NASA Technical Memorandum TM - 78665

A Vectorization of the Jameson-Caughey NYU Transonic Swept-Wing Computer Program FLO-22-VI for the STAR-100 Computer

Robert E. Smith, Joan I. Pitts, and Jules J. Lambiotte
Langley Research Center

March 1978

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

A VECTORIZATION OF THE JAMESON-CAUGHEY NYU TRANSONIC SWEPT-WING COMPUTER PROGRAM FLO-22-V1 FOR THE STAR-100 COMPUTER

R. E. Smith, J. I. Pitts, and J. J. Lambiotte
Langley Research Center

SUMMARY

The computer program FLO-22 for analyzing inviscid isentropic transonic flow past 3-D swept-wing configurations has been modified to use vector operations and run on the STAR-100 computer. The "vectorized version" described herein is called FLO-22-V1. Vector operations have been incorporated into Successive Line Over-Relaxation in the transformed horizontal direction. Vector relational operations and control vectors are used to implement upwind differencing at supersonic points. A high speed of computation and extended grid domain are characteristics of FLO-22-V1. The new program is not the optimal vectorization of Successive Line Over-Relaxation applied to transonic flow; however, it proves that vector operations can readily be implemented to increase the computation rate of the algorithm.

INTRODUCTION

The program FLO-22 has been developed for analyzing inviscid isentropic transonic flow past 3-D swept-wing configurations (ref. 1, 2, and 3). The program originators are Prof. Antony Jameson (New York University) and Prof. Dave Caughey (Cornell University) who developed the program under NASA grants NGR-33-016-167 and NRG-33-016-201 at the Courant Institute of Mathematical Sciencies of NYU. The program is designed to run on third generation computers

such as the CDC 6600, and CYBER 175. FLO-22-V1 is a first attempt to design FLO-22 to run efficiently on the STAR-100 computer at the Langley Research Center. The objectives have been to decrease the computational run time and to increase the grid domain.

The STAR-100 is a vector processing computer capable of achieving high result rates when a high degree of parallelism is present in the computation. The STAR-100 operates on vectors, most simply thought of as arrays in memory, as opposed to scalars. When an identical operation is to be performed on consecutive elements in memory (i.e. $c_i=a_i*b_i$ i=1,2,...N), a vector instruction is issued to perform the operations. Each vector instruction involves a time penalty, called vector start-up, regardless of the length N of the operation. As N increases in size, the operation becomes more efficient since the penalty becomes relatively less important. Another penalty that generally occurs is the necessity for additional storage for temporary vectors instead of temporary scalars.

The goal in designing (or vectorizing) a STAR program is to induce long vector operations wherever possible. The STAR-100 at the Langley Research Center has one half million words of primary memory with virtual memory architecture. The operating system automatically brings from secondary memory any piece of data which is referred to by the program but does not currently reside in primary memory. Hence, overlay or buffering procedures often coded into large data base programs can be avoided. More detailed information relative to the STAR-100 computer may be found in references 4 and 5.

The algorithm used in FLO-22 is the Successive Line Over-Relaxation technique (SLOR) applied with the Murman and Cole differencing scheme (ref. 2) to the full

three-dimensional transonic potential equation. There is a coordinate transformation (ref. 2) from the physical domain to the computational domain which further complicates the nonlinear partial differential equation, but simplifies the handling of the boundary conditions. Basic in the numerical procedure of SLOR is the solution of tridiagonal systems of linear algebraic equations along lines of the computational grid. In FLO-22 there is the option to use lines in the transformed horizontal direction (YSWEEP) or in the transformed normal direction (XSWEEP). Also FLO-22 has the option for grid refinement in the solution process. The first step is the computing of a solution over a coarse grid. Refined solutions are obtained by calling subroutine REFIN which halves the coarse grid and interpolates the coarse grid solution for initial guesses for the refined grid. For wing configurations typical of transport aircraft, the SLOR is normally applied in the horizontal direction using subroutine YSWEEP with the input variable STRIP=1. It is this routine in which the major revisions have been made to create FLO-22-V1. The other major revision is the direct addressing of the variable G(I,J,K) which contains the potential at all the grid points for any given iteration. FLO-22 has been designed to have four spanwise grid planes of the potential solution in primary memory at one time. After the correction to the potential on the Kth plane has been performed, the Kth+2 plane is buffered from secondary memory into primary memory and the Kth-2 plane is buffered out. Since the STAR-100 has virtual memory, FLO-22-V1 has been designed to address all variables as being in primary memory. This has required changes in several subroutines but primarily MIXFLO and YSWEEP. Other changes have been made but do not strongly affect the computational speed.

As stated before, the solution of tridiagonal systems of linear algebraic equations is the heart of SLOR technique. Typically on the finest grid using YSWEEP, over five hundred tridiagonal systems containing 192 linear equations are formed and solved per iteration. The efficient solution of a system of tridiagonal linear equations is a recursive process and is not in itself vectorizable. In addition, since the systems arising from the SLOR method are interdependent, they must be solved one at a time. However, in applying SLOR to the transformed nonlinear potential equation, the major computational activity is the forming of the coefficient matrix and the right hand side of the tridiagonal system. These computations are vectorizable, and if the number of equations is sufficiently large, the STAR-100 can be efficiently employed. Since the largest number of equations are created by sweeping in the horizontal direction, the vectorization of YSWEEP is a logical starting point. In FLO-22-V1 the coefficient matrix and right hand side of the tridiagonal systems are formed along lines in the transformed horizontal direction using vector operations and an efficient machine coded scalar tridiagonal solution is performed to obtain the correction to the potential. Vector lengths are equal to the number of grid points in the horizontal direction.

A particular problem in vectorizing a program for the solution of transonic potential flow is the necessary test for supersonic points and the subsequent computation of the upwind differences in the Murman and Cole scheme. Testing for supersonic points and modifying the computation inline with a vectorized operation for the subsonic points is impossible.

The problem is overcome at some computational expense by performing the supersonic computation with vector operations for all points and by using a

control vector to apply the computation only at those points where the flow is supersonic.

The conversion of FLO-22 is complete and several cases have been run on the STAR-100 computer and the solution times compared with those obtained on the CYBER 175. The maximum grid size in FLO-22-V1 has been expanded to 300x32x32 as compared with 192x24x32 in FLO-22. No serious problems have been encountered.

This report describes the major routines where changes have been made to vectorize FLO-22 to create FLO-22-V1. Comparison of computational times are presented. It should be noted that FLO-22-V1 is not the optimal vectorization of transonic potential flow calculations using SLOR. There is the possibility of generating two-dimensional vector operations of parts of the computation in each spanwise plane as opposed to each line of the plane. This would give rise to longer vectors and higher efficiency.

VECTORIZATION AND PERFORMANCE EVALUATION OF SLOR IN FLO-22-V1

The application of SLOR to transonic flow calculations is throughly discussed in references 1, 2, and 3, and is not presented herein. Instead, the functional operation of the algorithm in FLO-22 and FLO-22-V1 is contrasted. The general organization of the program is shown in Figure 1. SLOR is an iterative process, and in FLO-22 and FLO-22-V1, each iteration is under the control of the main program. After initialization, the main program calls subroutine MIXFLO for each iteration. For each plane of grid points in the spanwise direction (Figure 2), MIXFLO exercises the option for relaxation along horizontal lines using subroutine YSWEEP or along vertical lines using subroutine XSWEEP. XSWEEP has not been vectorized in FLO-22-V1 and, therefore, will receive no further discussion. Cases

requiring XSWEEP will not run with vector operations in FLO-22-V1. Subroutine MIXFLO, in addition to exercising the direction of relaxation, performs the setup of boundary conditions and in FLO-22 buffers the grid planes of the potential in and out of primary memory. In FLO-22-V1 the potential at any grid point is directly addressable.

