` again to update them; however, be sure to disable the outer plot loop first if you want only one point. Alternatively, you can change the step or endpoints of the plot loop and start plotting again.

This option only works on the current (open) plot file, and it is not useful until after a run which has produced plot points.

`(E)xtras` This option enters a submenu containing less commonly used features (described below).

`(d)isplay` shows information on the screen. It prompts the user to select the `.dat` file (option d), the `.plt` file (option p), the entire `out` file (option o), or a single segment in `.out`-file format (option `n`, the segment number). Typing 'N+' will display the `.out` file from segment N until the end, and typing 'S' will display an abbreviated list of the segment titles. On PC or Unix platforms, these screens have an automatic pause feature after 23 lines are typed—press <CR> to continue, or 'q' to quit the display. There is no 'back up' yet.

`(o)utput to printer` is the same as "display" above, but for a printed copy instead of screen display. We have made no effort to maintain this in recent pc operating systems, so don't be surpried if it doesn't work in your pc. In many Windows systems, (o)utput to printer now makes a file FORT009 which you can print by opening it with notepad. Alternatively, just use a text editor in your operating system to open and print the desired files.

`(f)orm feed printer` makes the printer finish the page and spit it out after doing an "output to printer."

`(t)hermophysical properties` is used to look at properties of any gas, liquid, or solid supported by DELTAE. The user is prompted to select material, temperature, pressure, etc., with current values as defaults, selectable with a carriage return. Data are displayed on the screen in this format:

```
FLUID: 0.880hexe, 302.0 K, 20.000 bar
gamma  a(m/s) rho(kg/m3) cp(J/kg/K) beta(1/K) k0(W/m/K) Prandtl  mu(kg/s/m)
1.67   465.91 15.356     1078.2     0.331E-02 .10586    0.260447 2.5572E-05
Frequency= 0.16 Hz,  delta_nu= 1.8250E-03 m,  delta_kappa= 3.5760E-03 m
Print this? (y/n):
```

`(e)xit DeltaE` returns us to the computer's operating system, prompting for several choices of saving the current model state.

`(p)lot another parameter` adds another parameter to the plot. When a number and *capital* letter is selected, its variable is added to the list of dependent plot variables. When a number and *lower case* letter is selected, its variable is used as an independent plot variable, and the user is prompted to choose it as either inner or outer loop variable, and to give its beginning, ending, and increment (or decrement) values.

`(P)lot status summary` simply displays the current plot status on the screen.

`(c)lear from vectors and plots` is used to eliminate variables from the guess vector, the target vector, the plot dependent-variable list, or the plot independent-variable list. Remember to use lower-case letters when selecting guesses,targets, or plot independent variables, and capital letters for plot dependent variables.

`(C)lear|set all guesses&targets` clears *everything* from the guess and target vectors, if they contain anything. This is most useful in the early stages of model development: If DeltaE doesn't converge, return to your initial guesses (they may be way out of line by now), clear everything, run it, and examine the results to see if one particular segment is giving ridiculous results due to a typographical error in the `.in` file. If the model has empty vectors after a previous (C)lear, selecting this option again causes DeltaE to generate a set of default iteration vectors appropriate to the model.

`(u)se in guess/target vector` allows the user to add a new variable to the guess or target vector, using a number to define ther segment number and a lower-case letter to define the variable. (Target pairs in `HARDEND`, `SOFTEND` can be set in pairs by using the letter 'z.')

`(v)ector status summary` shows the current members of the guess and target vectors on the screen.

`(m)odify parameter value` followed by a segment number and lower-case line letter allows the user to change the value of a guess or input variable. Three special Uppercase pseudo-parameters are also recognized. Selecting *n*`G` or *n*`S` permits the Gas type or Solid type, respectively, of segment *n* to be changed by selecting interactively from a list of all defined types (including currently active user-defined properties). The `T` character brings up the segment's title string.

