DESCRIPTION OF THE PUFL FORTRAN PROGRAM

Leota Barr
Eleanor Reed
Contents

Abstract ................................................. 1
Introduction ........................................ 1
PUFL Input Parameters ............................ 6
   Format of PUFL Input .......................... 6
   Structure of Data Deck ........................ 6
   PUFL Control Words ............................. 7
Starting, Stopping, and Dumping Controls ...... 7
PUFL Input Which Defines Regions of the Pipe . 8
Time Controls ....................................... 10
Artificial Viscosity Constants .................. 10
Cutoff Controls ..................................... 10
Overburden Controls ............................... 11
Special Density Calculations ..................... 11
Conditional Controls Which Hold Pressure Inside Defined Regions .... 12
Rezoning and Modifying Zones .................... 12
Input for Energy Conservation Studies .......... 13
Heat Transfer, Mass Addition, and Friction Input .......... 13
Equation-of-State Input .......................... 15
High Explosive (HE) Pressure Controls ......... 17
Boundary Condition Input ........................ 17
Cavity Zones Added at Boundary 1 as the Cavity Expands ...... 21
Editing Controls .................................... 21
Special Printouts .................................. 22
Plot Controls ........................................ 22
Time vs F(t) Plots at Fixed X-locations Along Pipe .... 22
Pressure vs Time Tracings ....................... 23
Movie Plots .......................................... 23
Description of PUFL Subroutines ............... 23
Changing the PUFL Code .......................... 35
Sample PUFL Problems ............................ 36
   With Minimum Input Data .................... 36
   With Extensive Input Data ................... 37
Acknowledgments .................................... 37
References .......................................... 37

-iili-
DESCRIPTION OF THE PUFL FORTRAN PROGRAM

Abstract

The PUFL code (Fortran program) performs a gasdynamic calculation for a pipe flow with entrainment. PUFL is used at LRL in the Plowshare program. PUFL consists of a large number of nested subroutines which help to separate the code into natural, distinct units of logical control or calculation. The numerous subroutines exist because usually each addition to PUFL was attached as a subroutine to the running program. PUFL has over 100 input variables all or part of which may be defined by the user.

Introduction

PUFL\(^1\) is a Fortran program that performs an "almost-Lagrangian" formulation in quasi-one-dimensional geometry for the conservation equations of gasdynamics that consider zones with mass sources and sinks. This report describes the PUFL code, its general logical structure, all of its input parameters, and its subroutines. Special aids to understanding the program are also given.

- The PUFL flow chart (Fig. 1) shows the general logical structure of the code. This structure is common to most large finite-difference calculation codes.
- The PUFL subroutine level structure (Fig. 2) shows the linkage among the subroutines.
- Tables 1 and 2 are indexes of input parameters and subroutines, respectively.
- Two sample problems are given at the end of the report, one with a minimum of input and the other containing a larger number of input parameters.

PUFL consists of an unusually large number of subroutines because, as the program developed, new additions were first organized into separate subroutines, which then were linked to the running program. The program was altered this way for two reasons; the changes could be made quickly, and, the many subroutines helped to compartmentalize the code with each set of related calculations in a separate subroutine.

PUFL has been an experimental code, not a fixed production code; therefore, extensive programming changes were regularly made to match the calculations with experiments in the field. The machine software does as much of the routine calculations and file manipulations as possible so that program modifications can be made quickly with the number of human errors kept to a minimum. It has always been most important to obtain meaningful results from PUFL as soon as possible. Many of the LRLTRAN new Fortran statements have been used and have proved valuable. (LRLTRAN is a Fortran-type language developed and used at LRL.)
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Page</th>
<th>Variable name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA</td>
<td>14</td>
<td>NO</td>
<td>16</td>
</tr>
<tr>
<td>BC1, BC2</td>
<td>17</td>
<td>NOEDIT</td>
<td>21</td>
</tr>
<tr>
<td>BCIX, BC2X, BCIP, RC2P</td>
<td>17</td>
<td>NSTOP</td>
<td>8</td>
</tr>
<tr>
<td>BETADT</td>
<td>14</td>
<td>OVERBD</td>
<td>11</td>
</tr>
<tr>
<td>C1, COSQ, COSQ</td>
<td>10</td>
<td>P</td>
<td>9</td>
</tr>
<tr>
<td>CITAU, C2TAU, UCTAU</td>
<td>17</td>
<td>PHI</td>
<td>14</td>
</tr>
<tr>
<td>CALPHI</td>
<td>14</td>
<td>PHOLD</td>
<td>12</td>
</tr>
<tr>
<td>CFRICT, CF</td>
<td>14</td>
<td>PICKSW</td>
<td>23</td>
</tr>
<tr>
<td>CMU</td>
<td>11</td>
<td>PLMT</td>
<td>22</td>
</tr>
<tr>
<td>DEBUG</td>
<td>8</td>
<td>PRESN</td>
<td>13</td>
</tr>
<tr>
<td>DELTAT</td>
<td>10</td>
<td>R</td>
<td>19</td>
</tr>
<tr>
<td>DOTMC</td>
<td>13</td>
<td>RADC</td>
<td>18</td>
</tr>
<tr>
<td>DTMIN</td>
<td>10</td>
<td>RHO</td>
<td>8</td>
</tr>
<tr>
<td>DTPICK</td>
<td>23</td>
<td>RC</td>
<td>21</td>
</tr>
<tr>
<td>DTU, DTSSP, DT, DTM, DTQ</td>
<td>10</td>
<td>RDR</td>
<td>19</td>
</tr>
<tr>
<td>DUMP</td>
<td>8</td>
<td>REWIND</td>
<td>8</td>
</tr>
<tr>
<td>DUMPLT</td>
<td>8</td>
<td>RMIN</td>
<td>11</td>
</tr>
<tr>
<td>DX</td>
<td>8</td>
<td>ROJPN</td>
<td>13</td>
</tr>
<tr>
<td>E</td>
<td>9</td>
<td>RP</td>
<td>18</td>
</tr>
<tr>
<td>EC</td>
<td>21</td>
<td>RPR</td>
<td>19</td>
</tr>
<tr>
<td>EDDT</td>
<td>21</td>
<td>RTAPE</td>
<td>7</td>
</tr>
<tr>
<td>EDDUMP</td>
<td>21</td>
<td>SDSP</td>
<td>17</td>
</tr>
<tr>
<td>EDITSW</td>
<td>22</td>
<td>SIGMA</td>
<td>14</td>
</tr>
<tr>
<td>EDITT</td>
<td>21</td>
<td>SKEDIT</td>
<td>21</td>
</tr>
<tr>
<td>EFORM</td>
<td>14</td>
<td>SKPLOT</td>
<td>22</td>
</tr>
<tr>
<td>END</td>
<td>7</td>
<td>STAN</td>
<td>74</td>
</tr>
<tr>
<td>EOS</td>
<td>16</td>
<td>SWQ1, SWQ2</td>
<td>13</td>
</tr>
<tr>
<td>ETA</td>
<td>13</td>
<td>SWQCH</td>
<td>14</td>
</tr>
<tr>
<td>EVAPOR</td>
<td>16</td>
<td>TC</td>
<td>21</td>
</tr>
<tr>
<td>F</td>
<td>17</td>
<td>THOLD</td>
<td>12</td>
</tr>
<tr>
<td>G</td>
<td>11</td>
<td>TITLE</td>
<td>16</td>
</tr>
<tr>
<td>GAMMAC</td>
<td>21</td>
<td>TLAG</td>
<td>20</td>
</tr>
<tr>
<td>GAMMAJ</td>
<td>10</td>
<td>TLAGD</td>
<td>12</td>
</tr>
<tr>
<td>GJ</td>
<td>8</td>
<td>TOTAL</td>
<td>10</td>
</tr>
<tr>
<td>HOFV</td>
<td>13</td>
<td>TSTOP</td>
<td>10</td>
</tr>
<tr>
<td>HOFW</td>
<td>13</td>
<td>U</td>
<td>9</td>
</tr>
<tr>
<td>JABL1, JABL2</td>
<td>13</td>
<td>UFORM</td>
<td>14</td>
</tr>
<tr>
<td>JACTIV</td>
<td>11</td>
<td>UJ1</td>
<td>18</td>
</tr>
<tr>
<td>JDR1, JDR2</td>
<td>19</td>
<td>UJ0NM</td>
<td>13</td>
</tr>
<tr>
<td>JEND1, JEND2</td>
<td>20,19</td>
<td>UMIN</td>
<td>10</td>
</tr>
<tr>
<td>JEOS</td>
<td>15</td>
<td>X</td>
<td>19</td>
</tr>
<tr>
<td>JEOSC</td>
<td>21</td>
<td>XABL</td>
<td>20</td>
</tr>
<tr>
<td>JL, JH</td>
<td>22</td>
<td>XABL1, XABL2</td>
<td>14</td>
</tr>
<tr>
<td>JHOLD</td>
<td>12</td>
<td>XADD</td>
<td>21</td>
</tr>
<tr>
<td>JIN, JOUT</td>
<td>12</td>
<td>XC</td>
<td>21</td>
</tr>
<tr>
<td>JLAST</td>
<td>8</td>
<td>XEDDT</td>
<td>22</td>
</tr>
<tr>
<td>JMU1, JMU2</td>
<td>11</td>
<td>XEDITT</td>
<td>22</td>
</tr>
<tr>
<td>JOVER</td>
<td>11</td>
<td>XJ1</td>
<td>10</td>
</tr>
<tr>
<td>JXS</td>
<td>22</td>
<td>XJON</td>
<td>13</td>
</tr>
<tr>
<td>MAT</td>
<td>15</td>
<td>XMIN, XMAX</td>
<td>22</td>
</tr>
<tr>
<td>MLAST</td>
<td>15</td>
<td>XMSSTAR</td>
<td>13</td>
</tr>
<tr>
<td>MOVIE</td>
<td>23</td>
<td>XOVER</td>
<td>11</td>
</tr>
<tr>
<td>MTMU</td>
<td>17</td>
<td>XPK</td>
<td>23</td>
</tr>
<tr>
<td>MTPR</td>
<td>17</td>
<td>XPICK</td>
<td>22</td>
</tr>
<tr>
<td>MTRID</td>
<td>17</td>
<td>XRIGID</td>
<td>18</td>
</tr>
<tr>
<td>NDELD</td>
<td>8</td>
<td>XSURF</td>
<td>18</td>
</tr>
<tr>
<td>NDTCON</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2. Index of PUFL subroutines.