Subroutine YSWEEP performs Successive Line Over-Relaxation along each horizontal line of the Kth spanwise plane of the grid (see Figure 2). This routine has been chosen for vectorization for several reasons. First, FLO-22 is predominately used at LaRC for transport type wings which require a relatively large number of points in the X or horizontal direction, and do not require the XSWEEP option. Second, even though the subroutine requires the solution to many dependent sets of tridiagonal equations, which is basically scalar in nature, it also performs transformation evaluation, PDE coefficient evaluation, finite difference approximations and applies the solution of the tridiagonal systems to the potential along each line. Except for the tridiagonal solutions, all of the computations are vectorizable. Since the transonic potential flow equation is nonlinear and transformed from a physical domain to a computational domain, the vectorizable part of the subroutine performs the majority of the computation. When the computation for each line on the Kth plane is complete, control is returned to MIXFLO. The Murman and Cole difference scheme (ref. 2 and 3) is implemented in YSWEEP. The scheme changes the difference approximation to upwind if the flow is supersonic at a grid point. Central difference approximations are used for subsonic points. Programmatically, the upwind differencing is applied differently in FLO-22 and FLO-22-V1. This will be explained further at a later point.

In FLO–22, YSWEEP primarily consists of two FORTRAN DO loops. The first loop performs the evaluation of all variables through the forward substitution in the tridiagonal solution. The second DO loop performs the backward substitution for the correction to the potential and adds the correction to the potential G(I,J,K) along the line. A conditional GO TO sends the control of the program to the beginning of the first loop to move through each line of the plane. Other peripheral computations are performed but are not discussed here. The application of upwind differencing in the Murman and Cole scheme is in the form of a comparison of the speed of sound and the latest velocity approximation with a branch to a section of the subroutine for upwind differencing if the flow is supersonic. This is performed in the first DO loop for each grid point along the line. Appendix A shows a listing of the scalar or unvectorized YSWEEP.

Appendix B shows a listing of the vectorized YSWEEP found in FLO–22–V1. Vector FORTRAN instructions and an efficient assembly coded tridiagonal equation solver replace the two major DO loops found in FLO–22 and discussed above. Using vector operations, arrays replace scalar variables. In the unvectorized program all variables for a grid point are computed before going to the next grid point. In contrast, a variable in the vectorized program is computed for all grid points before going to the next variable. Storage requirements are therefore higher in the vectorized program. In the vectorized YSWEEP the computation is not carried through the forward substitution in the solution of the tridiagonal equations. Instead, the subdiagonal, superdiagonal, and right hand side arrays are computed with vector FORTRAN instructions and stored. This is followed by calling a mathematical library subroutine Q4TRIDGE (ref. 5) for the solution of the tridiagonal system. The solution, which is the correction

to the potential at grid points along the line, is added to the potential from the previous iteration. In FLO-22 the test for supersonic points and application of upwind differencing is done within the first major DO loop. This computation is done for supersonic points only, but this type of testing and branching is not feasible for vector computation. In FLO-22-V1 a vector relational operator establishes a control vector based on the comparison of the latest velocity approximation and speed of sound. The control vector is a bit string with a "one" stored when the test is true and a "zero" stored when the test is false. It is done for all points in FLO-22-V1. The control vector is used to decide whether or not to apply the upwind computation in the tridiagonal system. There is some loss of efficiency due to the need for extra computations which are not used, but it is considerably faster than reverting to scalar computation. A control vector is also used to apply the proper direction of the upwind differencing along a horizontal line of the computational domain.

The control in the vectorized YSWEEP is the same as that in the unvectorized version. In fact, the vectorization in FLO-22-V1 follows the logic of the unvectorized program very closely. FLO-22-V1, however, is not the optimal vectorization of Successive Line Over-Relaxation for transonic flow calculations. As an example, further efficiency can be implemented in YSWEEP by computing with vector operations all those variables which are not updated on a grid plane. This would mean longer vectors are generated and, therefore, high computational efficiency. There is, nevertheless, a trade off between vector length and storage requirements.

Several changes have been made in other routines in FLO-22 to create FLO-22-V1. These changes, for the most part, have minor effect on the

computational speed. These changes have been either for primary memory addressing of the potential, increased dimensions, or convenience of vector operations. Except for incore addressing, subroutine REFIN has required the most attention. The interpolation in this routine, combined with buffering of planes of the potential in and out of primary memory, has required considerable modification; however, vector instructions have simplified the problem (see Appendix C).

In FLO-22 the maximum number of grid points for the finest grid solution is 192. In FLO-22-V1 this has been expanded to 300. Further, the maximum number of points in the transformed wing normal direction or vertical direction has been expanded from 24 to 32. This implies that considerably more refined solutions can be obtained with FLO-22-V1.

The efficiency of FLO-22-V1 has been observed by running 3-D wing cases on the STAR-100 computer using FLO-22-V1 and comparing the solutions and solution times with the same cases run on the CYBER 175 computer using FLO-22. The wing configurations that have been used in the case runs are the NACA 65_1A012 (stream wise) wing at mach .91 and the ARW-2 wing at mach .8. When the same input was submitted to both FLO-22 and FLO-22-V1, the output results were the same within the relative accuracy of the word lengths of the CYBER and STAR computers. Runs have also been made for the two wings with the expanded grid in FLO-22-V1.

Sizing and timing information for the NACA 65_1A012 wing at mach .91 are presented in table 1. The table shows the program and computer, the number of grid points in each coordinate direction, the total number of grid points, the vector length, the central processor time per grid point, and the ratio of the central processor time per grid point of FLO-22/FLO-22-V1. For the vectorized

program it is observed that the time per grid point in approximately 40% less when the vector length is 300 as compared to when it is 96. For the scalar program the time per grid point is the same regardless of the number of grid points. Another observation from the table is that FLO-22-V1 runs on the STAR-100 1.9 to 3.1 times faster than FLO-22 on the CYBER 175 for the same problem solution. With the dimensions of the potential equal to 300x32x32 FLO-22-V1 requires 464,246 decimal storage locations.

CONCLUDING REMARKS

The computer program (FLO-22) for determining 3-D transonic flow past swept-wing has been modified to use vector operations and run on the STAR-100 computer. The first vectorized version is called FLO-22-V1. Most of the vector operations have been incorporated into the subroutine YSWEEP for the application of Successive Line Over-Relaxation in the transformed horizontal direction. Vector relational operations and control vectors are used to implement upwind differencing at supersonic points. Tridiagonal systems of linear equations are solved with an efficient scalar assembly coded library subroutine. The potential G(I,J,K) is directly addressable as opposed to the buffering in and out of primary memory planes of the potential.

Increased speed of computation and extended grid domains are characteristics of FLO-22-V1. The increased speed of computation is a function of vector length used in the vector operations. In FLO-22-V1 the vector length is equal to the number of grid points in the transformed horizontal direction. For the cases that have been run on the STAR-100 where the vector lengths have been varied from 96-300, the computational time per grid point has

decreased by a factor 1.8–3.1 as compared to the computational time per grid point run on the CYBER 175.

FLO–22–V1 is not the optimal vectorization of Successive Line Over–Relaxation applied to transonic flow; however, it demonstrates that vector operations can readily be implemented to increase the computation rate of the algorithm.

