Title
BOUNDL: A PROGRAM FOR CALCULATING FLOW PAST A SEMI-INFINITE FLAT PLATE USING THE VORTEX SHEET METHOD

Permalink
https://escholarship.org/uc/item/4pg1f284

Author
Cheer, A.Y.

Publication Date
1978-02-01
BOUNDL: A PROGRAM FOR CALCULATING
FLOW PAST A SEMI-INFINITE FLAT
PLATE USING THE VORTEX SHEET METHOD

A. Y. Cheer

February 1978

Prepared for the U. S. Department of Energy
under Contract W-7405-ENG-48

For Reference

Not to be taken from this room
DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.
BOUNDL: A program for calculating flow past a semi-infinite flat plate using the vortex sheet method.

A. Y. Cheer

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract

BOUNDL is a computer program which implements the Vortex Sheet Method for approximating boundary layers [1]. The specific problem of flow past a semi-infinite flat plate is considered. Listings of the main program and its subprograms, together with their respective flow charts, are enclosed to facilitate the documentation process.

Work supported by the U. S. Department of Energy.
Introduction

Program BOUNDL was written by A. J. Chorin implementing his method described in [1]. In the process, it has gone through many generations of code, and thus can be confusing. In this paper, I will attempt to relate all the secrets that link the code to the method. References will be made to the lines of code and their corresponding formulae in [1] as much as possible.

Section one describes the function of the main routine and its relation to the subroutines. The next section describes subroutine WALL. This subroutine corresponds to pages 8, 9 and 10; the vorticity creation section of [1].

In section three, subroutine DISPL is documented. This routine calculates and prints the drag, the displacement thickness and the boundary layer for flow past a semi-infinite flat plate. Also it sets up formats for plotting figures 1 and 2 in [1]; and it sets up arrays like \( \text{UXA}(I) \) for further use by the main routine. Finally, section four documents subroutine STEP. Here, the random numbers \( \eta_i \) are generated and a random step taken according to the formulae on pages 4 to 7 of [1].
1. **Program BOUNDL (main routine)**

The following are definitions of variables in the COMMON file:

- **H**
  - \( h \) where \( h = \Delta x = \) length of sheets

- **N**
  - Total number of vortex sheets created. (\( N \) increases in time by the total number of vortex sheets created at each time step IADD, and is decreased by the number of sheets \( Q_i \) with center \((X_i, Y_i)\) that flowed out of the domain of interest.)

- **L**
  - Number of points on the x-axis where sheets are created (partition of the x-axis).

- **VEL**
  - \( U_\infty \) (velocity at infinity or freestream velocity)

- **RE**
  - Reynold's number

- **TSIG**
  - \( 2 \) times the variance

- **DT**
  - \( k \) where \( k = \Delta t = \) time step

- **PI**
  - Constant \( \pi = 3.1415926536 \)

- **TPI**
  - \( 2\pi \)

- **MM**
  - Number of sheets created at each point \((X_i)\) of the x-axis. (The sum of all MM for each \( X_i \ i = 1, ..., L \) is equal to IADD where IADD = total number of vortex sheets created at each time step.)

- **CM**
  - \( \xi_{\text{max}} \) (maximum allowable intensity for each vortex sheet).

- **LOOK (30,30)**
  - Variables used in subroutine DISPL to aid the output formatting of figure 1 of [1].

