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

Author
Cheer, A.Y.

Publication Date
1978-02-01

Supplement to LBL-6443 to appear in
the Journal of Computational Physics

LBL-6443 Supplement
Preprint

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

DISCLAIMER

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 (UXA(I)) for further use by the main routine. Finally, section four documents subroutine STEP. Here, the random numbers η_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_∞ (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	2π
MM	Number of sheets created at each point (X_i) of the x-axis. (The sum of all MM for each X_i $i = 1, \ldots, L$ is equal to IADD where IADD = total number of vortex sheets created at each time step).
CM	ξ_{max} (maximum allowable intensity for each vortex sheet).
LOOK (30,30), LS, MS, HX, XO, YO	Variables used in subroutine DISPL to aid the output formatting of figure 1 of [1].
X(500)	Array storing the X-coordinate X_i
Y(500)	Array storing the Y-coordinate Y_i
S(500)	Array storing ξ_i (intensity of vortex sheet Q_i)
DX(500)	Array storing U_i (first-velocity component of $\vec{U}_i = (U_i, V_i)$)
DY(500)	Array storing V_i (2nd velocity component)

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 MN(I). This tag aids in the assigning of the random variable η_i. Every sheet with the same value MN(I) gets the same η_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.

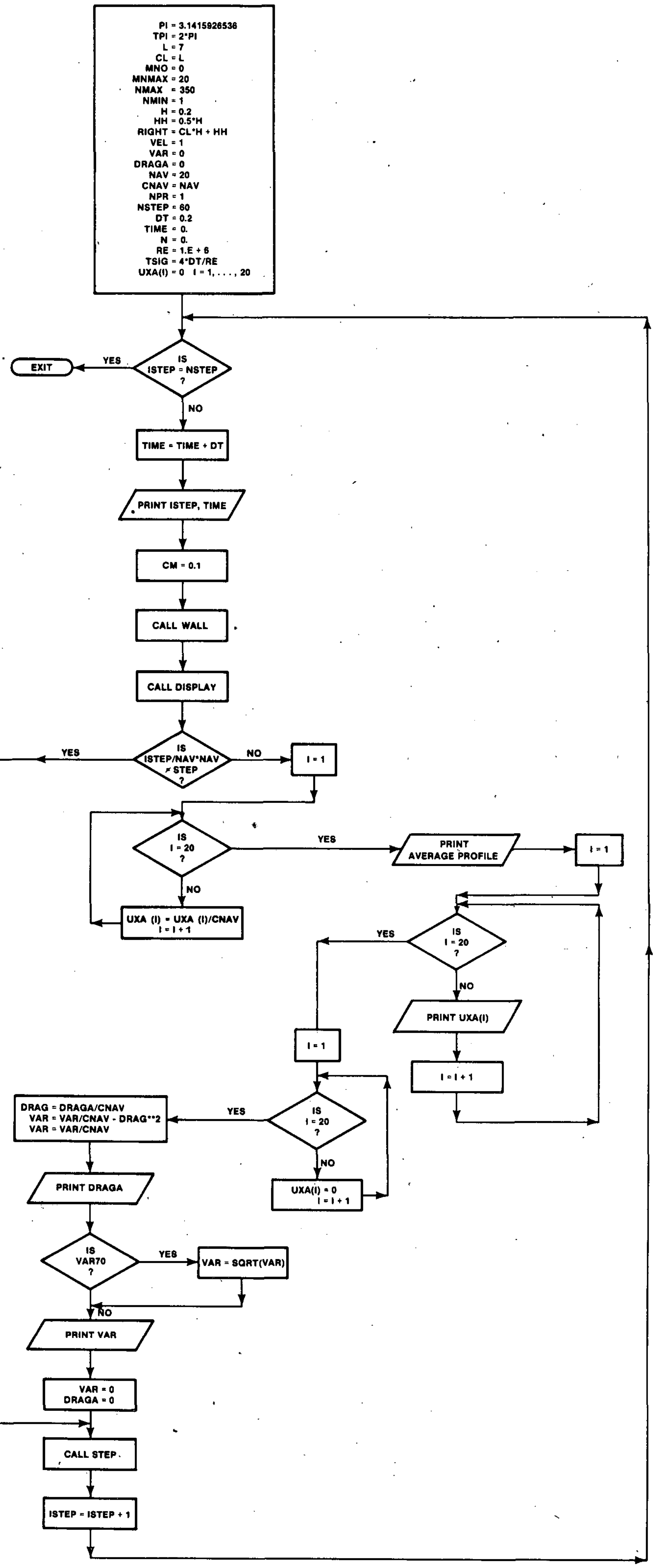
If twenty time steps have not ellapsed, or after averages are computed, BOUNDL calls subroutine STEP to generate random numbers. Each vortex sheet with different tags gets assigned a different random number, and a random step is taken.