APPENDIX-A

LISTING OF SUBROUTINE YSWEEP IN FLO-22

```
      SUBROUTINE  YSWEEP
      COMMON      G(193,26,4),SO(193,35),EO(131),ZO(131),
     1            IV(193,35),ITE1(35),ITE2(35),
     2            AO(193),A1(193),A2(193),A3(193),
     3            BO(26),B1(26),B2(26),B3(26),
     4            Z(35),C1(35),C2(35),C3(35),
     5            XC(35),XZ(35),XZZ(35),YC(35),YZ(35),YZZ(35),
     6            NX,NY,NZ,KTE1,KTE2,ISYM,KSYM,SCAL,SCALZ,
     7            YAW,CYAW,SYAW,ALPHA,CA,SA,FMACH,N1,N2,N3,ID
      COMMON/FLO/ STRIP,P1,P2,P3,BETA,FR,IR,JR,KR,DG,IG,JG,KG,NS
      COMMON/SWP/ GK1(193,26),GK2(193,26),
     1            SX(193),SZ(193),SXX(193),SXZ(193),SZZ(193),
     2            RO(193),R1(193),C(193),D(193),
     3            G10(26),G20(26),G30(26),G40(26),G1(26),G2(26),
     4            I1,I2,K,L,NO,LX,MX,KY,MY,T1,AAO,Q1,Q2,TYAW,S1
      J1          = 2
      IF (FMACH.GE.1.) J1 = 3
      C(I1-1)     = 0.
      D(I1-1)     = 0.
      DO 12 I=I1,I2
      RO(I)       = 1.
      R1(I)       = 1.
      GK1(I,1)    = G(I,1,L)
   12 GK1(I,J1-1) = G(I,J1-1,L)
      J           = J1
      I3          = I2
   31 BC          = -T1*B1(J)*C1(K)
      DO 32 I=I1,I3
      AB          = -T1*A1(I)*B1(J)
      AC          = T1*A1(I)*C1(K)
      YP          = SO(I,K)  +BO(J)
      A           = 1.  -RO(I)  +AO(I)*AO(I)  +YP*YP
      H           = RO(I)/A
      FH          = RO(I)*A
      P           = AO(I)*(4.*YP*YP  -FH)
      Q           = YP*(4.*AO(I)*AO(I)  -FH)
      A           = XZ(K)*XZ(K)  -YZ(K)*YZ(K)
      B           = (XZ(K)  +XZ(K))*YZ(K)
      AZ          = -AO(I)*XZ(K)  -YP*YZ(K)
      BZ          = -AO(I)*YZ(K)  +YP*XZ(K)
      CZ          = H*H*(P*A  -Q*B)  -AO(I)*XZZ(K)  -YP*YZZ(K)
      DZ          = H*H*(Q*A  +P*B)  -AO(I)*YZZ(K)  +YP*XZZ(K)
      DGI         = G(I+1,J,L)  -G(I-1,J,L)
      DGJ         = G(I,J+1,L)  -GK1(I,J-1)
      DGK         = G(I,J,L+1)  -GK1(I,J)
```

```
 DGII       = G(I+1,J,L)  -G(I,J,L)  -G(I,J,L)  +G(I-1,J,L)
1             +A3(I)*DGI
 DGJJ       = G(I,J+1,L)  -G(I,J,L)  -G(I,J,L)  +G(I,J-1,L)
1             -B3(J)*DGJ


 DGKK       = G(I,J,L+1)  -G(I,J,L)  -G(I,J,L)  +G(I,J,L-1)
1             +C3(K)*DGK
 DGIJ       = G(I+1,J+1,L)  -G(I-1,J+1,L)
1             -G(I+1,J-1,L)  +G(I-1,J-1,L)
 DGIK       = G(I+1,J,L+1)  -G(I+1,J,L-1)
1             -G(I-1,J,L+1)  +G(I-1,J,L-1)
 DGJK       = G(I,J+1,L+1)  -G(I,J-1,L+1)
1             -G(I,J+1,L-1)  +G(I,J-1,L-1)
 GX         = A1(I)*DGI
 GY         = -B1(J)*DGJ
 U          = GX  -SX(I)*GY  +CA*AO(I)  +SA*YP
 V          = GY  +SA*AO(I)  -CA*YP
 W          = RO(I)*(C1(K)*DGK  -SZ(I)*GY  +SYAW
1                  +CA*XZ(K)  +SA*YZ(K)  +H*(U*AZ  +V*BZ))
 AU         = U  +W*AZ
 AV         = V  +W*BZ
 QXY        = H*(U*U  +V*V)
 QQ         = QXY  +W*W
 AA         = DIM(AAO,.2*QQ)
 HZ         = AZ*SX(I)  -BZ  +FH*SZ(I)
 FXX        = 1.  +H*AZ*AZ
 FYY        = 1.  +SX(I)*SX(I)  +H*HZ*HZ
 FXY        = SX(I)  +H*AZ*HZ
 BV         = AV  -AU*SX(I)  -FH*W*SZ(I)
 UU         = H*AU*AU
 VV         = H*BV*BV
 WW         = FH*W*W
 UV         = H*AU*BV
 UW         = AU*W
 VW         = BV*W
 AXX        = R1(I)*(FXX*AA  -UU)
 AZZ        = FH*AA  -WW
 AXZ        = (RO(I)  +RO(I)) * (AZ*AA  -UW)
 R          = -(AXX*SXX(I)  +AZZ*SZZ(I)  +AXZ*SXZ(I))*GY
1             +T1*(AA*(CZ*GX +(DZ-SX(I)*CZ) *GY)* RO(I)
2                  -H*(CA*(AU*AU  -AV*AV)  +(SA  +SA)*AU*AV
3                      -QXY*(U*AO(I)  +V*YP
4                      +(W  +W)*(AO(I)*AZ  +YP*BZ)))
5             -WW*(CA*XZZ(K)  +SA*YZZ(K))  -W*W*(U*CZ  +V*DZ))
```

```
      AXT         = ABS(AU*A1(I))
      AYT         = ABS(BV*B1(J))
      AZT         = ABS(FH*W*C1(K))
      A           = RO(I)*BETA*AA/AMAX1(AXT,AYT,AZT,(1.  -RO(I)))
      AXT         = A*AXT
      AYT         = A*AYT
      AZT         = A*AZT
      IF (QQ.GE.AA) GO TO 33
      AXX         = AXX*A2(I)
      AYY         = (FYY*AA  -VV)*B2(J)
      AZZ         = AZZ*C2(K)
      AXY         = -R1(I)*(FXY*AA  +UV)*(AB  +AB)
      AXZ         = AXZ*AC
      AYZ         = -RO(I)* (HZ*AA  +VW) *(BC  +BC)
      BP          = AXX
      BM          = AXX
      B           = -AXX  -AXX  -Q1*(AYY  +AZZ)
      R           = AXX*DGII  +AYY*DGJJ  +AZZ*DGKK