- **LS**

- **MS**

- **HX**

- **XO**

- **YO**

- **X(500)**
  - Array storing the X-coordinate \( X_i \)

- **Y(500)**
  - Array storing the Y-coordinate \( Y_i \)

- **S(500)**
  - Array storing \( \xi_i \) (intensity of vortex sheet \( Q_i \))

- **DX(500)**
  - Array storing \( U_i \) (first-velocity component of \( U_i = (U_i, V_i) \))

- **DY(500)**
  - Array storing \( V_i \) (2nd velocity component)
Following is a list of local variables and their definitions:

UXA(100) Array storing average velocity: $U_i$ average
DRAGA Drag
VAR Variance of drag
RIGHT Rightmost point on the x-axis under consideration.
NOLD Holds the previous value of N (this variable is used to aid the sorting routine in subprogram WALL).
MN(500) Tags. (Each vortex sheet created is assigned a tag $M_N(I)$. This tag aids in the assigning of the random variable $\eta_i$. Every sheet with the same value $M_N(I)$ gets the same $\eta_i$).
MNO Holds the last tag used at the previous point. Usually $MNO = MNO + MNMAX$.
MNMAX Maximum number of sheets one is allowed to create at each position $X_i$.

Following is a list of local variables and their definitions:

NMAX Maximum number for N to reach
NMIN Minimum number for N

Note: NMAX and NMIN was not used in this particular problem.

NAV This is usually an integer $> 0$. For each NAV number of steps, the main routine calculate averages for drag, variance and velocity.
CNAV NAV (Real variable, not integer)
NSTEP Total number of steps to be taken
TIME $t = time$
The first task BOUNDL does is initialize all its variables, a list of which are given above. This corresponds to lines 3-40 of the code. Figure 10 is the output corresponding to the values L = 7, H = 0.2, DT = 0.2, NAV = 20, CM = 0.1 and RE = 1.E + 6.

Secondly, the time step is advanced and a call to subroutine WALL to create vortex sheets is executed. After the vorticity is created, another call is made to subroutine DISPL. Here, the drag, the variance and the velocity profile (UXA(I)) are calculated, stored and displayed if desired.

Next, BOUNDL checks to see if twenty steps have been taken since the last time averages were computed. If yes, then the average profile, the average drag and the variance are calculated and printed. Loop 5 calculates the average profile, Loop 6 prints the profile and Loop 3 reinitializes the array UXA(I) to zero. Lines 59 and 62 of the code calculates the average drag and variance; lines 60 and 65 outputs the values, and lines 66 and 67 reinitializes the variables VAR = 0. and DRAG = 0.

If twenty time steps have not ellapsed, or after averages are computed, BOUNDL calls subroutine STEP to generate random numbers. Each vortex sheet with different tags gets assigned a different random number, and a random step is taken.

Finally, the controlling loop; loop 1; is advanced and the program checks to see if the number of steps already taken is ≤ NSTEP. If yes, then the above process is repeated. If no, the program exits and execution halts.
**Fig. 1**
2. **Subroutine WALL**

This subroutine corresponds to the Vorticity Creation section of [1]. The new variables in this routine are:

- \( \text{EPS} \) \( 0.5 \times \text{CM} = 1/2 \ \xi_{\text{max}} \) (this is the maximum allowable intensity of each vortex sheet for this run)
- \( \text{SS} \) used to calculate \( U_i \)
- \( \text{IADD} \) Counter for the number of vortex sheets created on this call.

For each \( j=1, \ldots, N, Y_j > 0 \) and \( |X_i - X_j| < h \), Loop 2 of this routine calculates:

\[
U_i = U_\infty - \sum_{j=1}^{N} \xi_j \ d_j
\]

which is a modification of equation (4a) of [1]. This summation is done in the following steps:

1) Line 90 corresponds to the condition \( Y_j > 0 \)

2) Line 92 is: \( D = \text{ABS}(XX - X(J)) \)

\[
= |X_i - X_j|
\]

3) Line 93 is condition (4c) of [1]:

\[
|X_i - X_j| < h
\]

4) Line 95 is equation (4b) of [1] where:

\[
C = (H-D)/H = \frac{h - |X_i - X_j|}{h} = 1 - \frac{|X_i - X_j|}{h} = d_j
\]
5) Line 96: \[ SS = SS - S(J) * C \quad j=1, \ldots, N \]

\[ = U_\infty - \sum_{j=1}^{N} \xi_j d_j \]

where \( SS \) is initialized to \( U_\infty \), \( C = d_j \) and \( S(J) = \xi_j \). This quantity \( SS \) also corresponds to \( U_0 \) on pg. 8 of [1].