When editing lines of information (the segment title or the instruction string of an `RPNTArget`, several special editing characters are recognized. \match\replace replaces the first occurrence of *match* with the text in *replace* (replacement strings are always offset by spaces). ^*text* prepends *text* to the line; $*text* appends it.

`(s)pecial modes editing` allows parameter linking modes to be set for any address (segment number and parameter letter) that accepts them. Special modes allow geometric relationships to be maintained when parameters are changed by the solver, by the user, or as an independent plot loop variable.

`(D)OS command shell` temporarily suspends DeltaE and executes a new DOS command environment. This is intended to let the user examine files that are part of other models (not available under `(d)isplay`), to run plotting software on recent results, etc., without having to save all changes and leave DeltaE. To return to the program, type `exit`.

`(K)ill segment.` This option simply removes a segment from your model. It works on any type of segment (except `BEGIN`), and it does nothing intelligent with any lengths that are removed. The user must compensate another length where appropriate.

`(I)nsert segment.` DeltaE will prompt you for the correct number of parameters, giving the parameter name and units. This function is not perfectly interactive. If you make errors in typing in new parameter values, you will be left with a segment that is partly the same as the previous occupant of this spot. You may be able to recover by using the `(m)odify value` option in the main menu for numerical parameters. In the worst case (a bad segment type, for example), you may have to `(K)ill` the new segment and start over again. `(I)nsert` before #segments+1 is permitted to add a segment at the very end.

`(h)ighlight parameter.` Use this to select variables of special interest that you want to be displayed on the screen at the end of every `(r)un`. Add to this list with `(h)ighlight`, remove with `(c)lear`.

E.1. (E)xtra options

The following less-used menu options are accessible after entering `E` to enter the `(E)xtras` submenu:

`(S)plit segment.` This option automates the laborious process of splitting a duct segment (or anything else that has a length) into two segments, each with half the original length, correcting the `sameas` and math segment references, and correcting the iteration, optimization, and plot vectors. (All math segments, vectors, or `sameas` references to the segment specified are incremented by one; that is, the number of the original segment is incremented by one, and the 'clone' segment is effectively inserted before it.)

`(G)enerate state variable plot` performs a single run to generate output of position, area, temperature, pressure, and enthalpy throughout the model. The `.spl` file that is written contains `Nint/2`+1 (see Tolerances/debugging, below) for each integrated segment in the model (ducts are also integrated in this mode). Non-integrated segments dsiplay one line at the beginning and one at the end processing.

`(g)eometry file` causes X-Y points representing a simple sketch of the model to be written to a file ending in `.geo`. Plotted using most any graphing software, these points will give a visual feel for the shape of the design. See the Section V G for an example plot and discussion.

`(F)lip model.` This will take every segment between the `BEGIN` and the last `HARDEND` or `SOFTEnd` and reverse their order. Segments within `TBRAN`ches are left in their original order, however. `sameas`, math segment and plot references are all adjusted and an attempt is made to reform the guess and target vectors. Each `HSXFRst` segment becomes an `HXLAst`, and *vice versa*.