     1              +AXY*DGIJ  +AYZ*DGJK  +AXZ*DGIK  +R
      GO TO 35
   33 NS          = NS  +1
      S           = SIGN(1.,U)
      IM          = I  -IFIX(S)
      IMM         = IM  -IFIX(S)
      AXX         = UU*A2(I)
      AYY         = VV*B2(J)
      AZZ         = WW*C2(K)
      AXY         = 8.*S*UV*AB
      AXZ         = 8.*S*UW*AC
      AYZ         = 8.*VW*BC
      BXX         = (FXX*QQ  -UU)*A2(I)
      BYY         = (FYY*QQ  -VV)*B2(J)
      BZZ         = (FH*QQ  -WW)*C2(K)
      BXY         = -(FXY*QQ  +UV)*(AB  +AB)
      BXZ         = (AZ*QQ  -UW)*(AC  +AC)
      BYZ         = -(HZ*QQ  +VW)*(BC  +BC)
      AQ          = AA/QQ
      DELTAG      = BXX*DGII  +BYY*DGJJ  +BZZ*DGKK
     1              +BXY*DGIJ  +BYZ*DGJK  +BXZ*DGIK
      DGII        = G(I,J,L)  -G(IM,J,L)  -G(IM,J,L)  +G(IMM,J,L)
     1              +A3(I)*DGI
      DGJJ        = G(I,J,L)  -G(I,J-1,L)  -G(I,J-1,L)  +GK1(I,J-2)
     1              -B3(J)*DGJ
      DGKK        = G(I,J,L)  -G(I,J,L-1)  -G(I,J,L-1)  +GK2(I,J)
     1              +C3(K)*DGK
```

```
   DGIJ       = G(I,J,L)  -G(IM,J,L)
  1             -G(I,J-1,L)  +G(IM,J-1,L)
   DGIK       = G(I,J,L)  -G(I,J,L-1)
  1             -G(IM,J,L)  +G(IM,J,L-1)
   DGJK       = G(I,J,L)  -G(I,J,L-1)
  1             -G(I,J-1,L)  +G(I,J-1,L-1)
   GSS        = AXX*DGII  +AYY*DGJJ  +AZZ*DGKK
  1             +AXY*DGIJ  +AYZ*DGJK  +AXZ*DGIK
   B          = .5*(AQ  -1.)*(AXX  +AXX  +AXY  +AXZ)
   BP         = AQ*BXX  -(1.  -S)*B
   BM         = AQ*BXX  -(1.  +S)*B
   B          = -AQ*(BXX  +BXX  +Q2*(BYY  +BZZ))
  1             +(AQ  -1.)*(2.*(AXX  +AYY  +AZZ)  +AXY  +AYZ  +AXZ)
   R          = (AQ  -1.)*GSS  +AQ*DELTAG  +R


35 IF (ABS(R).LE.ABS(FR)) GO TO 37
   FR         = R
   IR         = I
   JR         = J
   KR         = K
37 R          = R  -AYT*(GK1(I,J-1)  -G(I,J-1,L))
  1             -AZT*(GK1(I,J)  -G(I,J,L-1))
   B          = B  -AXT  -AYT  -AZT
   BM         = BM  +AXT
   B          = 1./(B  -BM*C(I-1))
   C(I)       = B*BP
32 D(I)       = B*(R  -BM*D(I-1))
   CG         = 0.
   I          = I3
   DO 42 M=I1,I3
   CG         = D(I)  -C(I)*CG
   IF (ABS(CG).LE.ABS(DG)) GO TO 43
   DG         = CG
   IG         = I
   JG         = .
   KG         = K
43 GK2(I,J)   = GK1(I,J)
   GK1(I,J)   = G(I,J,L)
   G(I,J,L)   = G(I,J,L)  -CG
42 I          = I  -1
   J          = J  +1
   IF (J  -KY) 31,51,61
51 IF (I2.GT.ITE2(K)) I3 = ITE2(K)
   IF (ITE2(K).EQ.MX) I3 = LX
```

```
      DO 52 I=I1,I3
      LV          = IABS(1  -IABS(IV(I,K)))
      RO(I)       = AMIN0(LV,IABS(IV(I,K)))
   52 R1(I)       = LV
      GO TO 31
   61 N           = NO
      I           = LX  +1
      IF (K.LT.KTE1.OR.K.GT.KTE2) GO TO 71
      IO          = NX  +2  -I3
      DO 62 I=IO,I3
      A           = 1.  -RO(I)  +AO(I)*AO(I)  +SO(I,K)*SO(I,K)
      H           = RO(I)/A
      FH          = RO(I)*A
      AZ          = -AO(I)*XZ(K)  -SO(I,K)*YZ(K)
      BZ          = -AO(I)*YZ(K)  +SO(I,K)*XZ(K)
      HZ          = AZ*SX(I)  -BZ  +FH*SZ(I)
      FYY         = 1.  +SX(I)*SX(I)  +H*HZ*HZ
      FXY         = SX(I)  +H*AZ*HZ
      DGI         = G(I+1,KY,L)  -G(I-1,KY,L)
      DGK         = G(I,KY,L+1)  -GK2(I,KY)
      V           = SA*AO(I)  -CA*SO(I,K)
      U           = A1(I)*DGI  +CA*AO(I)  +SA*SO(I,K)
      W           = C1(K)*DGK  +SYAW  +CA*XZ(K)  +SA*YZ(K)
   62 G(I,KY+1,L) = G(I,KY-1,L)
     1              +(V*(1.  -H*BZ*HZ)  -U*FXY  -W*HZ)/(FYY*B1(KY))
      I           = IO
      IF (IO.NE.ITE1(K)) GO TO 71
      E           = G(I3,KY,L)  -G(IO,KY,L)
      NO          = NO  +1
      EO(NO)      = EO(NO)  +P3*(E  -EO(NO))
      N           = NO
   71 IF (I.LE.I1) RETURN
      I           = I  -1
      E           = 0.
      IF (IV(I,K).NE.1) GO TO 77
      ZZ          = Z(K)  -TYAW*(XC(K)  +S1*AO(I)*AO(I))
   73 IF (ZZ.GE.ZO(N-1)) GO TO 75
      N           = N  -1
      GO TO 73
   75 R           = (ZZ  -ZO(N-1))/(ZO(N)  -ZO(N-1))
      E           = R*EO(N)  +(1.  -R)*EO(N-1)
   77 M           = NX  +2  -I
      G(I,KY+1,L) = G(M,KY-1,L)  -E
      G(M,KY+1,L) = G(I,KY-1,L)  +E
      GK2(M,KY)   = GK1(M,KY)
      GK1(M,KY)   = G(M,KY,L)
      G(M,KY,L)   = G(I,KY,L)  +E
      GO TO 71
      END
```