<table>
<thead>
<tr>
<th>Subroutine name</th>
<th>Page</th>
<th>Subroutine name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTEST</td>
<td>31</td>
<td>P JKCP</td>
<td>33</td>
</tr>
<tr>
<td>ADMAS</td>
<td>29</td>
<td>PLOT P</td>
<td>33</td>
</tr>
<tr>
<td>CHECKT</td>
<td>31</td>
<td>POSTAP</td>
<td>33</td>
</tr>
<tr>
<td>COM</td>
<td>23</td>
<td>PRESET</td>
<td>33</td>
</tr>
<tr>
<td>CXADD</td>
<td>31</td>
<td>PRES</td>
<td>24</td>
</tr>
<tr>
<td>DATALD</td>
<td>31</td>
<td>PUFL</td>
<td>23</td>
</tr>
<tr>
<td>DETXAB</td>
<td>31</td>
<td>PUFLOK</td>
<td>33</td>
</tr>
<tr>
<td>DTCAL</td>
<td>32</td>
<td>QABL</td>
<td>33</td>
</tr>
<tr>
<td>DUMPE</td>
<td>31</td>
<td>QCAL</td>
<td>30</td>
</tr>
<tr>
<td>EDITOR</td>
<td>32</td>
<td>RADIUS</td>
<td>34</td>
</tr>
<tr>
<td>EPDVCL</td>
<td>32</td>
<td>REZONE</td>
<td>34</td>
</tr>
<tr>
<td>FMDOT</td>
<td>30</td>
<td>STOPSW</td>
<td>34</td>
</tr>
<tr>
<td>GCAL</td>
<td>32</td>
<td>TCAL</td>
<td>34</td>
</tr>
<tr>
<td>HECAL</td>
<td>33</td>
<td>TECHK</td>
<td>34</td>
</tr>
<tr>
<td>LAGRAN</td>
<td>25</td>
<td>UPTAB</td>
<td>34</td>
</tr>
<tr>
<td>OFFMON</td>
<td>33</td>
<td>XXPICK</td>
<td>34</td>
</tr>
<tr>
<td>ONMON</td>
<td>33</td>
<td>XXPLOT</td>
<td>34</td>
</tr>
</tbody>
</table>

In this report, the main PUFL equations are given in abbreviated forms when they help to describe the program. The equations are defined in detail in Ref. 1.

Figure 1 briefly shows the linkage between the initial loading of data, the presetting of data prior to cycle 0, the main calculations, the edit, dump, and exit packages.

Figure 2 shows PUFL has five levels of subroutines, most of which are at level three. A subroutine at level n conditionally or unconditionally calls a subroutine next in sequence at level n or n + 1, depending upon the logical flow of the code or the options the user has chosen. After all operations in a subroutine have been executed control is returned to its calling subroutine.

For example (referring to Fig. 2), MAIN, at level 1, is entered first. It calls DATALD at level 2 which calls UPTAB at level 3. (DATALD and UPTAB are called unconditionally.) Next DATALD calls STORMU only if the user has put equation-of-state tables in the input data. Then DATALD calls ONMON only if the problem is continuing. (Its cycle number is greater than 0.) This type of logical flow applies throughout PUFL.

When LAGRAN is reached, all introductory settings are fixed. During the remaining part of the PUFL run, controls repeatedly loop through MAIN, calling LAGRAN through OFFMON until the problem is finished or the problem termination signal is given manually.
Fig. 1. Flowchart of PUFL code.
Fig. 2. Levels of PUFL subroutines.
PUFL Input Parameters

FORMAT OF PUFL INPUT

PUFL uses the Data Directed Input (DDI) loader, written by Alex Cecil, to load data into memory. The loader allows data, written in a free-field format, to be read in from cards. A number can be loaded into the memory location assigned to a variable by the compiler, by placing the variable symbolic name followed by an equal sign, then a mandatory space in front of the number on the card. Similarly, arrays or lists can be loaded into their assigned memory locations. For example,

On data cards:

\begin{verbatim}
JLAST = 2 4 10
XJI = -600.
\end{verbatim}

In memory:

\begin{verbatim}
JLAST(1) = 2
JLAST(2) = 4
JLAST(3) = 10
XJI = -600,
\end{verbatim}

The variable name and its equal sign must not contain any spaces, and must be followed by at least one space. A space serves as a separator and signals the ending of a name, sentinel, or number.

STRUCTURE OF DATA DECK

The PUFL data input deck, placed following the * data card, has this general structure:

<table>
<thead>
<tr>
<th>Column No.</th>
<th>1</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card 1</td>
<td>Title card: title of the problem (all 80 columns)</td>
<td></td>
</tr>
<tr>
<td>Card 2</td>
<td>Tape 5 card: the 5-character vault name of the problem dump tape</td>
<td></td>
</tr>
<tr>
<td>Card 3 etc.</td>
<td>PUFL data: first set—(no limit to number of data cards)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End card (contains the word END)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PUFL data: second set—(no limit to number of data cards)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End card (contains the word END)</td>
<td></td>
</tr>
</tbody>
</table>

-6-
Title Card

The first card in the data deck is the title card. The first 60 characters on this card must remain unchanged during the entire run of the problem because they are used by PUFL to make sure the correct dump tape 5 is used with a given problem card deck. Characters 61 through 72 may be changed before a problem is continued. This way, starting at some advanced time, several variations of a problem may be run, and the printout titles can be varied to record the changes.

Dump Tape Card

The card following the title card contains the names of the dump tapes. Its format is:

Col. 1 and 2 contain *T.
Col. 7 through 11 contain the first dump tape name (must be given).
Col. 13 through 17 contain the second dump tape name (optional).