`(T)olerances/debugging` DeltaE has numerous internal parameters that can be altered by the experienced user to control the amount of diagnostic information printed or the behavior of DeltaE's solver. The dialog that appears for this command looks like this, if we keep all the default values by hitting `<CR>`:

```
Nprint <= 0, save only converged endpoint to .dat file.
Nprint > 0, save and display every Nprint intermediate iterations; also
If Nprint < 0, the iteration vector line is omitted.
Nprint = -1?

If PlotDat >= 0, all error messages are announced.
Otherwise, they are only written to the .dat file.
If PlotDat >= 1, all converged endpoints are written to the .dat file.
(for PlotDat = 0, only the most recent)
PlotDat = 0?

Convergence tolerance (1.e-2 > tol > 1.5e-9 recommended):
Tolerance = .300E-03
New value (<CR> to keep)=?

Number of Runge-Kutta steps (should be even:)
Nint = 10?

Normalization mode: 1=standard; 2=special
mode = 1?

Solver step bound factor (.01-100 recommended):
Bound = 100.
New value (<CR> to keep)=?

(Larger values of FCNerr can speed iterations, with a
slightly less accurate endpoint.  Too small a value
can cause the solver to loose its way completely.)
Solver assumed function error (>5.e-15):
FCNerr = .100E-09
New value (<CR> to keep)=?

Minimum Temperature (K):
Tmin=10.0
New value (<CR> to keep)=?

Display exergy in .dat file? 0=no 1=yes
Exergy display = 1?

Environment temperature for exergy calculation:
Envionement temperature (K) = 300.
New value (<CR> to keep)=?

Some plot/spreadsheet packages prefer delimited
columns when reading in plot or state variable files.
Plot field delimiter: 0=fixed space; 1=comma
plot_f_sep = 0?
```

For further details of the effect of each of these parameters, refer to the discussion in Chapter V.

`(e)xit to Main` Return to the main menu. Typing `<CR>` alone has the same effect.

F. Troubleshooting, Common Problems, and Suggested Techniques

The most common problem is failure of a brand-new model to converge, and the most common causes are order-of-magnitude typos in the input file and a premature attempt to run DeltaE with too many variables in the guess and target vectors. The easiest way to fix such a problem is to `(C)lear` everything from the guess and target vectors, run DeltaE, and display the `.dat` or `.out` file. Often it is obvious where your typo is—one of the output variables will go wild in a supposedly innocuous segment. If not, examine the results more closely for reasonableness. Modify suspicious variables a little, to see what effect they have on results. Try to get the model close to converging on your desired targets just by modifying your desired guesses one at a time, manually. Then add one guess-target pair at a time, running DeltaE each time, examining the results, and manually modifying your other desired guess variables to try to keep your other desired target variables under control. For further diagnostic information, try using the `Nprint` variable, found under the `(T)olerances/debugging` menu described in the previous section and in Chapter V.

It is also useful to keep the model as simple as possible. Examples:

- Rely on nonzero U in `BEGIN` instead of a transducer segment if possible.
- Don't try to model a thermoacoustically-driven thermoacoustic refrigerator from scratch without first succeeding with a thermoacoustic driver and then a piston-driven thermoacoustic refrigerator.
- Understand the acoustics of a complicated resonator before adding the stack and heat-exchanger segments.
- In really stubborn cases, start with only one segment; add segments one at a time, inspecting results carefully.

Another cause of failure to converge is poor choice of guess-vector members. Obviously, cross-sectional area of all the segments in a system has little effect on resonance frequency, but a large effect on thermoacoustic power; similarly, it has little effect on $\Im(1/z)$ but a large effect on $\Re(1/z)$, so don't try using area as a guess to achieve a target $\Im(1/z)$. Clearing vector members and running DeltaE, manually modifying potential guess-vector members individually to see if they have significant effects on potential target-vector members, is often educational.

The solver within DeltaE can sometimes become stuck around a local minimum, particularly if you are making incremental changes from a model that has already converged—and often, the internal representation of the 'best guess' does not agree with what we would like for a given model. Try manually changing one of the guess vector members slightly and see if DeltaE will loose its fondness for this particular point. Or, change one of the members of the guess vector, if you can think of an appropriate alternate.

If these steps fail, consider some of the options in (T)olerances/debugging. Some pathologically difficult cases converge better with tighter tolerance, alternate normalization mode, or one of the other tuning options described at the end of Chapter V.

Always check results carefully for reasonableness, particularly when calculating complicated models or using any of DeltaE's more elaborate features. While DeltaE is a useful tool, it is far from foolproof; for example, the shooting method can easily end up generating devices that are several wavelengths long, if initial convergence is slow. All `INS`-type segments, `TBRANches` and `UNIONs` containing thermoacoustic elements also deserve special skepticism.