APPENDIX-B

LISTING OF SUBROUTINE YSWEEP IN FLO-22-V1

```
 SUBROUTINE  YSWEEP
 COMMON      G(301,34,35),S0(301,35),E0(131),Z0(131),
1            IV(301,35),ITE1(35),ITE2(35),
2            A0(301),A1(301),A2(301),A3(301),
3            B0(34),B1(34),B2(34),B3(34),
4            Z(35),C1(35),C2(35),C3(35),
5            XC(35),XZ(35),XZZ(35),YC(35),YZ(35),YZZ(35),
6            NX,NY,NZ,KTE1,KTE2,ISYM,KSYM,SCAL,SCALZ,
7            YAW,CYAW,SYAW,ALPHA,CA,SA,FMACH,N1,N2,N3,IO
 COMMON/FLO/ STRIP,P1,P2,P3,BETA,FR,IR,JR,KR,DG,IG,JG,KG,NS
 COMMON/SWP/ GK1(301,34),GK2(301,34),
1            SX(301),SZ(301),SXX(301),SXZ(301),SZZ(301),
2            R0(301),R1(301),C(301),D(301),
3            G10(34),G20(34),G30(34),G40(34),G1(34),G2(34),
4            I1,I2,K,L,NO,LX,MX,KY,MY,T1,AA0,Q1,Q2,TYAW,S1
 COMMON /VSTAR/ YP(301),AJ(301),H(301),FH(301),AZ(301),BZ(301),
1CZ(301),DZ(301),DGI(301),DGJ(301),DGK(301),U(301),V(301),W(301),
2AU(301),AV(301),GG(301),AA(301),HZ(301),FXX(301),FYY(301),FXY(301)
3,BV(301),AG(301),UU(301),VV(301),WW(301),UV(301),UW(301),VW(301),
4AXX(301),AYY(301),AZZ(301),AXZ(301),AXY(301),AYZ(301),AXT(301),
5AYT(301),AZT(301),R(301),DGII(301),DGJJ(301),DGKK(301),DGIJ(301),
6DGIK(301),DGJK(301),SUBD(301),SUPD(301),DIAG(301),S(301),
7DELTAG(301),HOLD1(301),HOLD2(301),HOLD3(301),HOLD4(301),HOLD5(301)
8,TEMP(301),TEMP1(301),TEMP2(301)
9  ,SAVE(301),LU(301),ITEMP(301),CG(301),HOLD(301)
 BIT  BI1(301),BI2(301)
 DESCRIPTOR DR
 J1 =2
 IF (FMACH.GE.1.) J1=3
 C(I1-1)=0.0
 D(I1-1)=0.0
 LV= I2 -I1+1
 R0(I1;LV) =1.0
 R1(I1;LV)= 1.0
  GK1(I1,1;LV) = G(I1,1,L;LV)
 GK1(I1,J1-1;LV)= G(I1,J1-1,L;LV)
 J= J1
 I3= I2
```

```
31    LV= I3-I1 +1
      YP(1;LV)=SO(I1,K;LV) + BO(J)
      SAVE(1;LV)=1.0-RO(I1;LV)
      TEMP2(1;LV)=YP(1;LV)* YP(1;LV)
      TEMP(1;LV)= AO(I1;LV)* AO(I1;LV)
      AJ(1;LV) = SAVE(1;LV)+ TEMP(1;LV)+ TEMP2(1;LV)
      H(1;LV) =RO(I1;LV)/AJ(1;LV)
      FH(1;LV)=RO(I1;LV)*AJ(1;LV)
      TEMP1(1;LV)=AO(I1;LV)* (4.0*TEMP2(1;LV)-FH(1;LV))
      TEMP2(1;LV)=YP(1;LV)*(4.0*TEMP(1;LV)-FH(1;LV))
        A= XZ(K)*XZ(K)-YZ(K)*YZ(K)
        B= (XZ(K)+XZ(K))* YZ(K)
      AZ(1;LV)= -AO(I1;LV)*XZ(K) -YP(1;LV)*YZ(K)
      BZ(1;LV)= -AO(I1;LV)*YZ(K) +YP(1;LV)*XZ(K)
      TEMP(1;LV)= H(1;LV)* H(1;LV)
      CZ(1;LV)= TEMP(1;LV)*(TEMP1(1;LV)*A -TEMP2(1;LV)*B)
     1          -AO(I1;LV)*XZZ(K) -YP(1;LV)*YZZ(K)
      DZ(1;LV)= TEMP(1;LV)*(TEMP2(1;LV)*A +TEMP1(1;LV)*B)
     1          -AO(I1;LV)*YZZ(K) +YP(1;LV)*XZZ(K)
      DGI(1;LV)=G(I1+1,J,L;LV) - G(I1-1,J,L;LV)
      DGJ(1;LV)=G(I1,J+1,L;LV) - GK1(I1,J-1;LV)
      DGK(1;LV)=G(I1,J,L+1;LV) - GK1(I1,J;LV)
      TEMP1(1;LV)= A1(I1;LV) * DGI(1;LV)
      TEMP2(1;LV)= -B1(J) * DGJ(1;LV)
      U(1;LV)= TEMP1(1;LV)-SX(I1;LV)*TEMP2(1;LV)+CA*AO(I1;LV)
     1        +SA * YP(1;LV)
      V(1;LV)= TEMP2(1;LV)+SA *AO(I1;LV)-CA* YP(1;LV)
      W(1;LV)= RO(I1;LV)*(C1(K)*DGK(1;LV)-SZ(I1;LV)*TEMP2(1;LV)+SYAW
     1        +CA*XZ(K)+SA*YZ(K)+H(1;LV)*(U(1;LV)*AZ(1;LV)+V(1;LV)
     2        *BZ(1;LV)))
      AU(1;LV)= U(1;LV)+W(1;LV)*AZ(1;LV)
      AV(1;LV)= V(1;LV)+W(1;LV)*BZ(1;LV)
      TEMP(1;LV)=H(1;LV)*(U(1;LV)*U(1;LV) +V(1;LV)*V(1;LV))
      QQ(1;LV)= TEMP(1;LV) +W(1;LV)*W(1;LV)
       HOLD1(1;LV) = .2 * QQ(1;LV)
      AA(1;LV)= VDIM(AAO,HOLD1(1;LV);AA(1;LV))
      HZ(1;LV)= AZ(1;LV)*SX(I1;LV) - BZ(1;LV)+FH(1;LV)*SZ(I1;LV)
      FXX(1;LV)= 1.0+ H(1;LV) *AZ(1;LV)*AZ(1;LV)
      FYY(1;LV)= 1.0+ SX(I1;LV)*SX(I1;LV)+H(1;LV)*HZ(1;LV)*HZ(1;LV)
      FXY(1;LV)= SX(I1;LV) +H(1;LV)*AZ(1;LV)*HZ(1;LV)
      BV(1;LV)= AV(1;LV)-AU(1;LV)*SX(I1;LV)-FH(1;LV)*W(1;LV)*SZ(I1;LV)
      UU(1;LV)= H(1;LV)* AU(1;LV)* AU(1;LV)
      VV(1;LV)= H(1;LV)* BV(1;LV)* BV(1;LV)
      WW(1;LV)= FH(1;LV) * H(1;LV)* W(1;LV)
      UV(1;LV)= H(1;LV)* AU(1;LV) * BV(1;LV)
      UW(1;LV)= AU(1;LV)* W(1;LV)
      VW(1;LV)= BV(1;LV)* W(1;LV)
```

```
 AXX(1;LV)=R1(I1;LV)*(FXX(1;LV)*AA(1;LV)-UU(1;LV))
 AZZ(1;LV)=FH(1;LV)*AA(1;LV)-WW(1;LV)
 AXZ(1;LV)=(2.0* RO(I1;LV))*(AZ(1;LV)*AA(1;LV)-UW(1;LV))
 HOLD1(1;LV)=-TEMP2(1;LV)*(AXX(1;LV)*SXX(I1;LV)+AZZ(1;LV)
1             *SZZ(I1;LV)+AXZ(1;LV)*SXZ(I1;LV))
 HOLD2(1;LV)=(AA(1;LV)*(CZ(1;LV)*TEMP1(1;LV)+(DZ(1;LV)-SX(I1;LV)
1             *CZ(1;LV))*TEMP2(1;LV))) * RO(I1;LV)