For example:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
* T A B 3 0 0 A B 3 0 2
```

Tape AB300 is the first dump tape. Tape AB302 is the second dump tape. (Optional, used only if PUFL is expected to use a second dump tape.)

PUFL CONTROL WORDS

END This name serves as a sentinel indicating the end of a data set. Generally, only the word END is placed on a card.

There are two data sets, hence two END cards in the input. But the first time a problem is run only the first set is read.

The first set is read before the dump tape is read, therefore this set can contain information about reading the dump tape. The second set is read after the desired dump record has been loaded into memory. Thus, the second set can be used to change the information from the dump tape in memory before production cycling starts.

STARTING, STOPPING, AND DUMPING CONTROLS

RTAPE If RTAPE > 0, then the last dump on the dump tape is loaded into memory and execution starts from that time. The first time a problem is run, this variable name should not be included in the data deck because initially there are no dumps on the dump tape.
DUMP If DUMP = n, with n > 0 and an integer, then the nth dump on the dump tape is loaded into memory and execution starts from that time.

NSTOP If NSTOP = n, with n > 0, and an integer, then the problem runs until NCYCLE ≥ NSTOP, at which time a transfer is executed to the sense switch 1 coding where the problem is taken off the machine. This control is useful when debugging problems. NCYCLE is the PUFL cycle counter.

DUMPLT If DUMPLT = D is loaded, then after D dumps have been written on the dump tape, it is unloaded and a new one is called for by the code. DUMPLT is preset to 50.

REWIND If REWIND > 0, then, before the main calculations are started, the dump tape is rewound, and a new dump tape is requested by the program. The name of the new tape equals the 5 characters located in positions 13 through 17 on the card following the title card (second in the data deck).

NDELD If NDELD = n is loaded, then every n cycles a dump will be made onto the dump tape. NDELD is preset to 3000 cycles.

TSTOP When problem time ≥ TSTOP, calculations are terminated. TSTOP is preset to 100 sec.

DEBUG If DEBUG > 0, then a disk file is generated to replace the dump tape. This feature can be used for debug runs when only one or two dumps will be written by the code.

PUFL INPUT WHICH DEFINES REGIONS OF THE PIPE

A PUFL problem may divide a pipe into at most 50 regions, each of which needs to have these variables defined.

JLAST Each number in the JLAST list defines the end of a region in the pipe. These numbers are the subscript numbers of the end grid lines of the regions and must be given monotonically. The last non-zero number in the list is assumed to be JMAX, the subscript of the last grid line.

Each region with an entry in JLAST needs to have the following variables defined. The order in which these numbers are listed, following a variable name, corresponds to the listed order of the JLAST numbers.

DX Increment along the pipe. \( \Delta x = x_{j+1} - x_j \)

GJ \( \gamma = \frac{C_P}{C_V} \) is the ratio of specific heats for a gas.

RHO Density, \( \rho \)
Grid Generation

The x-coordinates of the grid lines along the pipe are generated initially using the JLAST and DX numbers. In general, if the data contains:

JLAST = \( J_1, J_2, \ldots, J_{K\text{MAX}} \)

\( DX = \Delta x_1, \Delta x_2, \ldots, \Delta x_{K\text{MAX}} \)

\( X_{J1} = X' \)

Then, first \( X' \rightarrow X_1 \)

Next, with \( k \) subscript defined by the relationship,

\[ J_{k-1} \leq j \leq J_k \]

then for \( j = 1, 2, \ldots, J_{\text{MAX}-1}, \)

\[ X_j + \Delta x_k \rightarrow X_{j+1} \]

In particular, if the pipe is divided into the zones shown in Fig. 3, then the card input data should be:

JLAST = \( 2 \; 3 \; 6 \)

\( DX = 5. \; 2. \; 10. \)

\( X_{J1} = -10. \)

![Fig. 3. Zoning of pipe along the x-coordinate.](image)

The resulting x-grid values stored in memory will be:

\[ x_1 = -10. \; (X_{J1} \rightarrow x_1) \]

\[ x_2 = -5. \; (x_1 + \Delta x_1 \rightarrow x_2) \]

\[ x_3 = 2. \; (x_2 + \Delta x_1 \rightarrow x_3) \]

\[ x_4 = 2. \; (x_3 + \Delta x_2 \rightarrow x_4) \]

\[ x_5 = 12. \; (x_4 + \Delta x_3 \rightarrow x_5) \]

\[ x_6 = 22. \; (x_5 + \Delta x_3 \rightarrow x_6) \]
At this time, the parameters \( \gamma, p, u, P, \) and \( e \), defined for each region, will be stored for each zone. The \( x \)-coordinates may be negative, zero, or positive.

**XJ1**
Defines the initial \( x \) value the code places in \( x_1 \). If not given in the input, then \( 0 \rightarrow x_1 \).

**GAMMAJ**
A constant \( \gamma \). It is used if \( GJ = 0 \).

**TIME CONTROLS**

**TSTOP**
Problem stopping time.

**TOTALT**
Problem starting time. During execution, this word is increased each cycle by \( \Delta t \). TOTALT is preset to 0.

**DELTAT**
\( \Delta t \) value for cycle 0.

**NDTCON**
If \( NDTCON > 0 \), then DELTAT will be held at its original value throughout the run. If \( NDTCON = 0 \), then the code will adjust DELTAT each cycle. NDTCON is preset to 0.

**DTMIN**
The minimum \( \Delta t \) allowed by the code. DTMIN is preset to \( 1 \times 10^{-11} \).

**DTU**
These five variables are constant multipliers of the five minimum \( \Delta t \) calculations made by PUFL. The user may either give values to these multipliers or use their preset values.

<table>
<thead>
<tr>
<th>Multiplier name</th>
<th>Preset value</th>
<th>( \Delta t ) using multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTQ</td>
<td>0.1</td>
<td>( \Delta u ) max. particle velocity</td>
</tr>
<tr>
<td>DTSSP</td>
<td>0.2</td>
<td>( \Delta v ) sound speed</td>
</tr>
<tr>
<td>DTV</td>
<td>0.01</td>
<td>( \Delta v ) volume</td>
</tr>
<tr>
<td>DTM</td>
<td>0.01</td>
<td>( \Delta m ) mass addition</td>
</tr>
<tr>
<td>DTQ</td>
<td>0.01</td>
<td>( \Delta q ) heat transfer</td>
</tr>
</tbody>
</table>

**ARTIFICIAL VISCOSITY CONSTANTS**

**C\( \phi \)SQ**
These numbers are used as weighting factors in the artificial viscosity (or \( \cosQ \)) equation

\[
C_j = \left\{ \frac{1}{\rho} \Delta \left[ C\phi \times \Delta \cdot C \left( \frac{\gamma P}{\rho} \right)^{1/2} \right] \right\}_j.
\]

**CUTOFF CONTROLS**

**UMIN**
Defines the minimum non-zero absolute value for velocity that is allowed. If \( |U_j| < UMIN \), then \( O \rightarrow U_j \).
RMIN Defines the minimum absolute relative change that is allowed in \( \rho \) (density), \( P \) (pressure), and \( e \) (specific internal energy). For example, if 
\[
\frac{|\rho' - \rho|}{\rho} < RMIN,
\]
then \( \rho \rightarrow \rho' \)

JACTIV If JACTIV > 0, then the calculations are made for all of the zones, 1 through JMAX.
If JACTIV = 0, then calculations are made only for the active zones plus an extra 10 zones on each end of the active zones.

OVERBURDEN CONTROLS

OVERBD If OVERBD > 0, then calculations for overburden pressures are made each cycle. 
\( P_{\text{overburden}} = \sum G_{\rho A}{\Delta x} \). Also, when OVERBD > 0, then all \( x_j \) that are greater than OVERBD are dropped from the calculations. If, in addition, XOVER \( \neq 0 \), then a search is made for \( j \) such that 
\[
x_j \leq \text{XOVER} < x_{j+1}, \quad \text{and} \quad j \rightarrow \text{JOVER}. 
\]

XOVER XOVER is the \( x \)-coordinate where overburden calculations are to start. They are made through \( X_{\text{JMAX}} \) (usually the surface coordinate).

JOVER JOVER is the \( j \) subscript of \( x_j \) where the overburden calculations are begun. JOVER is preset to 1. Hence if XOVER is not given, JOVER = 1 will be used.

G Gravity needs to be inserted into the input data when overburden calculations are to be made. Also, at other times, when G is to be included in the velocity equation (980 cm/sec\(^2\) in cgs units).

SPECIAL DENSITY CALCULATIONS

CMU If CMU > 0, then for all \( j \) such that JMU1 \( \leq j \leq JMU2 \) these density and mass equations are used in place of the standard ones.

\[
\rho_j = \pi r_{j+1}^2 \times \frac{CMU}{V_j}
\]

\[
m_j = \rho_j V_j
\]
If \( m_j' < m_j \), then \( m_j \rightarrow m_j' \).
CONDITIONAL CONTROLS WHICH HOLD PRESSURE INSIDE DEFINED REGIONS

PHOLD If JHOLD is such that 0 < JHOLD < JMAX, then each cycle, calculations
are made only for j = 1 through JHOLD until these two conditions are
satisfied.

JHOLD • Condition 1.
P_{JHOLD} ≥ PHOLD

• Condition 2.
A time lapse equal to TLAGD has passed after condition 1 has been
satisfied.

When these two conditions have been met, the code allows the calculations
to be made beyond j = JHOLD to the end of the pipe. JHOLD is preset to
0, hence this coding is turned off unless the user gives these variables
appropriate values.

REZONING AND MODIFYING ZONES

Removing Zones

JOUT JOUT is a list of zones by subscript/number which the user wants to remove
before the main calculations start. If the jth through the (j + k)th zones
are to be removed, then the variables describing the jth through (j + k)th
zones are combined with the variables of the (j - l)st zone. (Added, volume
weighted etc., as necessary to conserve mass, internal energy, and kinetic
energy.) For example: JOUT= 7 8 9 12 would cause zones 7, 8, 9, to
be combined with zone 6, and 12 to be combined with 11. Zones 6 through 9
must belong to the same equation-of-state region and 11 and 12 to the same
one. JOUT may have at most 200 entries.

Inserting Zones

JIN JIN is a list of zones, by number, each of which is to be divided into two or
more zones. If a zone number is listed once, then the zone will be divided