G. Error and Informational Messages

Most of DeltaE's diagnostics are meant to be self-explanatory, but some require additional information. In the following subsections, we offer some additional hints for the more obscure ones.

G.1. Convergence errors

These errors occur while DeltaE's solver is iterating during a `(r)un`:

`This is not going well...DeltaE gives up!` Associated with this error will be a message "info=4," and a listing of all the current guesses followed by all current target−result values. The solver is not able to find an iteration direction that gives improved results. During a plot loop, this error sometimes occurs multiple times, after which the solver once again finds its way. If it persists, a revision to the solution or target vector may be needed, or the starting point (or plot range) may need to be shifted significantly. Examine the `.dat` file for clues, and think carefully about what is occurring. Simplify the model or iteration if possible. If the error occurs on a model that you know to have good convergence under other conditions, you may be reaching a pathological point. You may be able to jump start it by manually (somewhat intelligently) changing the value of one or two members of the guess vector to put the solver on the right track, or, you may find it very stubborn at this point. Consider revising the guess and/or target vectors, or, `(C)lear` all vectors and targets and examine the outputs to see if you can find clues as to the difficulties. Often, the desired targets may not be reachable, given the constraints you have specified. In all cases, if you have spent some effort reaching this point, `(w)rite` the model to save your work because a floating point error that could cause a crash may occur soon.

`Iteration is complete but some results may not be near their targets.` If the text associated with this message is "info=1," and a listing of all the current guesses is followed by all current target−result values, this is a WARNING message, and not strictly an error. It may occur quite frequently. This message is produced by a secondary convergence check that is necessitated by the solver's inclination to be 'satisfied' with agreement that may not meet the users standards. The check is inadequate; it simply asks if the mean square error of targets−results is at least 100 times less than `tolerance` (see Sec. V J). Based on the relative magnitude of the target values, this threshold may be inappropriate. When this error message occurs during a plot sweep, the line written to the `.plt` file will be preceded by an '∗' to indicate that it requires closer examination. The most common cause of this message is inadequate agreement between results and targets at a `HARDEnd` or `SOFTEnd`. Often, the message will occur for values that are only slightly off. For a model that has this problem, a good way to judge the quality of the results is to add the residual acoustic power or energy flow (parameter `G` or `H`) to the plot list. If the residual flow is orders of magnitude less than the maximum acoustic power flow, the accuracy can usually be accepted. You may also consider setting the normalization mode to 2 (see Sec. IV H) to increase the significance of the endpoint errors.

Handling of convergence errors with plots

When a convergence error occurs during a plotting operation, a '∗' is prepended to the plot line for that point, followed by a single digit representing the "info" variable at that point. A typical line in the `.plt` file might look like this:

```
*4   2.990      98.61      528.2      45.36      559.3      3219.
```

Below is a key to interpreting the "info" codes:

`info=`	Significance
1	The solver considers the iteration successful, but the residual error is suspiciously large.
2	The solver was making progress, but the maximum number of iterations has been reached. Another `(r)un` might make further progress.
4	The solver was unable to progress toward convergence.

There are some cases when one may want to delete the prepended asterisk and number. For example, if the code is "info=2," and the remaining error that the solver was working on was already quite small, it might be quite valid to use this plot point instead of discarding it.

G.2. Input

The following messages can occur when DeltaE is reading in a model description file:

i `numerical parameters expected and only` *j* `were found in` *segtype* `segment, Segment number` *n*. `Edit model file and restart. Last input was:`
This message indicates that an input parameter could not be converted into a floating point value. The value may contain stray, inappropriate characters, or one or more lines may be missing and DeltaE may be trying to read the fluid name as the numerical parameter it needs. For a math segment, be sure you have specified the initial `Target` value first, even if you do not intend to use it.