Now that the value for \( U_0 \) is known, line 100 of the code checks to see if \( |2U_0| < \text{EPS} \) \((\text{=1/2 } \xi_{\text{max}}\)) If yes, then it advances to the next position \( x_{i+1} \) and calculate \( U_{i+1} \) as above. This is done so long as \( x_{i+1} \) is in the domain of interest, i.e., \( x_{i+1} < \text{RIGHT} \). If \( |2U_0| > \text{EPS} \), then it breaks \( 2U_0 \) into an even number of vortex sheets, MM or DIV, each with the same intensity \( SS/\text{DIV} \).

Next, loop 4 (lines 108-121) checks to see if the number of sheets created \( JADD < \text{MNMAX} \), where \( \text{MNMAX} \) is the maximum allowable. If no, then set \( JADD = \text{MNMAX} \) and print a warning. Else, increase the counter \( IADD \) and create a new tag for each sheet: \( \text{MN}(N + IADD) = \text{MNO} + JADD \).

\( \text{MN}(I) \) is the array of tags corresponding to the vortex sheet with center at \((X(I), Y(I))\) and intensity \( S(I) \).

Note that \( JADD \) is reinitialized at each \( X_i \) position, but \( \text{MNO} \) is kept fixed for all \( X_i \) at the same time step. Hence, different vortex sheets, at different \( X \) position, at same time step, can have the same tag \( \text{MN}(I) \).

Thus, we have that at each \( X_i \) position, \( i=1, \ldots, L \), \( U_i \) is calculated, and vortex sheets are created according to the above algorithms.

At this point, the variable \( N \) is updated: \( N = N + IADD \). In words, the total number of sheets to date, equals the number before, plus the number created. Also, \( \text{MNO} \) is reinitialized to \( \text{MNO} = \text{MNO} + \text{MNMAX} \).
This will ensure that brand new tags will be used at the next time step.

Loop 16 and 17 now takes over the task of sorting the sheets created above according to the values of their tag MN(I). This sort is done to facilitate the assignment of the random number \( n_i \) in subroutine STEP, where the same \( n_i \) is assigned to sheets with the same tag MN(I).

Now, for \( I = 1, \ldots, N \), loop 9 checks to see if \((X(I), Y(I)) \) is in the domain of interest. i.e., \( X(I) < \text{RIGHT} \) and \( Y(I) > 0 \). If yes, do nothing. If not in the domain of interest, delete if from the stack, decrease the number of vortex sheets \( N \) by one, and move the trailing stack up one position. This moving procedure is done in loop 10. Finally, print \( N \) and allow control to return to main routine.
3. **Subroutine** DISPL

This subroutine does the following:

1) Sets up formats to output figures 1 and 2 of [1]
2) calculates and prints the drag, the displacement and the boundary layer, and
3) stores results of velocity profile, drag and variance for further use by the main program.

Here are definitions of some local variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPR</td>
<td>If JPR = 1, then DISPL skips loops 1, 2 and 3 (lines 292 to 316) which outputs figure 1 of [1]. Also, skips loop 6 (lines 340 to 343) which outputs figure 2 of [1].</td>
</tr>
<tr>
<td>ETA</td>
<td>Similarity variable.</td>
</tr>
<tr>
<td>DRAGR</td>
<td>Real drag (for flow past a semi-infinite flat plate)</td>
</tr>
<tr>
<td>DRAG</td>
<td>Drag computed by this program</td>
</tr>
<tr>
<td>DISP</td>
<td>Displacement thickness</td>
</tr>
<tr>
<td>RBDL</td>
<td>Boundary Layer (computed)</td>
</tr>
<tr>
<td>RBDL1</td>
<td>Real boundary layer used for comparison.</td>
</tr>
</tbody>
</table>

The first thing DISPL does, after initializing its local variables, is set up formats to output figure 1 of [1]. Here, a printed "**" indicates the center of a vortex sheet. Note that this section of code (lines 291-317) is executed only if JPR ≠ 1.