into two equal parts, and if it is listed twice, into three equal parts and in
general, if listed n times, into n + 1 parts.

Example: Let JIN = 2 4*3
then old grid nos: 1 2 3 4 5 6
new grid nos: 1 234 56789 10
JIN may have at most 200 entries.
Modifying Zones

Pressure, density, velocity, x-coordinates for \( j = 1, 2, \ldots, 450 \). These variables can be changed after a problem has run a while by placing their new values in the second data set.

INPUT FOR ENERGY CONSERVATION STUDIES

If XMSTAR = 99., then \( X_{j_{\text{max}}} \rightarrow \text{XMSTAR} \). If XMSTAR > 0, then a fixed grid is generated by dividing \( X_1 \) through XMSTAR into 200 equally spaced zones. Each cycle, \( \Sigma PAV \) is calculated using this fixed grid, and calculated again using the PUFL Lagrangian grid. The two sums are printed offline.

HEAT TRANSFER, MASS ADDITION, AND FRICTION INPUT

ETA is the turbulent transpiration coefficient, HOFW is the specific heat of fusion and HOFV is the specific heat of vaporization. They are used in the equation for the specific heat of ablation, 

\[
Q^3_j = HOFW + HOFV + ETA \left( \frac{Ve}{2} \right)^2_j
\]

\[
\dot{m}_j = (Q_{\text{sum}}/Q^3_j) + \text{DOTMC}
\]

DOTMC is a constant \( \dot{m} \) applied to zones with \( j = \text{JABL1} \) through JABL2.

Mass addition equations are calculated for zones \( j = \text{JABL1} \) through JABL2. JABL1 and JABL2 are preset so that the mass addition calculations are not made. (JABL1 = 1 and JABL2 = \( \phi \) initially.) JABL1, JABL2 may be given values directly in the input, and then they remain fixed throughout the machine run. An alternative way of setting these subscripts is by "turning on" switches SWQ1, SWQ2, XABL1, or XABL2. Each is "turned on" by loading it with a number greater than zero. If SWQ1 (or SWQ2) is on, then the subscript of the maximum \( Q_j \) is used. If XABL1 (or XABL2) is on, then the subscript of the closest x-coordinate is used.

If SWQ1 > 0, then during each cycle the code finds the \( j \) such that \( Q_j \) is the maximum \( Q \) between \( j = 1 \) and JEND1; then \( (j + 1) \rightarrow \text{JABL1} \).

If SWQ2 > 0, then during each cycle the code finds the \( j \) such that \( Q_j \) is the maximum between \( j = \text{JEND1} \) and JMAX; then \( j \rightarrow \text{JABL2} \).
If $XABL1 > 0$, then during each cycle the $j$ is found such that
$x_{j-1} < XABL1 < x_j$; then $j = JABL1$.

If $XABL2 > 0$, then during each cycle the $j$ is found such that
$x_{j-1} < XABL2 < x_j$; then $(j - 1) = JABL2$.

For $j = JABL1$ through $JABL2$, the mass addition, $Q_{\text{radiation}}$, and
convection equations are calculated only if $e_j \geq \text{EFORM}$ and
$|u_j| \geq \text{UFORM}$.

$\text{SIGMA}$ is the Stefan-Boltzman Constant (or a constant times the Stefan-
Boltzman Constant). For $j = JABL1$ through $JABL2$, if $\text{SIGMA} > 0$, then
$Q_j^{\text{radiation}} = \sigma T^4_j / (\text{Opacity} \times \rho_j)$.

The code carries an air opacity table which is $f(\rho, T)$. The user may insert
another opacity table into the code.

$\text{STAN}$ is the local coefficient of heat transfer.

$\text{ALPHA}$ is used in the equation which calculates specific internal energy of
mass entering a zone. It is also used as a switch, which when negative,
causes a different pair of equations to be used for EW and UW.

If $\text{ALPHA} < 0$, then $E W_j = \text{HOFV} + \text{HOFW}$
$U W_j = \phi$.

If $\text{ALPHA} \geq 0$, then $E W_j = \text{ALPHA} X Q_j^e / (1 + \text{PHI})$
$U W_m = (2 \times \text{PHI} \times E W_j)^{1/2}$.

If $\text{SWQCH} = 0$, then
$Q_{\text{convection}} = \left| \text{STAN} \left( \frac{\gamma}{\gamma - 1} \right) P u_j \right|$

If $\text{SWQCH} > 0$, then
$Q_{\text{convection}} = \left| \text{STAN} \rho u \left( \gamma e \frac{u^2}{2} \right)_j \right|.$

$\text{CALPHI}$ is calculated as follows:

$\text{PHI} = .5 (u^2/e)_j$.

If $\text{CALPHI} = 0$, then $\text{PHI}$ is used as loaded in the input. It is used in EW
and UW equations which calculate the specific internal energy of the mass
and velocity of mass entering a zone. (See above equations.)

$\text{CFRICT}$ is used in the shearing stress equation,
(or $\text{CF}$)
$\tau W_j = \text{CFRICT} \rho u^2_j$.

$\text{BETADT}$ is used, if $\text{BETADT} > 0$, then the code will solve for
$\dot{m} = \frac{m}{S}$
$X \text{BETADT}$ and use it as $\text{DOTMC}$. 
EQUATION-OF-STATE INPUT

Equation-of-State Controls for Pressure Calculations

MAT  MAT and MLAST numbers are used to generate the internal EOS numbers called JEOS; (one for each j zone).

MLAST  MAT is a list of material equation-of-state numbers, and MLAST is a corresponding list of last zone numbers. They may have at most 50 entries each. All j zones that have not been given a MAT number by the user are initially preset with

\[ 0 - \text{JEOS}_j \text{ and } e\rho(\gamma - 1) \rightarrow P_j \] (User must give PUFL a constant \( \gamma \) in the input.)

MAT\(_k\) is the EOS number for all zones \( j \) such that \( \text{MLAST}) < j \leq \text{MLAST}_0 \) and for these \( j \)'s, \( \text{MAT}_k = \text{JEOS}_j \). (The code assumes \( \text{MLAST}_0 = 0 \).)

JEOS  As an alternative, EOS numbers may be entered directly into the JEOS\(_j\) list. If used, one JEOS\(_j\) needs to be defined for each j zone starting with \( j = 1 \).

<table>
<thead>
<tr>
<th>MAT(_k) values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>( \gamma_j = f(e_j) ) is used in place of ( \gamma = \text{constant} )</td>
</tr>
<tr>
<td>100</td>
<td>( P_j ) is calculated using HE equations.</td>
</tr>
<tr>
<td>( \geq 50 )</td>
<td>( P_j ) is calculated using equations located in subroutine PRSEQ(J).</td>
</tr>
<tr>
<td>( L = 1, 2, \ldots, 20 )</td>
<td>( P_j ) is obtained from EOS table subscripted by ( L ) with ( 1 \leq L \leq 10 ) or ( 11 \leq L \leq 20 ) depending upon ( e_j ) and EVAPOR values.</td>
</tr>
</tbody>
</table>

EVAPOR is the energy for vaporization limit. The PUFL code will increase or decrease the subscript \( L \) by 10, when necessary, so that these conditions will always be satisfied for each \( j \).

If \( e_j < \text{EVAPOR} \), then \( L \) must be \( 1 \leq L \leq 10 \).

If \( e_j \geq \text{EVAPOR} \), then \( L \) must be \( 11 \leq L \leq 20 \).

0  \( P = e(\gamma - 1)\rho \).

Equation-of-State Data

When the user wants to enter EOS data for one or more materials into the code, the material data needs to be written in a form similar to this example. \( N \) represents a number; \( N_1 \) is a list of numbers. The words enclosed in parentheses are only comments to the reader and are not to be placed on the cards.

-15-
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO = 2</td>
<td></td>
</tr>
<tr>
<td>TITLE = 'solid'</td>
<td>(place title in quote)</td>
</tr>
<tr>
<td>MTPR = N\textsubscript{i}</td>
<td>(pressure entries)</td>
</tr>
<tr>
<td>MTMU = N\textsubscript{i}</td>
<td>(the corresponding ( \mu = \frac{P}{P_0} - 1 ) entries)</td>
</tr>
<tr>
<td>C1TAU = N</td>
<td>C2TAU = N UCTAU = N</td>
</tr>
<tr>
<td>SDSP = N</td>
<td>(Initial density)</td>
</tr>
<tr>
<td>EVAPOR = N</td>
<td>(not needed if NO = 12 is not to be used)</td>
</tr>
<tr>
<td>EOS</td>
<td>(sentinel indicating end of EOS data set)</td>
</tr>
</tbody>
</table>

| NO = 12         |         |
| TITLE = 'gas'   |         |
| MTPR = N\textsubscript{i} |         |
| MTMU = N\textsubscript{i} |         |
| C1TAU = N       | C2TAU = N UCTAU = N | (friction constants) |
| SDSP = N        |         |
| EVAPOR = N      |         |
| EOS             |         |

The cards containing this data need to be placed, as sets, preceding the two END cards. Notice the word, EOS, which signals the end of a data set. Also, NO, which is the number assigned to the material EOS table, \( f(\mu, P) \) by the user.

**Description of the EOS Variables**

**TITLE**: Used to describe and help identify EOS data. The title of the equation-of-state data is placed between quote marks, i.e., 'H2O1/69'. This entry is optional.

**EOS**: Used as an exit sentinel similar to the END sentinel. The sentinel name, EOS, signals the end of an equation-of-state data card set. Each time EOS is detected by the loader, controls return to the PUFL coding where this set of equation-of-state data is stored into the memory location defined by NO. Controls then return to the loader.

**NO**: Used to tag the EOS table internally. When MAT\textsubscript{k} is set equal to NO, then zones between MLAST\textsubscript{k-1} and MLAST\textsubscript{k} will pick up their pressure values from equation-of-state tables number NO.

**EVAPOR**: EVAPOR is the energy-for-vaporation limit of a material. (It is preset to \( 1 \times 10^{-36} \).) The user may define two EOS tables for each material, one for its solid state (NO = N\textsubscript{1} with \( 1 \leq N_1 < 10 \)), and the other for its gaseous
state \((\text{NO} = N_2 \text{ with } 10 \leq N_2 \leq 19 \text{ and } N_2 = N_1 + 10)\)  the energy \(e_j\) of the zone containing this material determines which table is used.

If \(e_j < \text{EVAPOR}\), then table \(N_1\) is used.

If \(e_j \geq \text{EVAPOR}\), then table \(N_2\) is used.

**SDSP**  Sound speed of the material

**MTPR**  MTPR is the pressure list, and

**MTMU**  MTMU is the corresponding \(\mu = \frac{P}{\rho_0}\) list.

**MTRO**  Initial density, \(\rho_0\)

**C1TAU**  See subroutine LAGRAN.

**C2TAU**

**UCTAU**

---

**HIGH EXPLOSIVE (HE) PRESSURE CONTROLS**

\(F_j\) is the fraction of pressure released in zone \(j\) in the HE calculations.