`Unknown segment type:` *segtype*. The string at the beginning of the segment description does not match any segments in the library. Be sure that at least the first five characters are UPPERcase. The error could also be stray lines; for example, specifying the solid type twice.

`Illegal fluid:` *fluid string*. This message occurs when DeltaE cannot find the requested fluid in the internal library or as a `fluid.tpf` file in the current directory. Check the spelling of the fluid and be sure that there are enough spaces to fill a 10-character field before any other text occurs. If you are using an external fluid, be sure the file is present in the same directory. If all this appears correct, you may have one line too many of numerical parameters (the giveaway here will be the contents of *fluid string*).

`Unknown plate material:` *plate string*. The comments regarding the `Illegal fluid` message, above, also apply to the plate (solid) specification; however, the default `ideal` solid type may also be specified by a blank line. If this is the intent, be sure that a blank line truly separates each segment module.

`Error reading segment/parameter address in` *segtype* `segment, Segment number` *n*.
This error occurs while reading in one of the math segments (see Sec. V B.7) or when processing a `sameas` reference. The characters read do not decode to a valid address in the model.

`More than 5 external fluids found....`

`More than 5 external solids found....`
Only five distinct types of user-defined fluids or plates are allowed in a model at one time.

`Guess/Result vectors are too long.  Reduce count before proceeding.` The maximum problem order for this version of DeltaE is 18.

`Too many plot parameters are selected.  Reduce count before proceeding.` Up to 13 parameters for plotting can be specified, and the first N of these, where N is the guess vector length, will be selected automatically; these cannot be cleared. You must clear one of the user-selected parameters. In addition, one or two independent variables are also part of the 'plot' file.

`Nested TEE files are not permitted...compile one at a time.` An input file named in a `TEE` statement in turn contains another `TEE` statement. This is not supported. Running DeltaE on the input file first will generate a combined file that can then be included as a branch.

G.3. Model editing

The messages below occur when a model is being modified online:

`*** sameas relationship cancelled...` The parameter you are affecting, by using it in the guess vector, making it an independent plot variable, or `(m)odify`ing it, is not specified directly, but through a `sameas` statement. This connection is severed, and the parameter takes on the value it currently has, until you (or DeltaE) give it another.

`*** Special mode affecting this value must be disabled first.` This parameter is linked back to another parameter that may change, and thereby, modify this value. Such a link is not appropriate if you are trying to set the value independently, or if DeltaE will try to do so while it is plotting or iterating; therefore, you will get the message above when you are try to modify it or make it a guess or an independent plot variable. `(d)isplay` this segment to find the root of this link that must be cleared. It is indicated in () to the right of the parameter description.

`This variable must be cleared from the guess vector first.` A guess vector member cannot be the target of a link (it *may* be the root), or an independent plot variable, nor may it contain a `sameas` statement.

`This parameter is part of a plot loop.  It cannot participate in the guess vector.` Using this parameter as a guess would alter the independent variable of the plot loop as the solver iterates.

`This output is not in any vector.` An attempt was made to `(c)lear` a parameter that has not been `(u)sed` or `(p)lotted` in the target or plot vectors.

`WARNING: could not find appropriate default targets.  Modify iteration vectors before solving this model.` While trying to generate a set of guess and target vectors, DeltaE could not find anything suitable. Be sure all `HXFRST`, `STK*`s, and `HXLAST`s (or `HXMID1`s) are in proper sequence, and if this is not the problem, DeltaE is not smart enough to help in this case: set your vectors manually.

G.4. Consistency checks

These errors are detected when DeltaE begins processing during a `(r)un`:

`FATAL Error: First segment must be BEGIN.` A `BEGIN` segment is required as the first segment for any model you intend to `(r)un`; without it, DeltaE has no values for the initial conditions.

`SAMEAS parameter types do not match SAMEAS error: Seg#` *n*, `Parameter` *p*. Except for math segments (see Sec. V B.7), all parameters specified by `sameas` must have a parameter description that matches the root values description through the first four characters. Hence, parameter `a` in a duct may come from `areaI` or `areaF` of a cone, but not from its `length`.