Next, loop 40 initializes array UX(I) to U₀, for I = 1, ..., N.

Then, loop 4 calculates

\[ U_j = U_j - \sum_{i=1}^{N} \xi_i d_i \quad j = 1, \ldots, ku \]

for \(|x - x_j| < h\).
Here, \( ku \leq 20 \) since 20 is the maximum number of partition points in the y-direction for each \( X_i \).

Now, if \( JPR \neq 1 \), loop 6 calculates the values for \( \eta \)T which are used in plotting figure 2 of [1]. If \( JPR = 1 \), \( \text{DISPL} \) jumps control to line 346 where the values of \( UX(I) \) calculated above are stored in array \( UXA(I) \). This array is used by the main routine to compute the average velocity profile.

The remainder of the code is devoted to calculating the drag and the displacement according to the specification of formulae on pages 12-14 of [1]:

a) Loop 7 calculates:
\[
SS(J) = S(I) * C = \sum_{i=1}^{M} \xi_i d_i
\]
where \( M = \min(N,100) \) and \( |X - X_i| < h \).

b) Loops 10 and 11 sorts the array of vortex sheets such that
\[
Y_1 \leq Y_2 \leq \ldots \leq Y_m.
\]

c) Loop 13 calculates
\[
U = U - SS(JP) = U_\infty - \sum_{j=1}^{JJ} \xi_j d_j
\]
where \( SS(JP) \) is from loop 7 above.

d) Finally loop 12 calculates:
for \( I=1,\ldots,\text{JJ} \), where \( \text{JJ} = \min(N,100) = M \).
(i) Line 389 of code:

\[
\text{DRAG} = \text{DRAG} + U \ast (\text{VEL} - U) \ast Z
\]

\[
= \sum_{i=1}^{M} U_i(U_\infty - U_i) \Delta Y_i
\]

where \( Z = YY(I) - YY(I - 1) \) (line 382 of code)

\[
= Y_i - Y_{i-1}
\]

\[
= \Delta Y_i
\]

and \( U \) is from loop 13 above.

(ii) Line 390 of code:

\[
\text{DISP} = \text{DISP} + (\text{VEL} - U) \ast Z
\]

\[
= \sum_{i=1}^{M} (U_\infty - U_i) \Delta Y_i
\]

Now, the real drag \( \text{DRAGR} \) and the computed drag \( \text{DRAG} \) are computed and printed. Also, the real and computed boundary layer, \( \text{RBDL}1 \) and \( \text{RBDL} \) respectively, are outputted. Finally, variables \( \text{DRAGA} \) and \( \text{VAR} \) are incremented and stored away for used by the main routine.
Fig. 5
C DRAG

J1=0

DO7=I,N

DRAG( WHERE-X(I) )

IP(DO7,H)GOTC7

C=(-D/3)*T

J1=JJ+1

IP(JJ+CT,H)GOT07

YY(JJ)=Y(I)

SS(JJ)=S(I)

7 CONTINUE

IMNTG005, JJ

JJM=JJ-1

DO10=I, JJM

ID=I+1

DO11*IP, JJ

IF(VV(I),YL,YY(J))GETC11

YT=YY(I)

ST=SS(I)

YY(I)=YY(J)

SS(I)=SS(J)

YY(J)=YT

SS(J)=ST

8 CONTINUE

10 CONTINUE

ID=0

DISP=0

DO12=I, JJ

IP(JJ+GQ,0)GOTC12

Y=YY(I)

IF(I*NE.I)Z=YY(I)-YY(I-1)