Finally, the controlling loop; loop 1; is advanced and the program checks to see if the number of steps already taken is $\leqslant$ NSTEP. If yes, then the above process is repeated. If no, the program exits and execution halts.

```
          PROGRAMBOUNDL(INFUT,CUTFUT)
          COMMONH,HH,N,L,VEL,RE,TSIG,CT,TPI,MM,CM
        1 ,LOCK(30,30),LS,MS,HX,HY,X0,Y0
        1 ,X(5CC),Y(5CC),S(5CC),CX(500),CY(500)
        1 ,UXA(100),DRAGA,VAR,RIGHT,NOLD
        1 ,MN(500),MNO,MNMAX
C
C          INITIAL CALCULATICNS AND SETTINGS
           PI=3.1415926536
          TPI=2.*PI
          L=10
           L=12
          L=6
           L=13
          L=7
          CL=L
          MN0=0.
          MNMAX=20
          NMAX=350
          NMIN=1
          H=0.1
          H=0.2
          HH=0.5*H
          RIGHT=CL*H+HH
           VEL=1.
          VAR=0.
          DRAGA=C.
          NAV=5
          NAV=20
          CNAV=NAV
          NPR=1
          NSTEP=3
          NSTEP=20
          NSTEP=10
           NSTEP=30
          NSTEP=60
          DT=0.1
          CT=0.2
          TIME=0.
          N=0
          RE=1.E+4
          RE=1.E+6
          TSIG=4.*CT/RE
          DC2I=1,20
          UXA(I)=0.
     2    CONTINUE
C
C         TIME STEP
          DO1ISTEP=1,NSTEP
          TIME=TIME+DT
          PRINT9000
          PRINT9001,ISTEP,TIME
          CM=0.2
          CM=0.1
          CALL WALL
          CALL DISPL
          IF(((ISTEP/NAV)*NAV).NE.ISTEF)GCTO4
          DO5I=1,20
     5    UXA(I)=UXA(I)/CNAV
          PRINT9C04
          DO6I=1,20
     6    PRINT9002,UXA(I)
          DO3I=1,20
           UXA(I)=0.
     3    CONTINUE
          DRAGA=DRAGA/CNAV
          PRINT9003,DRAGA
          VAR=VAR/CNAV-DRAGA**2
          VAR=VAR/CNAV
          IF(VAR.GT.C.)VAR=SQRT(VAR)
          PRINT9005,VAR
          VAR=0.
           DRAGA=C.
     4    CONTINUE
          CALL STEP
     1    CONTINUE
          CALL EXIT
  9000    FORMAT(/)
  9001    FORMAT(* STEP*,I5,* TIME*,F11.7)
  9002    FORMAT(1X,EF9.5)
  9003    FORMAT(* AVERAGE DRAG*,F11.7)
  9004    FORMAT(* AVERAGE PROFILE*)
  9005    FORMAT(* VARIANCE*,F11.7)
          END
```

Fig. 1

PROGRAM BOUNDL



XBL 783-514

2. Subroutine WALL

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

EPS 0.5 * CM = 1/2 ξ_{max} (this is the maximum allowable intensity of each vortex sheet for this run)

SS used to calculate U_i

IADD Counter for the number of vortex sheets created on this call.

For each j=1,...,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 = 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 $\qquad j=1,\ldots,N$

$$= U_\infty - \sum_{j=1}^{N} \xi_j \, d_j$$

where SS is initialized to U_∞, C = d_j and S(J) = ξ_j. This quantity SS also corresponds to U_o on pg. 8 of [1].

Now that the value for U_o is known, line 100 of the code checks to see if $|2U_o| <$ EPS ($=1/2\ \xi_{max}$). 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., $X_{i+1} <$ RIGHT. If $|2U_o| >$ EPS, then it breaks $2U_o$ into an even number of vortex sheets, MM or DIV, each with the same intensity SS/DIV.

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

This will ensure that brand new tags will be used at the next time step.