`Circular reference found processing SAMEAS Circular SAMEAS: Seg#` *n*. This parameter is not rooted in an actual value. It is specified by a `sameas` that, either directly or through additional references, refers back to the same address.

`WARNING: you have` *i* `guess vector members and` *j* `target vector members defined. You must either add` *k* `new target parameters or delete` *k* `guesses.`
The guess and target vectors have different lengths. You must take some action to balance them.

`Adjustable length segment cannot refer to itself.` The length of this segment (parameter `c`) is either linked to itself or to another segment's length that, either directly or through additional segments, is linked back to this segment.

H. Known Bugs and Limitations

- DeltaE's internal solver is very efficient at converging to solutions for complicated systems; however, it knows nothing about acoustics, or any part of physics, for that matter. If it ventures too far, it can give you more wavelengths than you intended before reaching resonance. DeltaE does not know that negative frequencies, negative pressures, or negative lengths are improper; it simply does the math. In short, the reasonableness of the answers produced will almost always depend on the quality of the initial guesses.

- Numbers stored in `.out` files are stored with much less precision than DeltaE actually maintains internally. Sometimes, after storing a file in a particularly difficult computation, the solver will converge a little differently if the file is loaded back in and run again.
- Not all floating point errors are successfully trapped on all systems; some can cause the program to crash and lose unsaved work. Save your work frequently if you are exploring uncharted territory!

I. Registration

While there is no formal registration for this program, no fees, and no support or warranty of any kind (please read the copyright notice), we are interested in maintaining a list of users so that we can fix any bugs that are found and notify known users of serious errors. If you use this program, please send your name, address, and any comments to Bill Ward, by letter, fax, or electronic mail, at the addresses below. If you find any bugs to report, we would be especially appreciative:

```
Bill Ward
Los Alamos National Laboratory
Group ESA-AET
MS C914
Los Alamos, NM 87545

Fax:    505-665-7176
E-mail:  ww@lanl.gov
```

News of your successes using this code will encourage our sponsors to consider this effort worthwhile and will enable us to respond to user's questions. Please tell us how this code has been helpful to you. We are grateful for your acknowledgments in publications and reports and for mention of this work to individuals at agencies that support acoustics research. This will improve our chance to create and pass on improvements in the future.

J. Obtaining DeltaE

DeltaE is under continual development and regular users should update their copies frequently. The latest version is always available (for non-commerical and evaluation use) from the LANL World Wide Web server `www.lanl.gov/thermoacoustics/`. Users contemplating commercial use of the software should contact ww@lanl.gov (Bill Ward) for an update on the current policy.

K. Acknowledgments

The development of DeltaE has been supported in part by many agencies and entities: Tektronix Corporation, SPAWAR, the Naval Postgraduate School, ONR, and, most importantly, by several branches of the Department of Energy: Advanced Industrial Concepts, Materials Science in Basic Energy Sciences, our local Industrial Partnership Office, the Technology Transfer Initiative Office, and the Office of Fossil Energy. A long discussion with Pat Arnott helped us define the initial scope of this work, and comparisons with the results of parallel-plate-stack codes (written by Al Migliori and Dick Martin) were useful in the early stages. Suggestions by Kim Godshalk, Charles Jin, Tom Hofler, and Jeff Olson have led to significant improvements in DeltaE's capability and usability. Charles Jin, Ray Radebaugh, and the code `REGEN3.1` were indispensable in development of the stacked-screen algorithm.

Bibliography

[1] G. W. Swift. Thermoacoustic engines. *J. Acoust. Soc. Am.*, 84:1145–1180, 1988.

[2] G. W. Swift. *Thermoacoustics: A Unifying Perspective for some Engines and Refrigerators.* Acoustical Society of America Publications, Sewickley PA, 2002.