U=VEL

DO15=I,JJ

JJ=JJ+1

L=L-SS(JP)

13 CONTINUE

DRAG=CRAG*V*(VEL-U)*Z

DISP=DISP+(V*VEL-U)*Z

12 CONTINUE

DRAG=DRAG*654*SGF( WHERE / F E )

DTAG=DRAG*100

PRINT C07,DRAG,DRAG

RE1L=EPRE

SGF(EF=*FE)

PRINT C09,DISP,RE1L,RE1L

CRAC=CRAC+*C

VAR=VAR+*C

401 9000 FORMAT(* )

402 9001 FORMAT(X,3F11.7)

403 9002 FORMAT(* VELOCITY FRCFILE*)

404 9004 FORMAT(X,6OA1)

405 9005 FORMAT(* SHEET IN SLICF*,15)

406 9006 FORMAT(X,F11.7)

407 9007 FORMAT(* DRAG*2F11.7)

408 9009 FORMAT(* H*F11.7)

409 9010 FORMAT(* DISF AND FE*,3F15.7)

410 RETURN

END

Fig. 6
4. Subroutine **STEP**

First, this subroutine creates the random numbers \( \eta_i \). Then, each vortex sheet takes a random step as prescribed by formula (6a) and (6b) on page 6 of [1].

In order to use formulae (6a) and (6b), we need to know \( \hat{U}_i = (U_i, V_i) \). So, for each vortex sheet \( Q_i, i=1, \ldots, N, |X_i - X_j| < h, Y_j > 0 \), loop 1 and 2 calculates \( U_i \) and \( V_i \) in the following manner:

A) Lines 198 to 206 of the code corresponds to the integral (5b) of [1]: \( U_\infty - I_1 \).

1) line 198: \( D = \text{ABS}(X(J) - X(I) - HH) \)

\[
D = |X_j - X_i - \frac{h}{2}|
\]

\[
= |-(X_i - X_j + \frac{h}{2})|
\]

2) line 199: \( D \leq H \Rightarrow 0 \leq d^+ \leq 1 \)

3) line 201: \( C = (H - D)/H \)

\[
C = \frac{h - |-(X_i - X_j + \frac{h}{2})|}{h}
\]

\[
= \frac{1}{h} |X_i - X_j + \frac{h}{2}|
\]

\( = d^+ \)

the smoothing coefficient corresponding to formula (5d) of [1].

4) line 202 and 203 corresponds to (5f) of [1].

Namely, \( YY = \min(Y_j, Y_i) \)

\[
= Y_j^* \text{ the displacement} \]
5) line 205: \[ G_l = G_l + S(J) \cdot YY \cdot C \]

\[ = \sum_{j^+} \xi_j \cdot Y_j \cdot d_j^+ \]

\[ = U_\infty - I_1 \]

B) Lines 207 to 215 calculates the integral \( U - I_2 \) which corresponds to (5c) of [1]:

1) line 207: \( D = \text{ABS}(X(J) - X(I) + HH) \)

\[ = \left| X_j - X_i + \frac{h}{2} \right| \]

\[ = \left| -(X_i - X_j - \frac{h}{2}) \right| \]

2) line 208: \( D < H \Rightarrow 0 \leq d_j^- \leq 1 \)

3) line 210: \( C = (H - D)/H \)

\[ = \frac{h - \left| -(X_i - X_j - \frac{h}{2}) \right|}{h} \]

\[ = 1 - \left| \frac{X_i - X_j - \frac{h}{2}}{h} \right| \]

\[ = d_j^- \quad \text{formula (5e) of [1].} \]

4) lines 211 and 212 again is: \( YY = \min(Y_i, Y_j) = Y_j^* \)

5) line 214: \( G_2 = G_2 + S(J) \cdot YY \cdot C \)

\[ = \sum_{j^-} \xi_j \cdot Y_j \cdot d_j^- \]

\[ = U_\infty - I_2 \quad \text{by (5c) of [1].} \]

c) Lines 216 to 220 calculates (4a) of [1]:

\[ U_i = U_\infty - \frac{1}{2} \xi_i - \sum_j \xi_j d_j \]