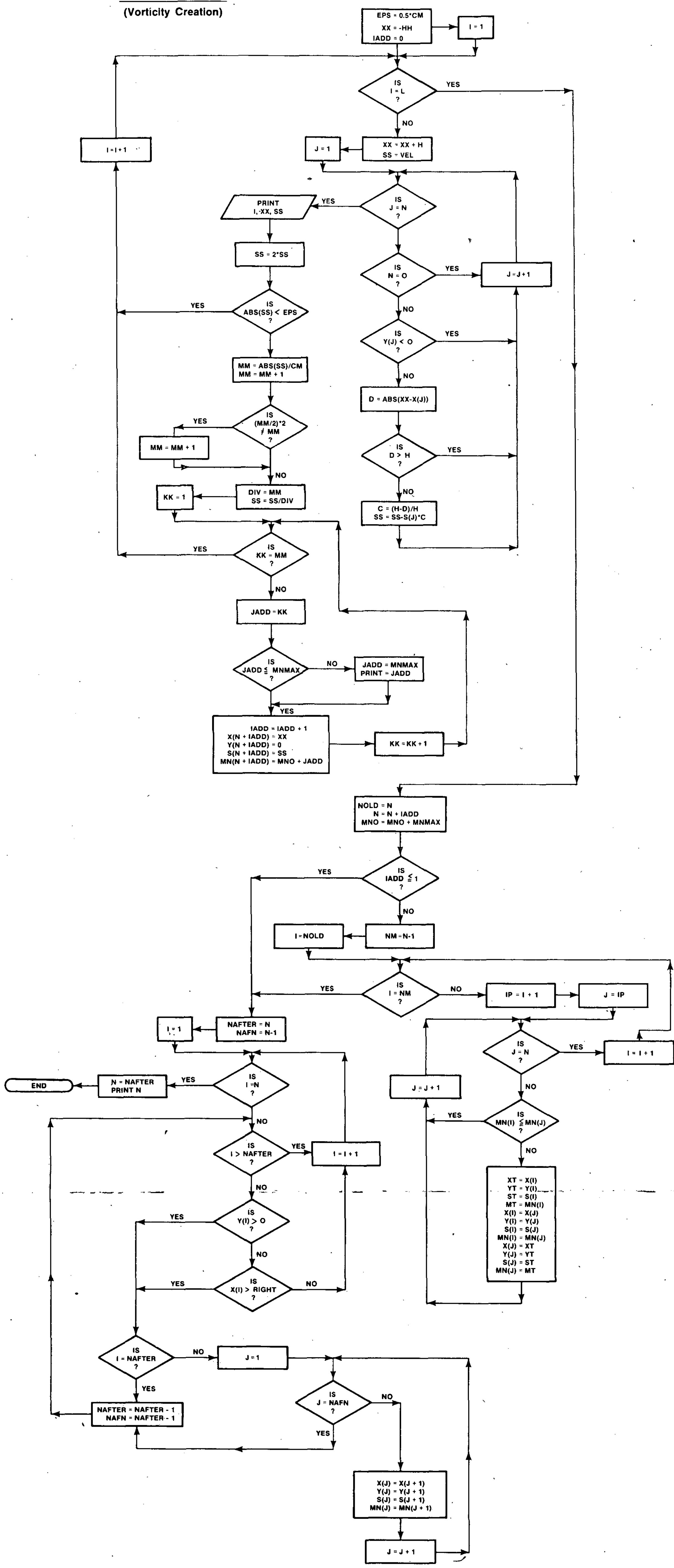
Loop 16 and 17 now takes over the task of sorting the sheets created above according to the values of their tag MN(I). This sort is done to facilitate the assignment of the random number η_i in subroutine STEP, where the same η_i is assigned to sheets with the same tag MN(I).