 HOLD3(1;LV)= CA*(AU(1;LV)*AU(1;LV)-AV(1;LV)*AV(1;LV))+(SA+SA)*
1             AU(1;LV)*AV(1;LV)
 HOLD4(1;LV)= TEMP(1;LV)*(U(1;LV)*AO(I1;LV)+V(1;LV)*YP(1;LV)+2.0*
1          W(1;LV)*(AO(I1;LV)*AZ(1;LV)+YP(1;LV)*BZ(1;LV)))
 HOLD5(1;LV)=-WW(1;LV)*(CA*XZZ(K)+SA*YZZ(K))-W(1;LV)*W(1;LV)*
1            (U(1;LV)*CZ(1;LV)+V(1;LV)*DZ(1;LV))
 R(1;LV)= HOLD1(1;LV) + T1*(HOLD2(1;LV)-H(1;LV)*(HOLD3(1;LV)-
1          HOLD4(1;LV))+ HOLD5(1;LV))
 AXT(1;LV)= AU(1;LV)*A1(I1;LV)
 AXT(1;LV)= VABS(AXT(1;LV);AXT(1;LV))
 AYT(1;LV)= BV(1;LV)* B1(J)
 AYT(1;LV)= VABS(AYT(1;LV);AYT(1;LV))
 AZT(1;LV)= FH(1;LV)* W(1;LV) * C1(K)
 AZT(1;LV)= VABS(AZT(1;LV);AZT(1;LV))
 HOLD1(1;LV)= AYT(1;LV)
 BI1(1;LV)= AXT(1;LV).GE.HOLD1(1;LV)
 HOLD1(1;LV)= Q8VCTRL(AXT(1;LV),BI1(1;LV);HOLD1(1;LV))
 BI1(1;LV)= AZT(1;LV).GE.SAVE(1;LV)
 SAVE(1;LV)= Q8VCTRL(AZT (1;LV),BI1(1;LV);SAVE(1;LV))
 BI1(1;LV)=HOLD1(1;LV).GE.SAVE(1;LV)
 SAVE(1;LV)= Q8VCTRL(HOLD1(1;LV),BI1(1;LV);SAVE(1;LV))
 HOLD1(1;LV)= RO(I1;LV)*BETA* AA(1;LV)/SAVE(1;LV)
 AXT(1;LV)= AXT(1;LV)* HOLD1(1;LV)
 AYT(1;LV)= AYT(1;LV)* HOLD1(1;LV)
 AZT(1;LV)= AZT(1;LV)* HOLD1(1;LV)
 SAVE(1;LV)= G(I1,J,L;LV) + G(I1,J,L;LV)
 DGII(1;LV)=G(I1+1,J,L;LV) -SAVE(1;LV) +G(I1-1,J,L;LV)+A3(I1;LV)
1             *DGI(1;LV)
 DGJJ(1;LV)=G(I1,J+1,L;LV)-SAVE(1;LV)+G(I1,J-1,L;LV)-B3(J)
1             *DGJ(1;LV)
 DGKK(1;LV)=G(I1,J,L+1;LV)-SAVE(1;LV)+G(I1,J,L-1;LV)+C3(K)
1             *DGK(1;LV)
 DGIJ(1;LV)=G(I1+1,J+1,L;LV)-G(I1-1,J+1,L;LV)-G(I1+1,J-1,L;LV)
1             +G(I1-1,J-1,L;LV)
 DGIK(1;LV)=G(I1+1,J,L+1;LV)-G(I1+1,J,L-1;LV)-G(I1-1,J,L+1;LV)
1             +G(I1-1,J,L-1;LV)
 DGJK(1;LV)=G(I1,J+1,L+1;LV)-G(I1,J-1,L+1;LV)-G(I1,J+1,L-1;LV)
1             +G(I1,J-1,L-1;LV)
```

```
 AXX(1;LV)= AXX(1;LV) * A2(I1;LV)
 AYY(1;LV)= (FYY(1;LV)*AA(1;LV)-VV(1;LV))*B2(J)
 AZZ(1;LV)= AZZ(1;LV)* C2(K)
  TEMP1(1;LV)=  A1(I1;LV)*B1(J)*(-T1)
  TEMP2(1;LV)=  A1(I1;LV)*C1(K)*T1
 AXY(1;LV)=-R1(I1;LV)*(FXY(1;LV)*AA(1;LV)+UV(1;LV))*2.0*TEMP1(1;LV)
 AXZ(1;LV)= AXZ(1;LV)* TEMP2(1;LV)
  CONST = -T1 * B1(J) * C1(K)
 AYZ(1;LV)= -R0(I1;LV)*(HZ(1;LV)*AA(1;LV)+VW(1;LV))*2.0*CONST
 SUPD(1;LV)= AXX(1;LV)
 SUBD(1;LV)= AXX(1;LV)
 DIAG(1;LV)=-2.0*AXX(1;LV) -Q1*(AYY(1;LV)+AZZ(1;LV))
   BI2(1;LV)= QQ(1;LV).LT.AA(1;LV)
   NNS = Q8SCNT(BI2(1;LV))
 TEMP(1;LV)= AXX(1;LV)*DGII(1;LV) +AYY(1;LV)*DGJJ(1;LV)+AZZ(1;LV)*
1         DGKK(1;LV)+AXY(1;LV)*DGIJ(1;LV)+AYZ(1;LV)*DGJK(1;LV) +
2         AXZ(1;LV)*DGIK(1;LV) + R(1;LV)
   R(1;LV) = Q8VCTRL(TEMP(1;LV),BI2(1;LV);R(1;LV))
 BI1(1;LV)= QQ(1;LV).GE.AA(1;LV)
 NS = Q8SCNT(BI1(1;LV)) +NS
  S(1;LV)=VSIGN(1.0,U(1;LV);S(1;LV))
 AXX(1;LV) = UU(1;LV)* A2(I1;LV)
 AYY(1;LV) = VV(1;LV)*B2(J)
 AZZ(1;LV) = WW(1;LV)* C2(K)
 AXY(1;LV) = 8.*S(1;LV)*UV(1;LV)*TEMP1(1;LV)
 AXZ(1;LV) = 8.*S(1;LV)*UW(1;LV)*TEMP2(1;LV)
 AYZ(1;LV) = 8.*S(1;LV)*VW(1;LV)*CONST
 HOLD1(1;LV)=(FXX(1;LV)*QQ(1;LV)-UU(1;LV)) *A2(I1;LV)
 HOLD2(1;LV)=(FYY(1;LV)*QQ(1;LV)-VV(1;LV)) *B2(J)
 HOLD3(1;LV)=(FH(1;LV)*QQ(1;LV)-WW(1;LV))*C2(K)
 HOLD4(1;LV)=-(FXY(1;LV)*QQ(1;LV)+UV(1;LV))* 2.0*TEMP1(1;LV)
 HOLD5(1;LV)=(AZ(1;LV)*QQ(1;LV)-UW(1;LV))*2.0*TEMP2(1;LV)
 TEMP(1;LV)= -(HZ(1;LV)*QQ(1;LV)+VW(1;LV))* 2.0*CONST
  TEMP1(1;LV)= 1.0
  TEMP1(1;LV)= Q8VCTRL(QQ(1;LV),BI1(1;LV);TEMP1(1;LV))
  AQ(1;LV) = AA(1;LV)/TEMP1(1;LV)
 DELTAG(1;LV)=HOLD1(1;LV)*DGII(1;LV)+HOLD2(1;LV)*DGJJ(1;LV)
1            +HOLD3(1;LV)*DGKK(1;LV)+ HOLD4(1;LV)*DGIJ(1;LV)
2            +HOLD5(1;LV)*DGIK(1;LV)+ TEMP(1;LV)*DGJK(1;LV)
 BI2(1;LV)=S(1;LV).NE.(1.0)
 TEMP1(1;LV) = G(I1-1,J,L;LV)
 TEMP1(1;LV) = Q8VCTRL(G(I1+1,J,L;LV),BI2(1;LV);TEMP1(1;LV))
 TEMP2(1;LV) = G(I1-2,J,L;LV)
 TEMP2(1;LV) = Q8VCTRL(G(I1+2,J,L;LV),BI2(1;LV);TEMP2(1;LV))
 SAVE(1;LV)= G(I1,J,L;LV) - TEMP1(1;LV)
 TEMP(1;LV)=   SAVE(1;LV) -    TEMP1(1;LV)+TEMP2(1;LV)+A3(I1;LV)
1            *DGI(1;LV)
```

```
      DGII(1;LV)= Q8VCTRL(TEMP(1;LV),BI1(1;LV);DGII(1;LV))
      TEMP1(1;LV)= G(I1-1,J-1,L;LV)
      TEMP1(1;LV)= Q8VCTRL(G(I1+1,J-1,L;LV),BI2(1;LV);TEMP1(1;LV))
      TEMP(1;LV) =SAVE(1;LV)+TEMP1(1;LV)- G(I1,J-1,L;LV)
      DGIJ(1;LV) =Q8VCTRL(TEMP(1;LV),BI1(1;LV);DGIJ(1;LV))
      TEMP1(1;LV)= G(I1-1,J,L-1;LV)
      TEMP1(1;LV)= Q8VCTRL(G(I1+1,J,L-1;LV),BI2(1;LV);TEMP1(1;LV))
      TEMP(1;LV)= SAVE(1;LV)-G(I1,J,L-1;LV)+ TEMP1(1;LV)