If \(J\text{EOS}_j\) is set equal to 100, then the HE calculations are turned on, and initially \(1 \times 10^{-5} \rightarrow F_j\). \(F_j\) may be loaded with initial values which will automatically override the ones preset by the code.

If \(J\text{EOS}_j \neq 100\), then \(1 \leftarrow F_j\) and the HE calculations are not made.

The equation used to calculate \(F\) each cycle is:

\[
F_j = 3.658 \times (1. - 1.84/\rho_j)
\]

then

\[
P_j \times F_j \rightarrow P_j
\]

(See subroutine HECAL for more details.)

---

**BOUNDARY CONDITION INPUT**

**Boundary Contributions at \(j = 1\) and \(j = J\text{MAX}\)**

**BC1**  The BC1 table gives boundary conditions at \(j = 1\) in terms of velocity,

**BC1P**  pressure or x-position vs time depending upon the setting of two switches called BC1P and BC1X.

<table>
<thead>
<tr>
<th>Switch setting</th>
<th>BC1 table entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC1P = 0 and BC1X = 0</td>
<td>velocity vs time</td>
</tr>
<tr>
<td>BC1P &gt; 0</td>
<td>pressure vs time</td>
</tr>
<tr>
<td>BC1X &gt; 0</td>
<td>x-position vs time</td>
</tr>
</tbody>
</table>
The maximum dimension of the BC1 table is 200, and the order of the \( t \) (time) and \( U \) (velocity, pressure, or x-coordinate) entries in the table is \( U_1 t_1 U_2 t_2 \ldots U_n t_n \) with \( t_1 < t_2 < \ldots < t_n \).

**BC2**

The BC2 table, and BC2P, BC2X switches are the same as above except these apply to the \( j_{\text{max}} \) boundary.

**BC2P**

**UJ1**

Defines the initial velocity for the first \( j = 1 \) boundary of the zone during the first cycle only.

\[ U_{J1} = u_{1} - 1, u_{1}^{+1/2}, u_{1}^{+1} \]

**XRIGID**

If XRIGID > 0, then the code does not allow any \( x_j \) to become less than XJ1. If for some \( j > 1 \), \( x_{j-1} < XJ1 < x_j \), then the code will set \( x_1 \) equal to XJ1 and move \( x_2, x_3, \ldots, x_{j-1} \) to equally spaced positions between XJ1 and \( x_j \).

**XSURF**

XSURF is the upper limit of the \( x \)-coordinate surface (end of the pipe). If XSURF \( \neq 0 \), then zones whose \( x_j \) become greater than XSURF are discarded. XSURF is preset to zero by the code.

### Defining Radii

**RADC**

This entry is the constant radius for each zone. If the RP array is empty, then RADC \( \rightarrow r_j \) for all \( j \).

**RP**

A double subscripted array which gives radius, \( r_{ij} \), as a function of time \( t_{ij} \).

The array is loaded by rows and has this order:

\[
\begin{align*}
\text{RP} & = x_1 \, ws \, r_{11} t_{11} r_{12} t_{12} \ldots r_{1n} t_{1n} + 0. *^{(50-n)} \\
& \quad x_2 \, ws \, r_{21} t_{21} r_{22} t_{22} \ldots r_{2n} t_{2n} + 0. *^{(50-n)} \\
& \quad \vdots \\
& \quad x_m \, ws \, r_{m1} t_{m1} r_{m2} t_{m2} \ldots r_{mn} t_{mn} + 0. *^{(50-n)}
\end{align*}
\]

"ws" is an abbreviation for working storage. Zeros need to be loaded at these places in the table so that the numbers will be spaced correctly in memory. Also, each row needs to be filled to the end with zeros, i.e. \( (0. *^{50-n}) \), so that the next string of input numbers will start loading into the beginning of the next row. The "*" is a repeat sentinel to the loader which causes zeros to be stored into the next (50-n) consecutive locations. The RP array has 50 \( \times \) 50 dimensions.

At time \( t \), for each \( x_k \) entry in the RP array, the code first calculates and stores \( R_k \) in the ws locations as follows:
For each $k$

$L$ is determined by $t_L < t \leq t_{L+1}$,

then

$$R_k = R_L + (R_{L+1} - R_L) \frac{(t - t_L)}{(t_{L+1} - t_L)}.$$

Next, the $R_j$ for each PUFL $x_j$ is calculated as follows:

$k$ is determined by $X_k < X_j \leq X_{k+1}$,

then

$$R_j = R_k + (R_{k+1} - R_k) \frac{(x_j - x_k)}{(x_{k+1} - x_k)}.$$

These tables may be used in place of the above RP table if the radii are to remain constant with time. They are paired, single-subscripted variables with $R_i$ the constant radii at locations $X_i$ for all the problem time. When the PUFL preset coding finds $X_1 > 0$ or $R_i > 0$, it generates an RP table for the main code using the $R, X$ entries. The dimension of $R$ and $X$ is 50 each.

If $JDR2 > 0$, then for $j = JDR1$ through $JDR2$, $RDR - R_j$. $RDR$ is a constant radius independent of time.

The code saves for each $j$ zone between $j = JEND1$ and JQMAX the problem time at which it first attains the maximum $Q$ value along the pipe. These times are saved in an internal list called $TFORR_j$. JQMAX is the number of the zone which has the largest $Q$ during the current cycle.

The $i, L$ subscripts of the $RPR_i, L$ array define its columns and rows respectively in these equations. "i" is the time, and radius subscript, and $L$ is the pressure or x-coordinate subscript.

Also, at this time, the released pressure, $PR$, for the $j$th zone is set equal to the pressure of the $j-2$th zone. $L$ is determined such that $RPR_{1, L-2} \leq PR < RPR_{1, L}$.

Next, a variable called $PFORR_j$ is calculated which relates $PR$ to the pressures in the $RPR$ array and is used later in the subroutine $RADIUS$.

$$PFORR_j = L + (PR - RPR_{1, L-1})/(RPR_{1, L} - RPR_{1, L-1}).$$

The $RADIUS$ subroutine calculates the expansion radii for $j$ zones with $j = JEND1$ through JQMAX if the $RPR$ array has non-zeros entries.
First TRON, (time radius on) an internal variable, is evaluated.

\[ \text{TRON} = \text{TOTALT} - \text{TFORR}_j - \text{TLAG} \]

When TRON > 0, then the expansion radius is calculated for that \( j \) zone.
Subscript \( i \) is found such that \( \text{RPR}_{i-2,1} \leq \text{TRON} < \text{RPR}_{i,1} \) (time subscript).

Next TFRACT is evaluated.

\[ \text{TFRACT} = \frac{(\text{TRON} - \text{RPR}_{i-2,1})}{(\text{RPR}_{i,1} - \text{RPR}_{i-2,1})} \]

If \( \text{RPR}_{1,1} < 0 \), then RPR array is \( f(X, T) \) and \( R_{\text{expansion}} \) is solved by Eqs. (1).

If \( \text{RPR}_{1,1} > 0 \), then RPR array is \( f(P, t) \) and \( R_{\text{expansion}} \) is solved by Eqs. (2).

\[ L \] is found such that \( \text{RPR}_{L-1,1} < x_j \leq \text{RPR}_{1,1} \).

\[
\begin{align*}
\text{R}_2 &= \text{RPR}_{1-3,1} + \text{TFRACT} \times (\text{RPR}_{1-1,1} - \text{RPR}_{1-3,1}) \quad \text{and} \quad x_j < \text{RPR}_{1,1} \leq x_{j+1}, \\
\text{R}_1 &= \text{same equation as R2 except subscript L-1 is used in place of subscript L.} \\
R_{\text{expansion}} &= \text{R}_1 + \frac{(x_j - \text{RPR}_{1,1})}{(\text{RPR}_{1,1} - \text{RPR}_{1-1,1})(\text{R}_2 - \text{R}_1)}. 
\end{align*}
\]

\[
\begin{align*}
\text{R}_1 &= \text{RPR}_{1-3,1} + \text{PFORR}_j + \text{TFRACT} \\
& \times (\text{RPR}_{1-1,1} - \text{RPR}_{1-3,1} - \text{PFORR}_j) \quad \text{and} \quad x_j < \text{PFORR}_j < x_{j+1}, \\
\text{R}_2 &= \text{same equation as R1 except subscript (PFORR}_j + 1) \quad \text{is used in place of PFORR}_j, \\
R_{\text{expansion}} &= \text{R}_1 + \frac{[\text{PFORR}_j - \text{integer (PFORR}_j)](\text{R}_2 - \text{R}_1).}{(\text{RPR}_{1,1} - \text{RPR}_{1-1,1})(\text{R}_2 - \text{R}_1)}. 
\end{align*}
\]

\( JEND_1 \)

\( JEND_1 \) is the number of the zone where radius expansion begins. \( JEND_1 \) is preset to 1.

\( XABL \)

If \( XABL > 0 \), the code finds \( j \) such that \( x_j < XABL < x_{j+1} \), then if \( JEND_1 > j \), \( j \rightarrow JEND_1 \).

\( JEND_2 \)

If \( JEND_2 = 0 \), then a search is made for QMAX starting at \( Q_{j_{\text{max}}} \) at the end of the pipe, and working toward \( Q_1 \). As soon as a \( j \) is found such that \( Q_j > Q_{j-1} \), \( Q_j \) is taken as \( Q_{\text{max}} \) and \( j \rightarrow JQ\text{MAX} \). If \( JEND_2 > 0 \), then radial expansion is turned on from \( JEND_1 \) through \( JMAX \).

\( TLAG \)

\( TLAG \) is the time lag between shock arrival and beginning of radial expansion.
CAVITY ZONES ADDED AT BOUNDARY 1 AS THE CAVITY EXPANDS

XADD  If XADD > 0 or XC > 0, then a test is made each cycle to determine if
XC  x_1 > XADD. If true, a zone is added at boundary 1, and all the j subscripted
GAMMAC  variables are moved up one; that is, j -> j + 1 for all j. The constant
JEOSC  variables defining the added cavity zones are: XC (all new X_1's), GAMMAC
TC  (all new \gamma_1's), and JEOSC (all new JEOS_1's).
RC
EC

If XADD > 0 or XC > 0, then a test is made each cycle to determine if
x > XADD. If true, a zone is added at boundary 1, and all the j subscripted
variables are moved up one; that is, j -> j + 1 for all j. The constant
variables defining the added cavity zones are: XC (all new X_1's), GAMMAC
(all new \gamma_1's), and JEOSC (all new JEOS_1's).

TC  TC is the time list, and corresponding to its entries are RC, a density list,
EC

RC

of time. A linear interpolation on these tables gives e_1(t) and \rho_1(t). The
dimension of these tables is 50.

If BC1P > 0, then P_1 is taken from the BC1 table. Otherwise, P_1 is calculated
using the EOS tables of P_1 = e_1 p_1(\gamma_1 - 1).

If BC1P < 0 and BC1X = 0, u_1 is obtained from the BC1 tables. Otherwise u_2 \to u_1.

EDITING CONTROLS

EDITT  It is adequate to give only one value to EDDT, then that value of \Delta t edit
EDDT  will be used during the entire machine run. If time-varied \Delta t edits are
NOEDIT  desired, then a maximum of five paired entries may be given defining those
\Delta ts.

Their preset values will give the user an edit every 500 \mu sec. That is,
initially EDDT = 500, E - 5 and EDITT = 100.

Either the pair EDDT and EDITT, or the pair NOEDIT and EDITT may
be given to define the editing frequency as a function of time.

EDDT_{i+1} is the \Delta t between edited outputs for problem time such that
EDITT_{i} \leq problem time < EDITT_{i+1} (\Delta t = EDDT_{i} is used during time t