Now, for I = 1, ..., N, loop 9 checks to see if (X(I),Y(I)) is in the domain of interest. i.e., X(I) < 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.

```
      SUBROUTINE WALL
      COMMCNH,HH,N,L,VEL,FE,TSIG,CT,TPI,MM,CM
     1 ,LOOK(30,30),LS,MS,HX,HY,X0,Y0
     1 ,X(500),Y(500),S(500),CX(500),CY(500)
     1 ,UXA(1CC),DRAGA,VAR,RIGHT,NCLD
     1 ,MN(500),MN0,MNMAX
C
      EPS=0.5*CM
C     VORTICITY CREATICN
      XX=-HH
      IACC=0
      DO1 I=1,L
      XX=XX+H
      SS=VEL
      DO2J=1,N
      IF(N.EQ.0)GOTC2
      IF(Y(J).LT.0.)GCTC2
      D=ABS(XX-X(J))
      IF(D.GT.H)GCTC2
      C=(H-D)/H
      SS=SS-S(J)
     1 *C
    2 CONTINUE
      PRINT9006, I,XX,SS
      SS=2.*SS
      IF(ABS(SS).LT.EPS)GOTO3
      MM=(ABS(SS)/CM)
      MM=MM+1
      IF(((MM/2)*2).NE.MM)MM=MM+1
      DIV=MM
      SS=SS/DIV
      DC4 KK=1, MM
      JADD=KK
      IF(JACC.LE.MNMAX)GOTO15
      JADD=MNMAX
      PRINT9007, JADD
 9007 FORMAT(* TCO FEW SLICES*, I6)
   15 CONTINUE
      IACC= IADD+1
      X(N+IADD)=XX
      Y(N+IADD)=0.
      S(N+IADD)=SS
      MN(N+IADD)=MNC+JADD
    4 CONTINUE
    3 CONTINUE
    1 CONTINUE
      NCLD=N
      N=N+IADD
      MN0=MN0+MNMAX
      IF(IACC.LE.1)GCTC18
      NM=N-1
      DC16 I=NCLD,NM
      IP=I+1
      DO17J= IP,N
      IF(MN(I).LE.MN(J))GCTC17
      XT=X(I)
      YT=Y(I)
       ST=S(I)
      MT=MN(I)
      X(I)=X(J)
       Y(I)=Y(J)
      S(I)=S(J)
      MN(I)=MN(J)
      X(J)=XT
      Y(J)=YT
       S(J)=ST
      MN(J)=MT
   17 CONTINUE
   16 CONTINUE
   18 CONTINUE
C
C     PURGE
      NAFTER=N
      NAFN=N-1
      DO9 I=1,N
   12 CONTINUE
      IF(I.GT.NAFTER)GOTO9
       IF(Y(I).LT.0.)GCTC14
      IF(X(I).GT.RIGHT)GOTO14
      GCTO9
   14 CONTINUE
      IF(I.EQ.NAFTER)GOTO13
      DO10J=I,NAFN
      X(J)=X(J+1)
      Y(J)=Y(J+1)
      S(J)=S(J+1)
      MN(J)=MN(J+1)
   10 CONTINUE
   13 CONTINUE
      NAFTER=NAFTER-1
      NAFN=NAFTER-1
      GOTO12
    9 CCNTINUE
      N=NAFTER
      PRINT9004,N
 9004 FORMAT(* N*,I5)
 9006 FORMAT(1X, I5,3F11.7)
      RETURN
      END
```

Fig. 3

SUBROUTINE WALL
(Vorticity Creation)



XBL 783-515

3. <u>Subroutine DISPL</u>

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:

JPR	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].
ETA	Similarity variable.
DRAGR	Real drag (for flow past a semi-infinite flat plate)
DRAG	Drag computed by this program
DISP	Displacement thickness
RBDL	Boundary Layer (computed)
RBDL1	Real boundary layer used for comparison.

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 $\neq$ 1.

Next, loop 40 initializes array UX(I) to U_∞ for I = 1, ..., N. Then, loop 4 calculates

$$U_j = U_j - \sum_{i=1}^{N} \xi_i d_i \qquad j = 1,\ldots, ku$$

for $|X - X_i| < h$.

Here, ku $\leqslant$ 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 which are used in plotting figure 2 of [1]. If JPR = 1, 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(JJ) = 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 $Y_1 \leqslant Y_2 \leqslant \ldots \leqslant Y_m$.

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,...,JJ, where JJ = min(N,100) = M.