      DGIK(1;LV)= Q8VCTRL(TEMP(1;LV),BI1(1;LV);DGIK(1;LV))
      DGJJ(1;LV)= G(I1,J,L;LV)-2.0*G(I1,J-1,L;LV)+GK1(I1,J-2;LV)
     1   -B3(J)  *DGJ(1;LV)
      DGKK(1;LV)= G(I1,J,L;LV)-2.0*G(I1,J,L-1;LV)+GK2(I1,J;LV)
     1           +C3(K) *DGK(1;LV)
      DGJK(1;LV)= G(I1,J,L;LV)-G(I1,J,L-1;LV) -G(I1,J-1,L;LV)
     1           + G(I1,J-1,L-1;LV)
      TEMP(1;LV)= AXX(1;LV)*DGII(1;LV) +AYY(1;LV)*DGJJ(1;LV)
     1          +AZZ(1;LV)*DGKK(1;LV)+AXY(1;LV)*DGIJ(1;LV)
     2          +AYZ(1;LV)*DGJK(1;LV)+AXZ(1;LV)*DGIK(1;LV)
      TEMP1(1;LV)= AQ(1;LV)-1.0
      TEMP2(1;LV)= AXY(1;LV) +AXZ(1;LV)
      SAVE(1;LV)= .5*TEMP1(1;LV)*(2.0*AXX(1;LV)+TEMP2(1;LV))
      HOLD4(1;LV)= AQ(1;LV)*HOLD1(1;LV)
      HOLD(1;LV) = HOLD4(1;LV) -(1.0-S(1;LV))*SAVE(1;LV)
       SUPD(1;LV)=Q8VCTRL(HOLD(1;LV),BI1(1;LV);SUPD(1;LV))
      HOLD(1;LV) = HOLD4(1;LV) -(1.0+S(1;LV))*SAVE(1;LV)
       SUBD(1;LV)=Q8VCTRL(HOLD(1;LV),BI1(1;LV);SUBD(1;LV))
      HOLD(1;LV) =-AQ(1;LV)*(2.0*HOLD1(1;LV)+Q2*(HOLD2(1;LV)+HOLD3
     1 (1;LV)))  +TEMP1(1;LV)*(2.0*(AXX(1;LV)+AYY(1;LV)+AZZ(1;LV))
     2           +TEMP2(1;LV) +AYZ(1;LV))
       DIAG(1;LV)=Q8VCTRL(HOLD(1;LV),BI1(1;LV);DIAG(1;LV))
      HOLD(1;LV)= TEMP1(1;LV)*TEMP(1;LV)+AQ(1;LV)*DELTAG(1;LV)+
     1 R(1;LV)
       R(1;LV)=Q8VCTRL(HOLD(1;LV),BI1(1;LV);R(1;LV))
      ASSIGN  DR,R(1;LV)
      CALL Q8MAX(X'04',0,DR,0,ICOUNT,0,CONST)
      IF (ABS(CONST).LE.ABS(FR)) GO TO 37
      FR = CONST
      IR = ICOUNT+I1
      JR = J
      KR = K
37     CONTINUE
      R(1;LV)=R(1;LV)-AYT(1;LV)*(GK1(I1,J-1;LV)-G(I1,J-1,L;LV))
     1       -AZT(1;LV) * (GK1(I1,J;LV)-G(I1,J,L-1;LV))
      DIAG(1;LV)=DIAG(1;LV)-AXT(1;LV)-AYT(1;LV)-AZT(1;LV)
      SUBD(1;LV)=SUBD(1;LV)+AXT(1;LV)
       IOPT=0
```

```
      CALL Q4TRIDGE(LV,SUBD,DIAG,SUPD,C,D,IOPT,R)
      CG(I1;LV)=R(1;LV)
     ASSIGN DR,CG(I1;LV)
     CALL Q8MAX(X'04',0,DR,0,ICOUNT,0,CONST)
     IF (ABS(CONST).LE.ABS(DG)) GO TO 43
     DG =CONST
     IG = ICOUNT+I1
     JG = J
     KG = K
43   GK2(I1,J;LV) = GK1(I1,J;LV)
     GK1(I1,J;LV) =G(I1,J,L;LV)
     G(I1,J,L;LV) = G(I1,J,L;LV) -CG(I1;LV)
     J=J+1
     IF (J-KY) 31,51,61
51   IF (I2.GT.ITE2(K)) I3=ITE2(K)
     IF (ITE2(K).EQ.MX) I3=LX
       LV=  I3-I1 +1
     ITEMP(1;LV)=  VIABS(IV(I1,K;LV);ITEMP(1;LV))
     LU(1;LV) = 1 -ITEMP(1;LV)
     LU(1;LV)= VIABS(LU(1;LV);LU(1;LV))
     BI1(1;LV) = LU(1;LV).LT.ITEMP(1;LV)
     ITEMP(1;LV)= Q8VCTRL(LU(1;LV),BI1(1;LV);ITEMP(1;LV)  )
     RO(I1;LV) = ITEMP(1;LV)
     R1(I1;LV) = LU(1;LV)
     GO TO 31
61   N  = NO
     I  = LX +1
     IF (K.LT.KTE1.OR.K.GT.KTE2) GO TO 71
     IO =NX + 2- I3
     LV = I3-IO +1
       SAVE(1;LV)=1.0 -RO(IO;LV)+AO(IO;LV)*AO(IO;LV)+SO(IO,K;LV)
    1 * SO(IO,K;LV)
       H(1;LV) = RO(IO;LV)/SAVE(1;LV)
       FH(1;LV) = RO(IO;LV)*SAVE(1;LV)
     AZ(1;LV)=-AO(IO;LV)* XZ(K)-SO(IO,K;LV)*YZ(K)
     BZ(1;LV)=-AO(IO;LV)* YZ(K)+SO(IO,K;LV)*XZ(K)
     HZ(1;LV)= AZ(1;LV) * SX(IO;LV)-BZ(1;LV)+FH(1;LV)*SZ(IO;LV)
     TEMP(1;LV)= H(1;LV)* HZ(1;LV)
     FYY(1;LV) = 1.0 +SX(IO;LV)*SX(IO;LV)+HZ(1;LV)*TEMP(1;LV)
     FXY(1;LV) = SX(IO;LV)+ AZ(1;LV)* TEMP(1;LV)
     DGI(1;LV) = G(IO+1,KY,L;LV)-G(IO-1,KY,L;LV)
     DGK(1;LV) = G(IO,KY,L+1;LV)-GK2(IO,KY;LV)
     V(1;LV) = SA*AO(IO;LV) -CA* SO(IO,K;LV)
     U(1;LV) =A1(IO;LV)*DGI(1;LV)+CA*AO(IO;LV)+SA*SO(IO,K;LV)
     W(1;LV) =C1(K)*DGK(1;LV) +SYAW +CA*XZ(K) +SA*YZ(K)
     G(IO,KY+1,L;LV)= G(IO,KY-1,L;LV)+(V(1;LV)*(1.0-BZ(1;LV)*TEMP(1;LV)
    1          )-U(1;LV)*FXY(1;LV)-W(1;LV)*HZ(1;LV))/(FYY(1;LV)*B1(KY))
```

```
      I = IO
      IF (IO.NE.ITE1(K)) GO TO 71
      E  = G(I3,KY,L)-G(IO,KY,L)
      NO = NO+1
      EO(NO) = EO(NO) +P3*(E -EC(NO))
      N  = NO
   71 IF(I.LE.I1) RETURN
      I = I-1
      E = 0.
      IF( IV(I,K).NE.1) GO TO 77
      ZZZ = Z(K) - TYAW*(XC(K) +S1*AO(I)*AO(I))
   73 IF (ZZZ.GE.ZO(N-1)) GO TO 75
      N = N-1
      GO TO 73
   75 CONST =(ZZZ -ZO(N-1))/(ZO(N)-ZO(N-1))
      E = CONST * EO(N) +(1.0-CONST)* EO(N-1)
   77 M = NX +2 -I
      G(I,KY+1,L)= G(M,KY-1,L) -E
      G(M,KY+1,L)= G(I,KY-1,L) +E
      GK2(M,KY) = GK1(M,KY)
      GK1(M,KY) = G(M,KY,L)
      G(M,KY,L) = G(I,KY,L) +E
      GO TO 71
       END
```