1) line 189: \( U = \text{Vel} \Rightarrow U = U_\infty \)

2) line 190: \( U = U - S(I) \cdot 0.5 \)

\[ = U_\infty - \frac{1}{2} \xi_i \]
3) line 216: \( D = \text{ABS}(X(J) - X(I)) \)
\[ = |X_j - X_i| \]
\[ = |-(X_i - X_j)| \]

4) line 217: \( D < H \Rightarrow |X_i - X_j| < h \)

5) line 219: \( C = (H - D)/H \)
\[ = 1 - \frac{|X_i - X_j|}{h} \]
\[ = d_j \quad \text{by (4b) of [1].} \]

6) line 220: For \( Y_j > Y_i \), \( U = U - S(J) \ast C \)
\[ = U_\infty - \frac{1}{2} \xi_i - \sum_j \xi_j d_j \]

Now, if \( Y_i = 0 \) (i.e. on the wall) then the \( U_i \) component of velocity \( DX(I) = 0 \). Else, \( DX(I) = U = U_\infty - \frac{1}{2} \xi_i - \sum_j \xi_j d_j \).

Also, if \( X_i > RRI \) (i.e. outside domain of interest) then the \( V_i \) component \( DY(I) = 0 \). Else, \( DY(I) = (G1 - G2)/H \)
\[ = \frac{(U_\infty - I_1) - (U_\infty - I_2)}{h} \]
\[ = \frac{-I_1 + I_2}{h} \]
\[ = V_i \quad \text{by (5a) of (1).} \]

Having calculated \( U_i \) and \( V_i \), we have yet to find \( \eta_i \) before we can take a random step. This random variable \( \eta_i \) is drawn from a gaussian distribution with mean 0 and variance \( \sqrt{4\nu k} \) in the following way:

\( Q1 = \text{RANF}(0.) \)
\( Q2 = \text{RANF}(0.) \) returns a pseudo-random number s.t.
\( 0 < Q1 < 1 \)
\( 0 < Q2 < 1 \)

\( QR = \text{SQRT}(-\text{T SIG} \ast \text{ALOG}(Q1)) \)
\( \text{WALK} = QR \ast \text{SIN}(\text{TPJ} \ast Q2) \)

where \( \eta_i = \text{WALK}. \) For further discussion, see reference [2] pg. 39.
Before actually taking the random step, a few things need to be checked. First, line 234 checks to see if the tags of the successive vortex sheets are the same. If they are, then they are assigned the same \( \eta_i \). Since the tags are ordered linearly, this process is straightforward.

Next, the subroutine checks to see if \( Y_i = 0 \). If it is, then we wish to choose \( \eta_i \) s.t. \( \eta_i \) have different signs each time. This alternating of signs of the random variable \( \eta_i \) is to ensure that the random step, steps across the wall exactly one half of the time. This process of alternating signs is aided by the variable LIP which takes on values \( \pm 1 \). Now, \( Y_i = 0 \) and \( \text{LIP} > 0 \) implies that \( \eta_i > 0 \) the last time. So, jump to line 254 and generate \( \eta_i = \text{WALK} > 0 \) and set \( \text{LIP} < 0 \). If \( Y_i = 0 \) and \( \text{LIP} < 0 \) then generate \( \eta_i = \text{WALK} > 0 \) and so on. This corresponds to lines 239-260 of code.