(i) Line 389 of code:

```
DRAG = DRAG + U * (VEL - U) * 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:

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

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

Now, the real drag DRAGR and the computed drag DRAG are computed and printed. Also, the real and computed boundary layer, RBDL1 and RBDL respectively, are outputted. Finally, variables DRAGA and VAR are incremented and stored away for used by the main routine.

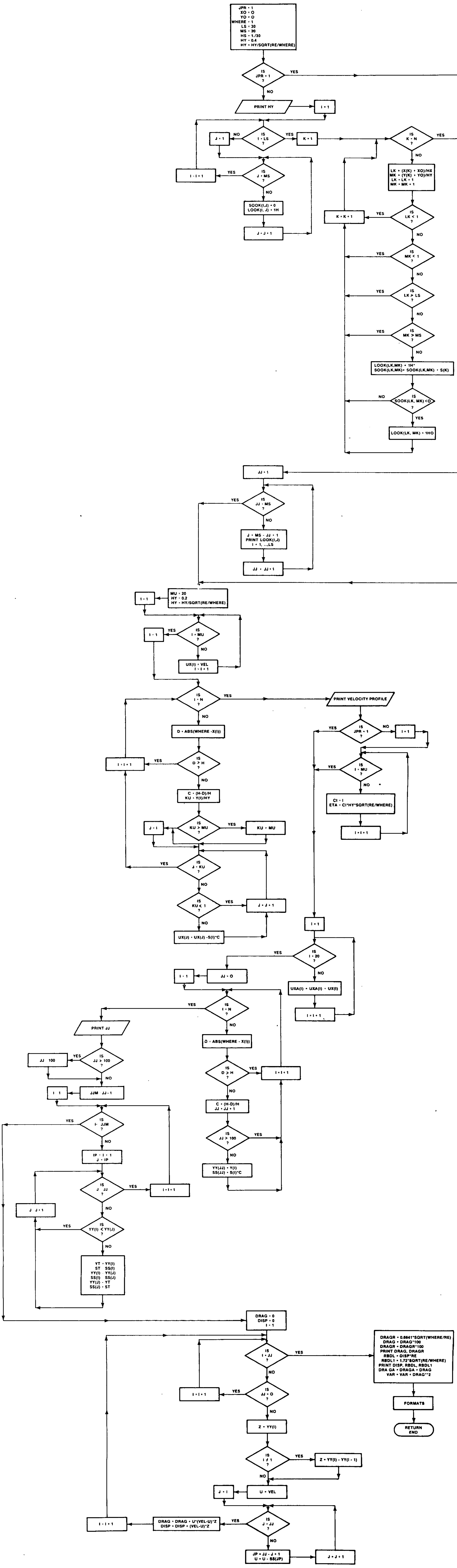
```
      SUBFCUTINE CISPL
      COMMONH,HH,N,L,VEL,RE,TSIG,CT,TPI,NN,CN
     1 ,LOOK(30,30),LS,MS,HX,HY,X0,Y0
     1 ,X(500),Y(500),S(500),CX(500),CY(500)
     1 ,UXA(100),DRAGA,VAR,RIGHT,NOLD
     1 ,MN(500),MN0,MNMAX
     1 ,SOOK(30,30)
       DIMENSIONUX(100),YY(100),SS(100)
      JPR=1
      X0=0.
      Y0=0.
      WHERE=1.
      LS=30
      MS=30
      HX=1./30.
      HY=0.4
      HY=HY/SQRT(RE/WHERE)
       IF(JPR.EQ.1)GOTO20
      PRINT9008,HY
C.
       DO1I=1,LS
      DO1J=1,MS
      SOOK(I,J)=0.
    1 LOOK(I,J)=1H
       DO2K=1,N
      LK=(X(K)+X0)/HX
      MK=(Y(K)+Y0)/HY
      LK=LK+1
      MK=MK+1
      IF(LK.LT.1)GOTO2
      IF(MK.LT.1)GOTO2
      IF(LK.GT.LS)GOTO2
      IF(MK.GT.MS)GOTO2
      LOOK(LK,MK)=1H*
      SOOK(LK,MK)=SOOK(LK,MK)+S(K)
      IF(SOOK(LK,MK).LT.0.)LOOK(LK,MK)=1HO
    2 CONTINUE
      DO3JJ=1,MS
      J=MS-JJ+1
    3 PRINT9004,(LOOK(I,J),I=1,LS)
   20 CONTINUE
C
       MU=20
      HY=0.2
      HY=HY/SQRT(RE/WHERE)
C     VELOCITY PROFILE
      DO40I=1,MU
   40 UX(I)=VEL
      DO4I=1,N
      D=ABS(WHERE-X(I))
      IF(D.GT.H)GOTO4
      C=(H-D)/H
      KU=Y(I)/HY
      IF(KU.GT.MU)KU=MU
      DO5J=1,KU
      IF(KU.LT.1)GOTO5
      UX(J)=UX(J)-S(I)
     1 *C
    5 CONTINUE
    4 CONTINUE
      PRINT9002
       IF(JPR.EQ.1)GOTO21
      DO6I=1,MU
      CI=I
      ETA=CI*HY*SQRT(RE/WHERE)
    6 PRINT9001,UX(I)
     1 ,ETA
   21 CONTINUE
      DO14I=1,20
   14 UXA(I)=UXA(I)+UX(I)
C
```

Fig. 5