[3] G. W. Swift. *Encyclopedia of Applied Physics*, volume 21, chapter Thermoacoustic engines and refrigerators, pages 245–264. Wiley, for American Institute of Physics, 1997.

[4] G. W. Swift. Analysis and performance of a large thermoacoustic engine. *J. Acoust. Soc. Am.*, 92:1551–1563, 1992.

[5] J. R. Olson and G. W. Swift. Similitude in thermoacoustics. *J. Acoust. Soc. Am.*, 95:1405–1412, 1994.

[6] T. J. Hofler. *Thermoacoustic refrigerator design and performance.* PhD thesis, Physics department, University of California, San Diego, 1986.

[7] T. J. Hofler. Concepts for thermoacoustic refrigeration and a practical device. In Chairman Paul Lindquist, editor, *Proceedings of the 5th International Cryocoolers Conference*, pages 93–101, Wright-Patterson AFB, OH, August 1988. Wright-Patterson Air Force Base.

[8] I. Urieli and D. M. Berchowitz. *Stirling Cycle Engine Analysis.* Adam Hilger, Bristol UK, 1984.

[9] A. J. Organ. *Thermodynamics and Gas Dynamics of the Stirling Cycle Machine.* Cambridge University Press, 1992.

[10] G. W. Swift and W. C. Ward. Simple harmonic analysis of regenerators. *J. Thermophysics and Heat Transfer*, 10:652–662, 1996.

[11] W. M. Kays and A. L. London. *Compact Heat Exchangers.* McGraw-Hill, New York, 1964.

[12] M. A. Lewis, T. Kuriyama, F. Kuriyama, and R. Radebaugh. Measurement of heat conduction through stacked screens. *Adv. Cryogenic Eng.*, 43:1611–1618, 1998.

[13] J. R. Olson and G. W. Swift. Acoustic streaming in pulse tube refrigerators: Tapered pulse tubes. *Cryogenics*, 37:769–776, 1997.

[14] J. R. Olson and G. W. Swift. Suppression of acoustic streaming in tapered pulse tubes. In R. G. Ross Jr., editor, *Cryocoolers 10*, pages 307–313. Plenum, New York, 1999.

[15] G. W. Swift, M. S. Allen, and J. J. Wollan. Performance of a tapered pulse tube. In R. G. Ross Jr., editor, *Cryocoolers 10*, pages 315–320. Plenum, New York, 1999.

[16] K. M. Godshalk, C. Jin, Y. K. Kwong, E. L. Hershberg, G. W. Swift, and R. Radebaugh. Characterization of 350 Hz thermoacoustic driven orifice pulse tube refrigerator with measurements of the phase of the mass flow and pressure. *Adv. Cryogenic Eng.*, 41:1411–1418, 1996.

[17] R. Yaron, S. Shokralla, J. Yuan, P. E. Bradley, and R. Radebaugh. Etched foil regenerator. *Adv. Cryogenic Eng.*, 41:1339–1346, 1996.

[18] M. Iguchi, M. Ohmi, and K. Maegawa. Analysis of free oscillating flow in a U-shaped tube. *Bull. JSME*, 25:1398–1405, 1982.

[19] W. P. Arnott, H. E. Bass, and R. Raspet. General formulation of thermoacoustics for stacks having arbitrarily shaped pore cross sections. *J. Acoust. Soc. Am.*, 90:3228–3237, 1991.

[20] G. W. Swift and R. M. Keolian. Thermoacoustics in pin-array stacks. *J. Acoust. Soc. Am.*, 94:941–943, 1993.

[21] R. S. Reid. *Open cycle thermoacoustics.* PhD thesis, Georgia Institute of Technology, School of Mechanical Engineering, 1999.

[22] R. S. Reid and G. W. Swift. Experiments with a flow-through thermoacoustic refrigerator. *J. Acoust. Soc. Am.*, 108:2835–2842, 2000.