such that 0 \leq t \leq EDITT_{i}).

NOEDIT_{i+1} is the number of edits between times EDITT_{i} and EDITT_{i+1}.
During the time interval t = EDITT_{i} to EDITT_{i+1}, the number of PUFL
edits will equal NOEDIT_{i+1}.

NOEDIT_{i} is used during time t such that 0 \leq t \leq EDITT_{i}.

SKEDIT  If SKEDIT > 0, the HSP edits will be skipped.

EDDUMP  If EDDUMP > 0, then all dump records on the dump tape will be edited and no
PUFL calculations will be made.

-21-
SPECIAL PRINTOUTS

JXS
JXS, are the subscripts of the x's that are to be saved on disk at each edit time along with the problem time and cycle number. At the end of each run, after sense switch 1 is turned on, these saved x values and their times are edited. The maximum number of x's for a given time that can be saved is 5.

JL
JL, and JH, are the lower and upper j limits of four sums (mass, KE, IE, and momentum) which are calculated and saved at edit times. JL and JH may have five or fewer entries each. After sense switch 1 is turned on, these sums along with their times are printed out.

PLMT
PLMT is a list of pressure limits which has a dimension of 6. If PLMT, > 0, then extra printouts and extra plots will be made. All of the P, are tested against the PLMT pressure limits. The first time a P, exceeds a limit, a line is printed out giving

\[ j \times P_j \] pressure limit exceeded.

Also two extra plots are made, \( x \) vs \( u^2 \) and \( x \) vs \( u^2 + P \).

EDITSW
If EDITSW > 0, then the pressure will be printed out in dynes/cm\(^2\) and energy in TON\(^{TNT}\).

PLOT CONTROLS

SKPLOT
If SKPLOT > 0, then the DD80 plots will not be made. SKPLOT is preset equal to zero.

XMIN
If XMIN ≠ 0, then a second set of five plots are made using XMIN and XMAX as the limits on the x-coordinate of the plots.

The eight DD80 plots that are made each edit times are x-coordinate vs u (velocity), P (pressure), \( \rho \) (density), m (mass), e (specific internal energy), \( Q_{\text{convection}} \), \( Q_{\text{radiation}} \) and \( \dot{m} \) (rate of mass addition).

TIME VS F(t) PLOTS AT FIXED X-LOCATIONS ALONG PIPE

XPICK
If XPICK, numbers are submitted in the data by the user, then at the end of the problem, after sense switch 1 is put down, PUFL will make eight time-vs-F(t) plots for each XPICK, submitted by the user, namely t (time) versus \( \rho \) (density), u (velocity), P (pressure), R (radius), e (energy), \( \dot{m} \) (rate of mass addition), \( Q_{\text{convection}} \) and \( Q_{\text{radiation}} \).
During the production run, these variables are saved on tape 7 at times equal to $XEDDT_i$ when $XEDITT_i < \text{problem time} < XEDITT_{i-1}$. At most 300 points per plot will be used. At present, those saved after the first 300 will be disregarded.

$XEDITT$ and $XEDDT$ may each have at most 5 entries, and $XPICK$ may have at most 50 entries.

**PICKSW**

If PICKSW > 0, then the points used to make the DD80 curves will be listed on the HSP after sense switch 1 is put down.

If these time plots are requested by the user, a tape 7 needs to be defined in the production card deck (*T xxxxx=7=xxxxx) with xxxxx the vault number where the tape is stored. Initially tape 7 is a blank tape.

**PRESSURE VS TIME TRACINGS**

**DTPICK**

If DTPICK > 0, then at time intervals of DTPICK, the pressures from at most 20 x-locations enumerated in the $XPK_i$ list, are saved on tape or disk. At the end of a run, special plots, time vs pressure, at each $XPK_i$ location, are made. The first two consecutive zero entries in the $XPK_i$ list imply the end of the list.

**MOVIE PLOTS**

**MOVIE**

If MOVIE > 0, then special DD80 plots are made which can be converted to a movie film. Making a movie usually requires special coding inside the subroutines PUFLOK and MOGRID.

**Description of PUFL Subroutines**

<table>
<thead>
<tr>
<th>Subroutine Name</th>
<th>Function of Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM</td>
<td>Contains the cliche (block of coding) named PUFCOM where the parameter and common statements used by PUFL subroutines are located.</td>
</tr>
<tr>
<td>PUFL</td>
<td>This is the main routine in the PUFL code. Initialization is done here. Also dumps and edits are made at their requested intervals, and the final pre-exit instructions are executed after sense switch 1 has been put down.</td>
</tr>
</tbody>
</table>
PRS Calculates \( P_j \) (pressure) for the \( j \)th zone if \( 0 < \text{JEOS}_j < 100 \). The value of \( \text{JEOS}_j \) determines the EOS that is used, and the \( \text{JEOS}_j \) values are defined by the user by loading a MAT list.

<table>
<thead>
<tr>
<th>( \text{JEOS}_j ) value</th>
<th>( P_j ) calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1 \leq \text{JEOS}_j \leq 20 )</td>
<td>Obtained from ((P, \mu)) table. Up to 10 such tables are stored in the code and any of these may be overwritten by the user with tables placed in the data card deck.</td>
</tr>
<tr>
<td>( \text{JEOS}_j = 21 )</td>
<td>Obtained from ((\rho, P)) table for water.</td>
</tr>
<tr>
<td>( 50 \leq \text{JEOS}_j &lt; 100 )</td>
<td>Obtained from subroutine ( \text{PREQ}(J) ) where special pressure equations are stored.</td>
</tr>
</tbody>
</table>

In addition, if \( \text{OVERBD} > 0 \), then overburden pressure is added to \( P_j \). That is, for \( j \geq \text{JOVER} \)

\[
P_j = P_j + g \sum_{k=j}^{\text{JMAX}} (\rho \Delta x)_k
\]

\( \text{JOVER} \) is preset to 1, but may be given a different value by the user. To save machine time, the sum for \( k = \text{JOVER} \) to \( \text{JMAX} \) is calculated only once each cycle and, as necessary, the \( \rho \Delta x \)'s are subtracted from this sum to obtain each sum from \( k = j \) to \( \text{JMAX} \) which is used in the equation.

At present, the code carries an alluvium table (\( \text{JEOS}_j = 1 \)) iron table (\( \text{JEOS}_j = 2 \)) and water table (\( \text{JEOS}_j = 21 \)). Also, for \( \text{JEOS}_j = 50, 51 \) the subroutine \( \text{PRSEQ} \) calculates pressure as follows: (Tillotson Equations of State)

\[
n = \frac{\rho}{\rho_0}
\]

\[
\mu = \frac{\rho}{\rho_0} - 1
\]

\[
E = \rho e
\]

and \( A, B, a, b, \alpha, \beta, \rho_0 \), and \( ES \) are functions of the material (\( \text{JEOS}_j = 50, 51 \)).

If \( E < ES \), or \( E \geq ES \) and \( V \leq 1 \), then

\[
P = \left( a + \frac{b}{E_0 \eta^2 + 1} \right) E \eta + A \mu + B \mu^2
\]

-24-
If $\mu < 0$, then $B\mu^2$ is set equal to 0.

If $E \geq E_S$, and $V > 1$, then

$$P = a_1E + \left[ \frac{b_1E}{E} + A_\mu \epsilon \left( \frac{\rho_0}{\rho} - 1 \right) \right] \epsilon - a \left( \frac{\rho_0}{\rho} - 1 \right)^2$$

LAGRAN

Most of the PUFL equations are located here. Each variable is calculated for zones with $J = J_1$ through $J_2$ before the next variable is evaluated. $J_1$ and $J_2$ are lower and upper limits of activity in the pipe.

A brief outline of equations and logic flow in the LAGRAN subroutine follows:

1. Wall shear stress, $\tau_j$, is calculated first.
   - If $JEOS_j > 11$ or $JEOS_j < 4$, then
     $$\tau_j = \frac{CFRICT\rho u^2_j}{\rho \rho_j}$$
   - But if $4 < JEOS_j < 11$, then $\tau_j$ is calculated as follows. (See Fig. 4 for a flow diagram of $\tau_j$ calculation.)

![Flow diagram of $\tau_j$ calculation in LAGRAN subroutine.](image)

Fig. 4. Flow diagram of $\tau_j$ calculation in LAGRAN subroutine.
(a) If \( |u_{j+1}| < U\text{MIN} \), then \( \tau_j = 0 \). Go to velocity calculation. Otherwise, proceed.

(b) If switch \( j \) is on, calculate \( \tau_j \) using Eq. (4). Otherwise, proceed.

(c) If \( |u_{j+1}| < U\text{CTW} \) \( J\text{EOS}_j \), then \( \tau_j = C\text{I}TW \) \( J\text{EOS}_j \).
Otherwise, calculate \( \tau_j \) using Eq. (4).
To use Eq. (4)

\[
\tau_j = C2\text{TW} \times P_j
\]

(d) Finally,

- if \( u_j < 0 \), then \( \tau_j = -|\tau_j| \),
- if \( u_j \geq 0 \), then \( \tau_j = |\tau_j| \).

(2) Velocity is calculated.

\[
u_j^1 = \left\{ u - \frac{\Delta t}{m} \left[ \frac{V}{\Delta x} \Delta(P + Q) + (u - u_w) \Delta S + \tau S + mg \right] \right\}_j
\]

The velocity at the lower boundary, \( u_1 \), is calculated one of three ways. The BC1 table, which defines \( u_1 \), can be used as \( f(u, t) \), \( f(x, t) \), or \( f(P, t) \).

**Method 1:**

The BC1 table is most commonly used this way. If both switches BC1X and BC1P\( ^0 \), then the BC1 table is assumed to be \( f(u, t) \), and \( u_1 \) is picked up directly from the table. (Both BC1X and BC1P are preset to zero but may be changed by the user.)

**Method 2:**

If BC1X > 0, then the BC1 table is used by PUFL as \( f(x, t) \). First \( x_t \) is obtained from the table, then \( u_1 \) is calculated.

\[
u_1 = (x_t - x)/\Delta t.
\]

**Method 3:**

If BC1P > 0, then BC1 table is assumed to be \( f(P, t) \). \( P_t \) pressure at time \( t \) is obtained from the table, then \( u_1 \) is calculated.

\[
u_1 = \left\{ u - \frac{\Delta t}{m} \left[ \frac{V}{\Delta x} \left( (P + Q) - P_t \right) + (u - u_w) \Delta S + \tau S + mg \right] \right\}_j
\]

The velocity at the upper boundary, \( u_2 \), is calculated in a similar way except variable names BC2P, BC2X, and BC2 are used.

(3) New \( x \)-coordinates along the pipe are calculated.

\[
x_j^1 = [x + u_1 \Delta t]_j.
\]
(4) If $XADD > 0$ or $XC > 0$, then an additional test is made here. The code wants to know if $XJ\text{N}_1 > XADD$, and, if true, controls transfer to CXADD subroutine where a zone is added. (a cavity zone)

(5) If $OVERBD > 0$, then during each cycle the last zone is tested. It is discarded if $X_{JMAX} > OVERBD > 0$. That is $JMAX - 1 \to JMAX$.