```
C       DRAG
        JJ=0
        DO7I=1,N
        D=ABS(WHERE-X(I))
        IF(D.GT.H)GOTO7
        C=(H-D)/H
        JJ=JJ+1
        IF(JJ.GT.100)GOTO7
        YY(JJ)=Y(I)
        SS(JJ)=S(I)
        C
  7     CONTINUE
        PRINT9005,JJ
        IF(JJ.GT.100)JJ=100
        JJM=JJ-1
        DO10I=1,JJM
        IP=I+1
        DO11J=IP,JJ
        IF(YY(I).LT.YY(J))GOTO11
        YT=YY(I)
        ST=SS(I)
        YY(I)=YY(J)
        SS(I)=SS(J)
        YY(J)=YT
        SS(J)=ST
 11     CONTINUE
 10     CONTINUE
        DRAG=0.
        DISP=0.
        DO12I=1,JJ
        IF(JJ.EQ.0)GOTO12
        Z=YY(I)
        IF(I.NE.1)Z=YY(I)-YY(I-1)
        U=VEL
        DO13J=I,JJ
        JP=JJ-J+I
        U=U-SS(JP)
 13     CONTINUE
        DRAG=DRAG+U*(VEL-U)*Z
        DISP=DISP+(VEL-U)*Z
 12     CONTINUE
        DRAGR=0.6641*SQRT(WHERE/RE)
        DRAG=DRAG*100.
        DRAGR=DRAGR*100.
        PRINT9007,DRAG,DRAGR
        RBDL=DISP*RE
        RBDL1=1.72*SQRT(RE*WHERE)
        PRINT9009,DISP,RBDL,RBDL1
        DRAGA=DRAGA+DRAG
        VAR=VAR+DRAG**2
 9000   FORMAT(/)
 9001   FORMAT(1X,3F11.7)
 9002   FORMAT(* VELOCITY PROFILE*)
 9004   FORMAT(1X,60A1)
 9005   FORMAT(* SHEETS IN SLICE*,I5)
 9006   FORMAT(1X,F11.7)
 9007   FORMAT(* DRAG*,2F11.7)
 9008   FORMAT(* HY*,F11.7)
 9009   FORMAT(* DISP AND RE*,3F15.7)
        RETURN
        END
```

Fig. 6

SUBROUTINE DISPL



XBL 783-517

4. Subroutine STEP

First, this subroutine creates the random numbers η_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 $\vec{U}_i = (U_i, V_i)$. So, for each vortex sheet Q_i, i=1, ..., 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 = 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 199: $D \leqslant H \Rightarrow 0 \leqslant d_j^+ \leqslant 1$

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

$$= \frac{h - \left| -(X_i - X_j + \frac{h}{2}) \right|}{h}$$

$$= 1 - \frac{\left| X_i - X_j + \frac{h}{2} \right|}{h}$$

$$= d_j^+$$

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

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

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

$= Y_j^*$ the displacement

5) line 205: G1 = G1 + S(J) * YY * C

$$= \sum_{j^+} \xi_j \, Y_j^* d_j^+$$

$$= U_\infty - I_1$$

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

1) line 207: D = 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 $\Rightarrow$ $0 \leqslant d_j^- \leqslant 1$

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

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

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

$$= d_j^-$$ formula (5e) of [1].

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

5) line 214: G2 = G2 + S(J) * YY * C

$$= \sum_{j-} \xi_j \, Y_j^* d_j^-$$

$$= U_\infty - I_2$$ 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 = Vel $\Rightarrow$ $U = U_\infty$

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

$$= U_\infty - \frac{1}{2} \xi_i$$

3) line 216: D = 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

$$= 1 - \frac{|X_i - X_j|}{h}$$

$$= d_j \qquad \text{by (4b) of [1].}$$

6) line 220: For $Y_j > Y_i$, U = U - S(J) * 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

$$= \frac{(U_\infty - I_1) - (U_\infty - I_2)}{h}$$

$$= \frac{-I_1 + I_2}{h}$$

$$= V_i \qquad \text{by (5a) of (1).}$$

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

```
Q1 = RANF(O.)          RANF(O.) returns a pseudo-randon number s.t.
Q2 = RANF(O.)          0 < Q1 < 1
                       0 < Q2 < 1