With the appropriate \( \eta_i \) on hand, the random step is taken in the following manner:

Line 263: \( X(I) = X(I) + DT \ast DX(I) \)

\[
\begin{align*}
X_i^{n+1} &= X_i^n + kU_i \\
\text{which is formula (6a) of [1].}
\end{align*}
\]

Line 265: \( Y(I) = Y(I) + DY(I) \ast DT + \text{WALK} \)

\[
\begin{align*}
Y_i^{n+1} &= Y_i^n + kV_i + \eta_i \\
\text{which is formula (6b) of (1).}
\end{align*}
\]

Finally, if \( Y_i^{n+1} < 0 \) and \( Y_i^n \neq 0 \), then reflect the point by symmetry, setting \( Y_i^{n+1} = -Y_i^{n+1} \). However, if \( Y_i^{n+1} < 0 \) and \( Y_i^n = 0 \) then \( Y_i^{n+1} \) is lost across the wall. This whole process is done for each vortex sheet \( Q_i \), \( i=1, \ldots, N \).
Fig. 8
### OUTPUT FROM PROGRAM BOUNDL

```
<table>
<thead>
<tr>
<th>STEP</th>
<th>TIME</th>
<th>DISP AND RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.004000</td>
<td>1.330000</td>
</tr>
<tr>
<td>2</td>
<td>1.000000</td>
<td>1.330000</td>
</tr>
<tr>
<td>3</td>
<td>1.000000</td>
<td>1.330000</td>
</tr>
<tr>
<td>4</td>
<td>1.000000</td>
<td>1.330000</td>
</tr>
<tr>
<td>5</td>
<td>1.000000</td>
<td>1.330000</td>
</tr>
<tr>
<td>6</td>
<td>1.000000</td>
<td>1.330000</td>
</tr>
</tbody>
</table>

```

### VELOCITY PROFILE

```
<table>
<thead>
<tr>
<th>SHEETS IN SLICE 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP AND RE 0.122</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>STEP</th>
<th>TIME</th>
<th>DISP AND RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.003000</td>
<td>1.7200000</td>
</tr>
<tr>
<td>2</td>
<td>1.000000</td>
<td>1.7200000</td>
</tr>
<tr>
<td>3</td>
<td>1.000000</td>
<td>1.7200000</td>
</tr>
<tr>
<td>4</td>
<td>1.000000</td>
<td>1.7200000</td>
</tr>
<tr>
<td>5</td>
<td>1.000000</td>
<td>1.7200000</td>
</tr>
<tr>
<td>6</td>
<td>1.000000</td>
<td>1.7200000</td>
</tr>
</tbody>
</table>

```

### VELOCITY PROFILE

```
<table>
<thead>
<tr>
<th>SHEETS IN SLICE 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP AND RE 0.064100</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>STEP</th>
<th>TIME</th>
<th>DISP AND RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>2</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>3</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>4</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>5</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>6</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
</tbody>
</table>

```

### VELOCITY PROFILE

```
<table>
<thead>
<tr>
<th>SHEETS IN SLICE 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP AND RE 0.064130</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>STEP</th>
<th>TIME</th>
<th>DISP AND RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>2</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>3</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>4</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>5</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
<tr>
<td>6</td>
<td>1.000000</td>
<td>1.7700000</td>
</tr>
</tbody>
</table>

```

### VELOCITY PROFILE

```
<table>
<thead>
<tr>
<th>SHEETS IN SLICE 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP AND RE 0.064160</td>
</tr>
</tbody>
</table>

```

### VELOCITY PROFILE

```
<table>
<thead>
<tr>
<th>SHEETS IN SLICE 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP AND RE 0.064180</td>
</tr>
</tbody>
</table>
```

---

**Fig. 10**
OUTPUT FORMAT FOR PROGRAM BOUNDL

<table>
<thead>
<tr>
<th>STEP</th>
<th>ISTEP</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>I = 1</td>
<td>X(I)</td>
<td>S(I) = i</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>X(L)</td>
<td></td>
</tr>
</tbody>
</table>

VELOCITY PROFILE

SHEETS IN SLICE JJ

DRAG DRAG DRAG

DISP AND RE DISP RBDL RBDL1
REFERENCES


This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.