If, in addition, $XOVER > 0$, first $j$ is found such that $x_j \leq XOVER < x_{j+1}$, then $j \to JOVER$. JOVER is the number of the first zone where over burden calculations are started.

(6) If $XRIGID > 0$, then $x_j$'s are never allowed to become less than $XJ1$. (The initial value of $x_1$.) Instead, all $x_j$ less than $XJ1$ are transferred to position between $XJ1$ and the first $x$ greater than $XJ1$.

(7) If $XABL < X_{JMAX}$, then the code will determine the value of $JEND1$. That is, $j$ is found such that $X_{j-1} < XABL < X_j$ and if $j < JEND1$, then $j \to JEND1$. JEND1 is allowed to move toward $j = 1$ direction only.

(8) Radius of pipe is calculated.

\[ r_j = \text{constant or } r_j = f(x, t). \]  
(RP tables are used.)

(9) Wrap-around surface area

\[ S_j = 2\pi \left( r \sqrt{\Delta x^2 + \Delta r^2} \right)_j. \]

(10) Volume

\[ V_j = \frac{\pi}{3} \Delta x_j \left( r_j^2 \Delta r_j + r_{j+1}^2 \Delta r_{j+1} + r_j r_{j+1} \right). \]

(11) Density

If $CMU > 0$, then for $j = JMU1$ through $JMU2$

\[ \rho_j = \left( \frac{\pi r_j^2 CMU}{V} \right)_j. \]

Otherwise,

\[ \rho_j = \left( \rho \frac{V_j}{V} + \Delta t \frac{S_j}{V} \right)_j. \]

(12) If $JACTIV = 0$, then controls transfer to subroutine ACTEST where activity tests are made.

(13) If $XSURF > 0$, then $X_{jmax}$ is tested. If $X_{jmax} > XSURF$, then the $jmax$ zone is discarded.

(14) If $JHOLD > 0$, then calculations along the pipe are stopped at $j = JHOLD$ until $P_{JHOLD} > PHOLD$. The pressure is allowed to build up in zone $JHOLD$ until the pressure limit has been exceeded. At this time, an addition increment of time equal to $TLAG$ is allowed to pass before the pressure is released and allowed to travel up the pipe.
(15) The artificial viscosity calculation is made next.
\[ Q_j = \rho \Delta u (C_0^2 \Delta u - C_1 A), \]
with \( A = \sqrt{\frac{\rho}{\gamma - 1}} \) if \( \text{JEOS}_j = 0 \) or \( \text{JEOS}_j > 100 \), or \( A = \text{sound speed of material} \) if \( 1 \leq \text{JEOS}_j \leq 100 \).

(16) Ablation limits \( J_{\text{ABL1}} \) and \( J_{\text{ABL2}} \) are determined. If \( SWQ_1 \) or \( SWQ_2 > 0 \), then a transfer to subroutine \( Q_{\text{ABL}} \) is made where \( J_{\text{ABL1}} \) and \( J_{\text{ABL2}} \) are determined. If \( X_{\text{ABL1}} \) or \( X_{\text{ABL2}} > 0 \), then a transfer to subroutine \( D\text{ETXAB} \) is made where \( J_{\text{ABL1}} \) and \( J_{\text{ABL2}} \) are found.

(17) If \( J_{\text{END1}} < J_{\text{MAX}} \), then as the maximum \( Q \) (shock front) presses down the pipe through zones with \( j \) subscripts \( \geq J_{\text{END1}} \), the time it passes each zone is saved along with the pressure of the zone two zones behind the shock front. If an RPR table has been entered in the data, the subroutine \( \text{RADIUS} \) will use it to obtain radii for these zones behind the shock front.

(18) Subroutine \( Q\text{CAL} \) makes \( Q_{\text{convection}} \) and \( Q_{\text{radiation}} \) calculations here.

(19) A transfer is made to subroutine \( D\text{T\text{CAL}} \) where the new delta \( t \) is calculated.

(20) The energy and pressure calculations are made.
(a) If \( \text{JEOS}_j = 100 \), then \( e_j^{n+1} \) and \( P_j^{n+1} \) are calculated in subroutine \( \text{HECAL} \), and these remaining \( e, P \) calculations are skipped.
(b) Otherwise these calculations are made. The subroutine \( \text{PRS} \) calculates the pressure, or \( P = e\rho(\gamma - 1) \) is used.

1) \[ e_j^{n+1} = f(P_j^n, Q_{\text{sum}}^n). \]
2) If \( \text{JEOS}_j = 200 \) then function \( G_j \) calculates a \( \gamma_j = f(e_j) \).
3) If \( \sigma > 0 \), \[ T_j^{n+1} = f(e_j^{n+1}) \text{ for } j = J_{\text{ABL1}} \text{ through } J_{\text{ABL2}}. \]
4) \[ P_j^{n+1} = f(e_j^{n+1}). \]
5) \[ Q_{\text{sum}}^{n+1} = f(e_j^{n+1}, T_j^{n+1}). \]
6) \[ e_j^{n+1} = f(P_j^{n+1}, Q_{\text{sum}}^{n+1}). \]
7) If \( \text{JEOS}_j = 200 \), then \( \gamma_j = f(e_j) \).
8) \[ P_j^{n+1} = f(e_j^{n+1}). \]
9) If \( \sigma > 0 \), then \[ T_j^{n+1} = f(e_j^{n+1}) \text{ for } j = J_{\text{ABL1}} \text{ through } J_{\text{ABL2}}. \]
10) \[ Q_{\text{sum}}^{n+1} = f(e_j^{n+1}, T_j^{n+1}). \]

(21) Variables at time \( n+1 \) replace variables at time \( n \), and the problem time is increased by \( \Delta t \).
(22) Special sums are made:

$$EPDV = \sum_n \sum_j \left( \frac{\Delta P + \Delta P_{n+1}^n}{2\Delta t_{n+1/2}} \right) \left( \frac{V'}{\Delta x} - \frac{V}{\Delta x} \right) \left( \frac{\Delta x + \Delta x'}{2} \right)_j$$

$$QLOSS = \sum_n \sum_j (Q_{sum} S\Delta t)^n_{j+1}$$

$$ADDMAS = \sum_n \sum_j (m S\Delta t)^n_{j+1}$$

$$ETOTMS = \sum_n \sum_j (E w \Delta mass)^n_{j+1}$$

(23) Subroutine ADMAS calculates mass added to zones subscripted by $j = JABL1$ through $JABL2$. If MFRACT > 0, then subroutine FMDOOT calculates fraction of $\dot{m}$ to be added to zones.

(24) When edit, plot, or dump time is reached a transfer to the subroutine PUFL is made here.

(25) Lagrange calculation loop is repeated.

ADMAS Mass addition calculations are made for zones with $j = JABL1$ through $JABL2$.

If $e_j \geq EFORM$ and $|U_j| \geq UFORM$, then $m_j$ is calculated. Otherwise, $\dot{m}_j = 0$.

$$Q^*_j = HOFW + HOFV + ETA \left( \gamma e + \frac{U^2}{2} \right)_j$$

If $Q^*_j = 0$, then DOTMC $\rightarrow m_j$.

Otherwise,

$$\dot{m}_j = \frac{Q_{sum} j}{Q_j} + DOTMC$$

If $\dot{m}_j = 0$, then $0 \rightarrow EW_j$ and $0 \rightarrow UW_j$.

Otherwise, if CALPHI = 0, then the input value of PHI is used. If CALPHI > 0, then PHI is calculated as

$$PHI = \left( \frac{U^2}{2e} \right)_j$$

If ALPHA < 0, then $EW_j = HOFV + HOFW,$

$UW_j = 0$. 

-29-
If $\text{ALPHA} \geq 0$, then

$$E_W = \text{ALPHA} \times Q_{j}^{+}/(1 + \text{PHI})$$

$$U_W = (2 \times \text{PHI} \times EW)^{1/2}$$

If $U_j < 0$, then $-U_W \rightarrow UW_j$.

Finally, the total energy from the source is calculated which is printed at edit time

$$T_{EFROMS} = \sum_{n=1}^{N} \sum_{j=\text{JABL1}}^{\text{JABL2}} \left( m_{\text{SAt}} \left( EW + \frac{U_{W}^2}{2} \right) \right)^{n+1}$$

**FMDOT**

If $\text{FRACTM} > 0$, then $w_j$, the weighting factors for $\dot{m}_j$, are calculated and used. $(w_j \times \dot{m}_j \rightarrow \dot{m}_j)$

This calculation attempts to simulate a time-dependent mass entrainment rather than the instantaneous mixing. Also, this method adds mass along the Eulerian grid which is fixed instead of along the Lagrangian grid which moves.

Initially the pipe is divided into $m$ Eulerian zones ($m = 200$) by generating $x_m$, $m = 1, 2, \ldots, 200$. They remain fixed during execution. Each cycle, $w_m$, the weighting factor for $\dot{m}_m$, is calculated for each $m$th zone.

$$w_m = 1. - \left[ 1. - \left( \frac{T_n |U_m| \sqrt{CF}}{2} \right) \right]/R_n$$

with $T_n$, the time elapsed since $Q_{SUM}_m \neq 0$

$U_n$, the velocity

$R_n$, the radius

$Q_{SUM}_m = (Q_{\text{convection}} + Q_{\text{radiation}})_m$

All the $m$-subscripted variables refer to the $m$th zone of the fixed Eulerian grid and are obtained from corresponding variables associated with the Lagrangian grid which are subscripted by $j$.

Next $w_j$ is obtained from $w_m$ by projecting $w_m$ onto $w_j$. Finally, for each $\dot{m}_j \neq 0$, $w_j \times \dot{m}_j \rightarrow \dot{m}_j$.

**QCAL**

Heat flux is calculated for zones with $j = \text{JABL1}$ through $\text{JABL2}$.

If $e_j \geq \text{EFORM}$ and $|U_j| \geq \text{UFORM}$, then $Q_{CON}_j$ is calculated. Otherwise, $Q_{CON}_j$ is set to 0.

If switch $\text{SWQCH} = 0$, then

$$Q_{CON}_j = \left( \text{STAN} \times \gamma \times P \times U/\gamma - 1 \right)_j$$
If SWQCH > 0, then
\[
Q\text{CON}_j = \left[ \left( \text{STAN} \times \rho \times U \times \left( \gamma \times e + \frac{U^2}{2} \right) \right) \right]_j.
\]
\[
Q\text{RAD}_j = \sigma \left( \frac{\text{TEV}^4}{\text{OPACITY} \times \rho} \right)_j
\]
\[
Q\text{sum} = Q\text{CON}_j + Q\text{RAD}_j
\]
If \( (Q\text{sum} \times \Delta t)/m > \frac{e_j}{100} \), then \( Q\text{sum}_j = \left( \frac{1}{100} \frac{\text{me}_{j}}{\text{SAT}} \right)_j \) replaces the above equation.

**ACTEST** Tests to find the lower and upper limits of zone activity along the pipe. A zone is active if \( \rho' \neq \rho \), if \( r' \neq r \), or if \( u' > 0 \) for the zone.

**CHECKT** Checks to determine if the correct dump tape is being read into memory for the submitted problem card deck. This check is made by comparing the first 60 characters on the data deck title card with the first 80 characters of the title in a dump record. All 60 characters must be equal for the program to continue. The PUFL code makes this check for the first record and also the one from which the problem starts running. The remaining 20 characters on the title card may be changed each time the problem is resubmitted, if desired.

**CXADD** If \( X\text{ADD} \neq 0 \) or \( XC \neq 0 \), then each cycle that \( X_1 > X\text{ADD} \) this subroutine will add a zone at the cavity boundary \( (j = 1) \). When a zone is added, all of the original zones are moved up by one subscript, and the variables for the first zone are generated and stored in the vacated subscript 1 locations.

**DATALD** Reads in the PUFL data submitted in the card deck. Two sets of data cards are loaded into memory using Alex Cecil's Data Directed Leader routine. The end of each is indicated by a card containing the word END. The first set is loaded before the dump tape is read and contains the initial conditions. If, in this set, the problem calls for a dump record from the dump tape, that is, \( \text{RTAPE} > 0 \), then the dump record requested is read into memory. Next, the second data set submitted in the card deck is read in modifying the dump record in memory. This second set may consist of only the card containing the word END.

**DUMPE** If \( \text{EDDUMP} > 0 \), then, shortly after execution starts, controls transfer to this subroutine where each dump on the dump tape is edited followed by termination of the program.

**DETXAB** Determines the lower and upper limits for ablation. If \( X\text{ABL1} > 0 \), first \( j \) is found such that \( X_{j-1} \leq X\text{ABL1} < X_j \), then \( j \rightarrow J\text{ABL1} \). If \( X\text{ABL2} > 0 \), first \( j \) is found such that \( X_{j-1} \leq X\text{ABL2} < X_j \), then \((j - 1) \rightarrow J\text{ABL2} \).
DTCA1: During each cycle, the new $\Delta t$ is chosen as the minimum of these seven $\Delta t$'s.