QR = SQRT(-TSIG * ALOG(Q1))
WALK = QR * SIN(TPI * Q2)
```

where η_i = 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 η_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 η_i s.t. η_i have different signs each time. This alternating of signs of the random variable η_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 ± 1. Now, $Y_i = 0$ and LIP > 0 implies that $\eta_i > 0$ the last time. So, jump to line 254 and generate η_i = WALK > 0 and set LIP < 0. If $Y_i = 0$ and LIP < 0 then generate η_i = WALK > 0 and so on. This corresponds to lines 239-260 of code.

With the appropriate η_i on hand, the random step is taken in the following manner:

Line 263: X(I) = X(I) + DT * DX(I)

$$X_i^{n+1} = X_i^n + kU_i$$

which is formula (6a) of [1].

Line 265: Y(I) = Y(I) + DY(I) * DT + WALK

$$Y_i^{n+1} = Y_i^n + kV_i + \eta_i$$

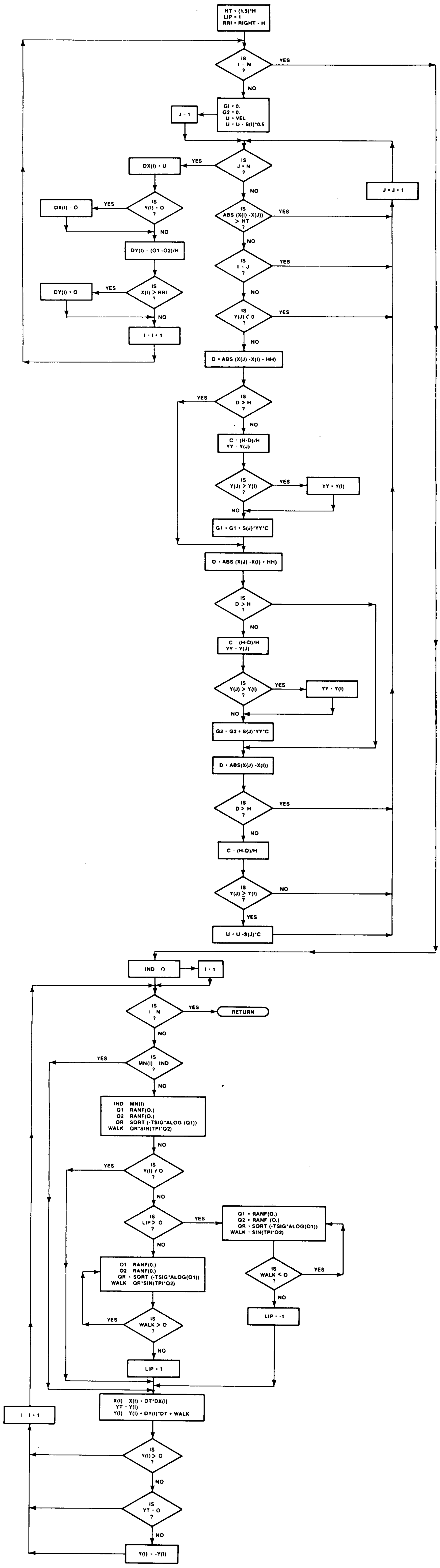
which is formula (6b) of (1).

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, ..., N.

```
      SUBROUTINE STEP
      COMMONH,HH,N,L,VEL,FE,TSIG,CT,TPI,MM,CM
     1 ,LOOK(30,30),LS,MS,HX,HY,X0,Y0
     1 ,X(500),Y(500),S(500),DX(500),DY(500)
     1 ,UXA(100),DRAGA,VAR,RIGHT,NOLD
     1 ,MN(500),MNO,MNMAX
      HT=1.5*H
      LIP=1
      RRI=RIGHT-H
      DO1I=1,N
      G1=0.
      G2=0.
      U=VEL
      U=U-S(I)*0.5
      DO2J=1,N
      IF(ABS(X(I)-X(J)).GT.HT)GOTO2
      IF(I.EQ.J)GOTO2
      IF(Y(J).LT.0.)GOTO2
      D=ABS(X(J)-X(I)-HH)
      IF(D.GT.H)GOTO3
      C=(H-D)/H
      YY=Y(J)
      IF(Y(J).GT.Y(I))YY=Y(I)
      G1=G1+S(J)*YY
     1 *C
    3 CONTINUE
      D=ABS(X(J)-X(I)+HH)
      IF(D.GT.H)GOTO4
      C=(H-D)/H
       YY=Y(J)
      IF(Y(J).GT.Y(I))YY=Y(I)
      G2=G2+S(J)*YY
     1 *C
    4 CONTINUE
      D=ABS(X(J)-X(I))
      IF(D.GT.H)GOTO2
      C=(H-D)/H
       IF(Y(J).GE.Y(I))U=U-S(J)*C
    2 CONTINUE
      DX(I)=U
      IF(Y(I).EQ.0.)DX(I)=0.
      DY(I)=(G1-G2)/H
      IF(X(I).GT.RRI)DY(I)=0.
    1 CONTINUE
      IND=0
      DO5I=1,N
      IF(MN(I).EQ.IND)GOTO6
      IND=MN(I)
      Q1=RANF(0.)
      Q2=RANF(0.)
      QR=SQRT(-TSIG*ALOG(Q1))
      WALK=QR*SIN(TPI*Q2)
      IF(Y(I).NE.0.)GOTO8
      IF(LIP.GT.0)GOTO9
   10 CONTINUE
      Q1=RANF(0.)
      Q2=RANF(0.)
      QR=SQRT(-TSIG*ALOG(Q1))
      WALK=QR*SIN(TPI*Q2)
      IF(WALK.GT.0.)GOTO10
      LIP=1
      GOTO8
    9 CONTINUE
   11 CONTINUE
      Q1=RANF(0.)
      Q2=RANF(0.)
      QR=SQRT(-TSIG*ALOG(Q1))
      WALK=QR*SIN(TPI*Q2)
      IF(WALK.LT.0.)GOTO11
      LIP=-1
    8 CONTINUE
    6 CONTINUE
      X(I)=X(I)+DT*DX(I)
      YT=Y(I)
      Y(I)=Y(I)+DY(I)*DT
     1 +WALK
      IF(Y(I).GT.0.) GOTO7
      IF(YT.EQ.0.)GOTO7
      Y(I)=-Y(I)
    7 CONTINUE
    5 CONTINUE
 9001 FORMAT(1X,10F9.5)
 9002 FORMAT(1X,2I5,5F9.5)
      RETURN
      END
```

Fig. 8

Subroutine STEP



XBL 783-516

OUTPUT FROM PROGRAM BOUNDL

```
STEP    1 TIME   .2000000
     1   .1000000   1.0000000
     2   .3000000   1.0000000
     3   .5000000   1.0000000
     4   .7000000   1.0000000
     5   .9000000   1.0000000
     6  1.1000000   1.0000000
     7  1.3000000   1.0000000
N  140
VELOCITY PROFILE
SHEETS IN SLICE     40
DRAG    0.           .0664100
DISP AND RE        0.              0.            1720.0000000

STEP    2 TIME   .4000000
     1   .1000000   -.0000000
     2   .3000000   -.0000000
     3   .5000000   -.0000000
     4   .7000000   -.0000000
     5   .9000000   -.0000000
     6  1.1000000   -.0000000
     7  1.3000000   -.0000000
N   70
VELOCITY PROFILE
SHEETS IN SLICE     20
DRAG    .0181800     .0664100
DISP AND RE          .0000035     503.4693629    1720.0000000

STEP    3 TIME   .6000000
     1   .1000000    .0000000
     2   .3000000   -.0000000
     3   .5000000   -.0000000
     4   .7000000   -.0000000
     5   .9000000   -.0000000
     6  1.1000000   -.0000000
     7  1.3000000   -.0000000
N   80
VELOCITY PROFILE
SHEETS IN SLICE     20
DRAG    .0304548     .0664100
DISP AND RE          .0007364     736.3841169    1720.0000000

STEP   20 TIME  4.0000000
     1   .1000000    .4276434
     2   .3000000    .0713727
     3   .5000000    .0494275
     4   .7000000   -.0091353
     5   .9000000    .0512212
     6  1.1000000   -.0599686
     7  1.3000000    .0566237
N  144
VELOCITY PROFILE
SHEETS IN SLICE     47
DRAG    .0651838     .0664100
DISP AND RE          .0024371    2437.1317559    1720.0000000
AVERAGE PROFILE
   .13770
   .28023
   .37104
   .47500
   .55844
   .60854
   .67399
   .74609
   .78223
   .81019
   .86073
   .87980
   .89894
   .91307
   .93119
   .94748
   .95074
   .95789
   .96810
   .97418
AVERAGE DRAG    .0420176
VARIANCE    .0042745
```

Fig. 10

OUPUT FORMAT FOR PROGRAM BOUNDL

STEP	ISTEP TIME	TIME
I = 1	X(I)	S(I) = i
2	.	.
.	.	.
.	.	.
.	.	.
L	X(L)	

N

VELOCITY PROFILE

SHEETS IN SLICE JJ

DRAG DRAG DRAGR

DISP AND RE DISP RBDL RBDL1

REFERENCES

[1]: Chorin, A. J., Vortex Sheet Approximation of Boundary Layers, J. Comput. Physics, 1978.

[2]: J. M. Hammersley and D. C. Hamdscomb, Monte Carlo Methods, Methuen, London 1964.

GPO 971-331

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.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720