The first five are minimums over the range of active $j$ zones.

1. $\Delta t_u = \min \left( \frac{\Delta u}{|u|} \right) DTU$, with $\Delta u = x_{j+1} - x_j$ if $u_j > 0$
   $\Delta u = x_j - x_{j-1}$ if $u_j < 0$
   DTU preset to 0.1

2. $\Delta t_{ssp} = \min \left( \frac{\Delta s}{\gamma P^{1/2}} \right) DTSSP$
   DTSSP preset of 0.2

   If $JEOS_j \geq 1$, then a sound speed number for the material in the zone is used in place of $\left( \frac{\gamma P}{\rho} \right)^{1/2}$.

3. $\Delta t_v = \min \left( \frac{V \Delta t}{|\Delta v|} \right) DTV$ for $\Delta v = V' - V \neq 0$
   DTV preset to 0.01

4. $\Delta t_m = \min \left( \frac{m}{S |\dot{m}|} \right) DTM$
   DTM preset to 0.01

5. $\Delta t_Q = \min \left( \frac{m e}{Q_{\text{convection}}} \right) DTQ$
   DTQ preset to 0.01

   $\Delta t_a$, $\Delta t_b$ are determined by non-zero velocity at the lower and upper boundaries, respectively.

6. $\Delta t_a = 1.05 \left( x_2 - x_1 \right) / \left| x_2 U_2 - x_1 U_1 \right|$
   if $U_1 \neq 0$ and $\left| x_2 U_2 - x_1 U_1 \right| > 10^{-15}$.

7. $\Delta t_B = 1.05 \left( x_{j_{\text{max}}} - x_{j_{\text{max}}-1} \right) / \left| x_{j_{\text{max}}} U_{j_{\text{max}}} - x_{j_{\text{max}}-1} U_{j_{\text{max}}-1} \right|$
   if $U_{j_{\text{max}}} \neq 0$, and $\left| x_{j_{\text{max}}} U_{j_{\text{max}}} - x_{j_{\text{max}}-1} U_{j_{\text{max}}-1} \right| > 10^{-15}$.

DTU, DTSSP, DTV, DTM, DTQ are constant multipliers defined by the user.

EDITOR: Generates edited output at time intervals requested by the user. Also, after sense switch 1 is down, extra end-of-run edits are made.

In each edited output package, the first set of nine columns listed for each zone, at time $n$, are $j$, EOS number, $x$, $m$, $u$, $\rho$, $Q$, $e$, $P$. The energy check sums are printed out next, followed by the second set of nine columns which are $j$, EOS number, $Q$ radiation, $r$, $T$, $m$, $T$, $S$, and $Q$ convection.

EPDVCL: Makes a comparison, only if XMSTAR > 0, between the value of $\Sigma P_{dv}$ calculated two different ways.

- using the PUFL grid which is not stationary, and
- using a superimposed fixed grid.

GCAL: Calculates $\gamma = f(e)$ using $(\gamma, e)$ tables, when $JEOS_j = 200$. 

---
HECAL  Calculates e and P using the high explosive equation, when \( J_{EOS_j} = 100 \).

\[
e'_j = \left[ e - (P + Q)(V' - V)/m \right]_j
\]

\[
\eta = \rho_j / 1.84
\]

If \( F_j > 0.96 \), then go to pressure, energy calculations.

If density (\( \rho \)) decreases, set \( F_j = 1 \). (Total pressure is released.)

Otherwise, \( F_j = 3.658 \left( 1 - \frac{1}{\eta} \right) \).

**P, e calculations**

\[
\text{Term} = \left\{ \left[ -3.4844 \times 10^{-3} \eta^4 + 6.43586 (1 - 0.0875\eta) e^{4/\eta} \right] \times 10^{12} \right\}_j
\]

\[
P'_j = \left[ \left( \text{Term} + 0.35pe \right) F \right]_j
\]

\[
e'_j = \left[ e' - (P' - P)(V' - V)/2m \right]_j
\]

\[
P'_j = \left[ \left( \text{Term} + 0.35pe \right) F \right]_j
\]

**OFFMON**  Writes dump records onto tape 5.

**ONMON**  Reads dump records from tape 5. **ONMON** is used when \( RTAPE > 0 \) in the card deck. In addition, if \( DUMP = n > 0 \), then the \( n \)th dump record on tape 5 is found and loaded into memory.

If \( DUMP = 0 \), its preset value, then the last dump record on tape 5 is loaded into memory, and the program continues calculations from this dump.

**PICKP**  Picks up and saves pressures at locations \( XPK_L \) and at time intervals \( DTPICK \) chosen by the user. At each of these times and for each \( L \), \( j \) is found such that \( X_{j-1} < XPK_L < X_j \), then the corresponding \( P_L \) is calculated and saved with the time. After sense switch 1 is turned on, these pressures are edited and plotted.

**PLOTP**  Generates the \( (T, P_L) \) plots after sense switch 1 is put down.

**POSTAP**  Positions the tape which contains \( (T, P_L) \) data, at the beginning of a machine run, but before a problem continues.

**PRESET**  Initializes variables at the beginning of cycle 0, after all of the card data has been read.

**PUFLOK**  Makes eight DD80 plots after data is edited. The plots show \( x \), the independent variable, vs \( u \), \( P \), \( \rho \), \( m \) and \( e \), \( Q_{conv} \), \( Q_{rad} \), and \( m \).

**QABL**  If \( SQW1 > 0 \), then finds the maximum \( Q \) for \( j = 1, 2, \ldots, J_{END1} \) and places \( j + 1 \rightarrow J_{ABL1} \). If \( SQW2 > 0 \), then finds the maximum \( Q_j \) for \( j = J_{END1}, \ldots, J_{MAX} \) and inserts \( j \rightarrow J_{ABL2} \).
RADIUs Calculates radii. If the RP table has no entries, then RADC \( \rightarrow r_j \). Otherwise, \( r_j \)'s are taken from the RP table which is \( f(X, R, T) \). In addition, the RPR table can be used to define the radius expansion a short time after the shock front has passed. That is, for all \( j = JEND1 \) through \( j \) for \( Q \) maximum, \( r_j \) is taken from the RPR table after a time lag equal to TLAG. If \( RPR_j < 0 \), then the table is used as \( f(X, R, T) \). If \( RPR_j \geq 0 \), then it has the structure \( f(P, R, T) \). For each \( j \) zone, the larger \( j \) from these sources is always used.

REZONE Removes zones whose subscripts are given in the JOUT list before the main calculations are started.

TCAL Calculates temperature, if SIGMA > 0, of each zone for \( j = JABL1 \) through \( JABL2 \) using the \((\log E, \log T)\) table.

TECHK Makes energy check sums which are printed out at edit times.

UPTAB Presets \((u, P)\) tables.

XXPICK If XPICK_1 or XPICK_2 \( \neq 0 \), then the points for the TIME plots are picked up here and saved on tape 7.

XXPLOT After sense switch 1 is put down if XPICK_1 or XPICK_2 \( \neq 0 \), then TIME vs \( \rho, u, P, R, E, Q_{conv}, Q_{rad} \) plots, one set for each XPICK_1 submitted, are made here.

STOPSW This subroutine is called by the PUFL code so that it can run on "background" during the daytime. During evening production runs, a dummy STOPSW binary deck needs to be inserted in the PUFL card deck to prevent the STOPSW routine from stopping the production run unexpectedly early.
Changing the PUFL Code

Usually a new instruction tape is made when any of the PUFL FORTRAN source decks are changed. Binary cards are never used. The PUFL card deck, which generates a new tape and tests it with a PUFL problem, has this general structure:

* ID CARD
* T NEWTPE=14=NEWTPE
* T OLDTPE=6=OLDTPE
* XEQ
* LIBE

- PUFL XTRA deck which contains the common cliche.
- All source decks that have been changed (in alphabetic order with respect to their subroutine names).
- The five-card PRELIBE deck which causes a new library tape to be made on tape 10 by updating the old library tape on 6.
* DATA
* PUFL data cards

If the changes are so extensive that a new instruction tape needs to be made, then the card deck structure is as follows:

* ID CARD
* NEWTPE=14=NEWTPE
* XEQ
* LIBE

- Subroutine XTRA deck which contains the common cliche.
- All PUFL FORTRAN source decks in alphabetic order with respect to their subroutine names.
- The three-card PRELIBE deck which causes a new library tape to be written on tape 10.
* DATA
* PUFL data cards
Sample PUFL Problems

WITH MINIMUM INPUT DATA

* DATA
  PUFL SMALL TEST PROBLEM
  *T BLANK
    NSTCP= 10
    HADC= +1.
    CFRICT= +0.02  STAN= +0.002  ETA= +2.  SIGMA= +2.84E-5
    ALPHA= -1.  HOFV= +1.3E+11
    JABL1= 1  JABL2= 200
    JLAST= 100  200
    DX= +.1 +1.
    RHO= +.2 +.001
    P= +4.E+11 +1.E+6
    GJ= 1.4*2
    MAT= 0 2
    MLAST= 100 200
  END
  END
  RTAPE= 1.

Comments:

* NSTCP= 10 will cause PUFL to exit after running 10 cycles.
* RTAPE= 1. is stored at the end of the data deck where PUFL does not see it.

To continue a problem, this card is placed in front of the two end cards.
WITH EXTENSIVE INPUT DATA

* DATA
PUFL TEST PROBLEM
*T AB319
UMIN= +10.  TOTALT= +1.E-5
RAD= +1.
BC1P= +0.8  BC1X= +0.  BC1= +0.3 +1.
BC2P= +1.  RC2= +1.E+6  +0.  +1.E+6 +1.
EDDT= +.5E-6 +2.E-6
EDITT= +20.E-6 +1.
CFRIC= +.002  STAN= +.002  ETA= +.2  SIGMA= +2.84E-5
ALPHA= -1.  MO= +1.3E+11
JBL1= 1  JBL2= 200
JLAST= 100  200
DX= +.1 +1.
RHO= +.2 +.001
U= +0. #2
P= +4.E+11 +1.E+6
GJ= +1.4 +1.4
MAT= 0 2
MLAST= 100 200
RP= -100. +0. +1. +0. +1. +100. +0. +44
+10. +0. +1. +0. +1. +100. +0. +44
+20. +0. +2. +0. +2. +100. +0. +44
JL= 1 51 101 151
JH= 50 100 150 200
JXS= 50 100 150
END
END
RTAPE= 1.

Acknowledgments

PUFL calculations have been written by B. K. Crowley. Support and encouragement for the development of PUFL have been given by J. Knox. J. Shaw is currently working on a CDC 7600 version of PUFL which offers a variety of teletype controls over the running program.

References