APPENDIX-C

LISTING OF SUBROUTINE REFIN IN FLO-22-V1

```
      SUBROUTINE  REFIN
      COMMON      G(301,34,35),SO(301,35),EO(131),ZO(131),
     1            IV(301,35),ITE1(35),ITE2(35),
     2            AO(301),A1(301),A2(301),A3(301),
     3            BO(34),B1(34),B2(34),B3(34),
     4            Z(35),C1(35),C2(35),C3(35),
     5            XC(35),XZ(35),XZZ(35),YC(35),YZ(35),YZZ(35),
     6            NX,NY,NZ,KTE1,KTE2,ISYM,KSYM,SCAL,SCALZ,
     7            YAW,CYAW,SYAW,ALPHA,CA,SA,FMACH,N1,N2,N3,ID
      MX          = NX  +1
      KY          = NY  +1
      MY          = NY  +2
      MZ          = NZ  +3
      MXO         = NX/2  +1
      MYO         = NY/2  +2
      MZO         = NZ/2  +2
         IVL= 301 * MY
      K           = 1
       KK=K
      IF (KSYM.EQ.0) GO TO 11
      MZO         = NZ/2  +3
      K           = 2
       KK=K
11    DO 53  K=KK,MZO
      J           = NY/2  +1
      JJ          = KY
21    I           = MXO
      II          = MX
31    G(II,JJ,K)  = G(I,J,K)
      I           = I  -1
      II          = II  -2
      IF (I.GT.0) GO TO 31
      J           = J  -1
      JJ          = JJ  -2
      IF (J.GT.0) GO TO 21
      DO 42 J=1,KY,2
      DO 42 I=2,NX,2
42    G(I,J,K)    = .5*(G(I+1,J,K)  +G(I-1,J,K))
      DO 52 I=1,MX
      DO 54 J=2,NY,2
54    G(I,J,K)    = .5*(G(I,J+1,K)  +G(I,J-1,K))
52    G(I,MY,K)   = 0.
53    CONTINUE
```

```
      IF (KSYM.NE.0)  GO TO 111
      MZM  = MZO
      MZST = NZ+1
      GO TO 112
111   MZM  = MZO
      MZST = MZ
112   G(1,1,MZST;IVL) = G(1,1,MZM;IVL)
      IF (MZST.EQ.1)  GO TO 113
      MZST= MZST - 1
      G(1,1,MZST;IVL) = .5 * (G(1,1,MZM;IVL) + G(1,1,MZM-1;IVL))
      MZM  = MZM -1
      MZST = MZST - 1
      GO TO 112
 113   CONTINUE
     TYAW          = SYAW/CYAW
     S1            = .5*SCAL
     NO            = KTE1  -1
     EC(NO)        = 0.
     K             = 2
       KK=K
     IF (KSYM.NE.0) KK =  K+1
      DO  251  K=KK,MZ
     N             = NC
     I             = MXC  +1
     IF (K.LT.KTE1.OR.K.GT.KTE2) GO TO 231
     I1            = ITE1(K)
     I2            = ITE2(K)
     DO 212 I=I1,I2
     DSI           = SO(I+1,K)  -SC(I-1,K)
     DSK           = SO(I,K+1)  -SC(I,K-1)
     SX            = A1(I)*DSI
     SZ            = C1(K)*DSK
     DGI           = G(I+1,KY,K)  -G(I-1,KY,K)
     DGK           = G(I,KY,K+1)-G(I,KY,K-1)
       R   = AMINC(1,IV(I,K))
       A   = 1.0 -R + AO(I)*AO(I) +SO(I,K)*SO(I,K)
       H = R/A
       FH = R*A
       AZ = -AO(I)*XZ(K) - SO(1,K)*YZ(K)
       BZ = -AO(I)*YZ(K) + SC(I,K)*XZ(K)
       HZ = AZ *SX -BZ +FH *SZ
       FYY = 1.0 +SX*SX +H*HZ*HZ
       FXY = SX + H*AZ*HZ
     U             = A1(I)*DGI  +CA*AC(I)  +SA*SO(I,K)
     W             = C1(K)*DGK  +SYAW +CA *XZ(K) +SA*YZ(K)
     V             = SA*AO(I)  - CA*SO(I,K)
```

```
212 G(I,KY+1,K) = G(I,KY-1,K)
   1        +(V*(1.0 -H*BZ*HZ)-U*FXY-W*HZ)/(FYY*B1(KY))
    NO          = NO  +1
    EO(NO)      = G(I2,KY,K)  -G(I1,KY,K)
    N           = NO
    I           = I1
    IF (K.NE.KTE2.OR.YAw.LE.0.) GO TO 231
221 I           = I  +1
    M           = NX  +2  -I
    NO          = NO  +1
    EO(NO)      = G(M,KY,K)  -G(I,KY,K)
    IF (I.LT.MXO) GO TO 221
    I           = I1
231 I           = I  -1
    E           = 0.
    IF (IV(I,K).NE.1) GO TO 237
    ZZ          = Z(K)  -TYAW*(XC(K)  +S1*AO(I)*AO(I))
233 IF (ZZ.GE.ZO(N-1)) GO TO 235
    N           = N  -1
    GO TO 233
235 R           = (ZZ  -ZO(N-1))/(ZO(N)  -ZO(N-1))
    E           = R*EO(N)  +(1.  -R)*EC(N-1)
237 M           = NX  +2  -I
    G(I,KY+1,K) = G(M,KY-1,K)  -E
    G(M,KY+1,K) = G(I,KY-1,K)  +E
    IF (IV(I,K).NE.-1) GO TO 241
    G(I,KY,K)   = .5*G(I,KY,K-1)+.25*(G(I,KY,K+1)+G(M,KY,K+1))
    IF (IV(I,K+1).LT.1)
   1G(I,KY,K)   = .5*G(I,KY,K+1)+.25*(G(I,KY,K-1)+G(M,KY,K-1))
    G(M,KY,K)   = G(I,KY,K)
    G(I,KY-1,K) = .5*(G(I,KY,K)  +G(I,KY-2,K))
    G(M,KY-1,K) = .5*(G(M,KY,K)  +G(M,KY-2,K))
241 IF (I.GT.2) GO TO 231
251    CONTINUE
261 EO(NO+1)  = 0.
    RETURN
    END
```

No. Grid Points				CPU Time Per Grid Point		STAR Vector Length	Ratio of CY 175 CPU Time to STAR CPU Time
X	Y	Z	Total	CY 175	STAR		
96	8	16	12280	1.4×10^{-4}	7.5×10^{-5}	96	1.87
192	16	32	98304	1.4×10^{-4}	$5.\times10^{-5}$	192	2.8
150	8	16	19200	1.4×10^{-4}*	6.5×10^{-5}	150	2.1
300	16	32	153600	1.4×10^{-4}*	4.5×10^{-5}	300	3.1

Total Time = (Time per grid pt.) × (No. of grid pts.) × (No. of iterations)

*This case has not been run on the CYBER computer because of storage and total time limitations, however, CPU time per grid point is the same as that for smaller cases.

Table 1 (NACA 65_1A012 wing)

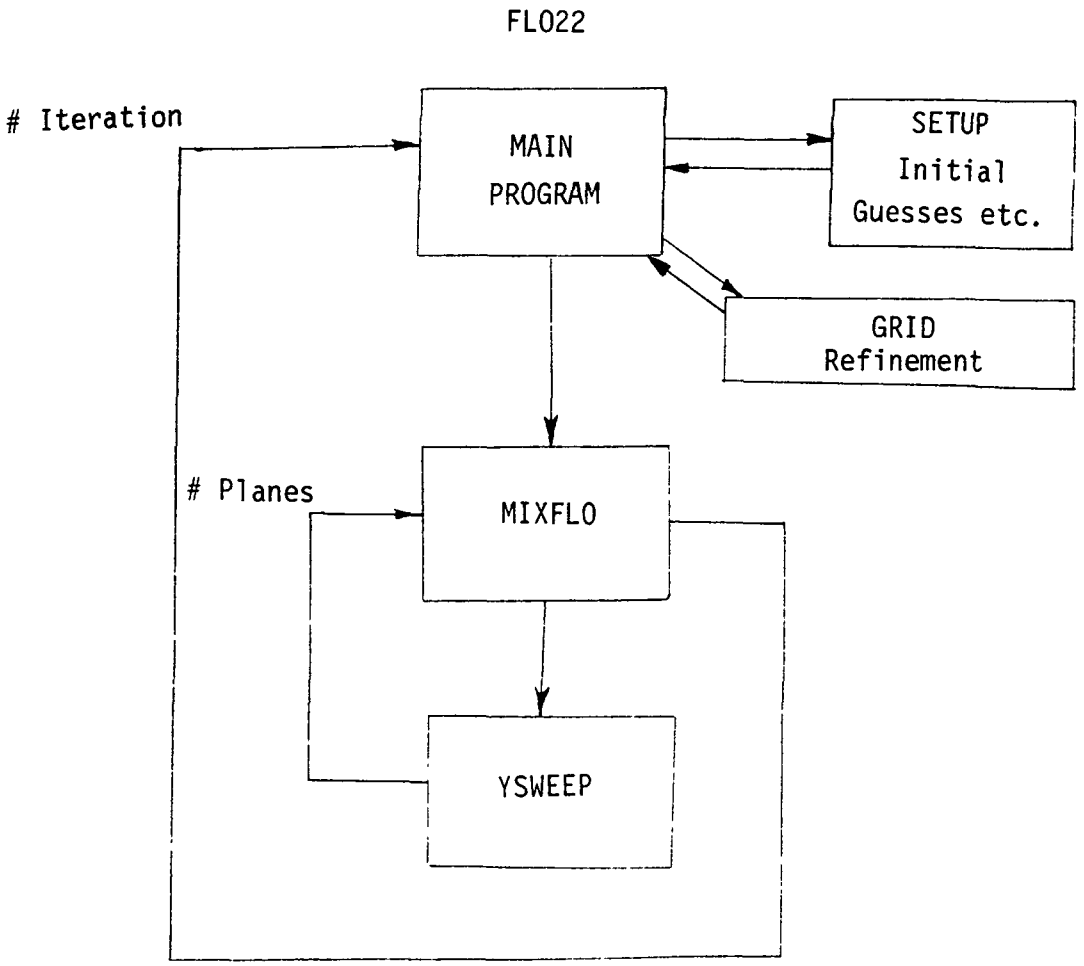

Figure 1 (General organization of FLO-22-V1)

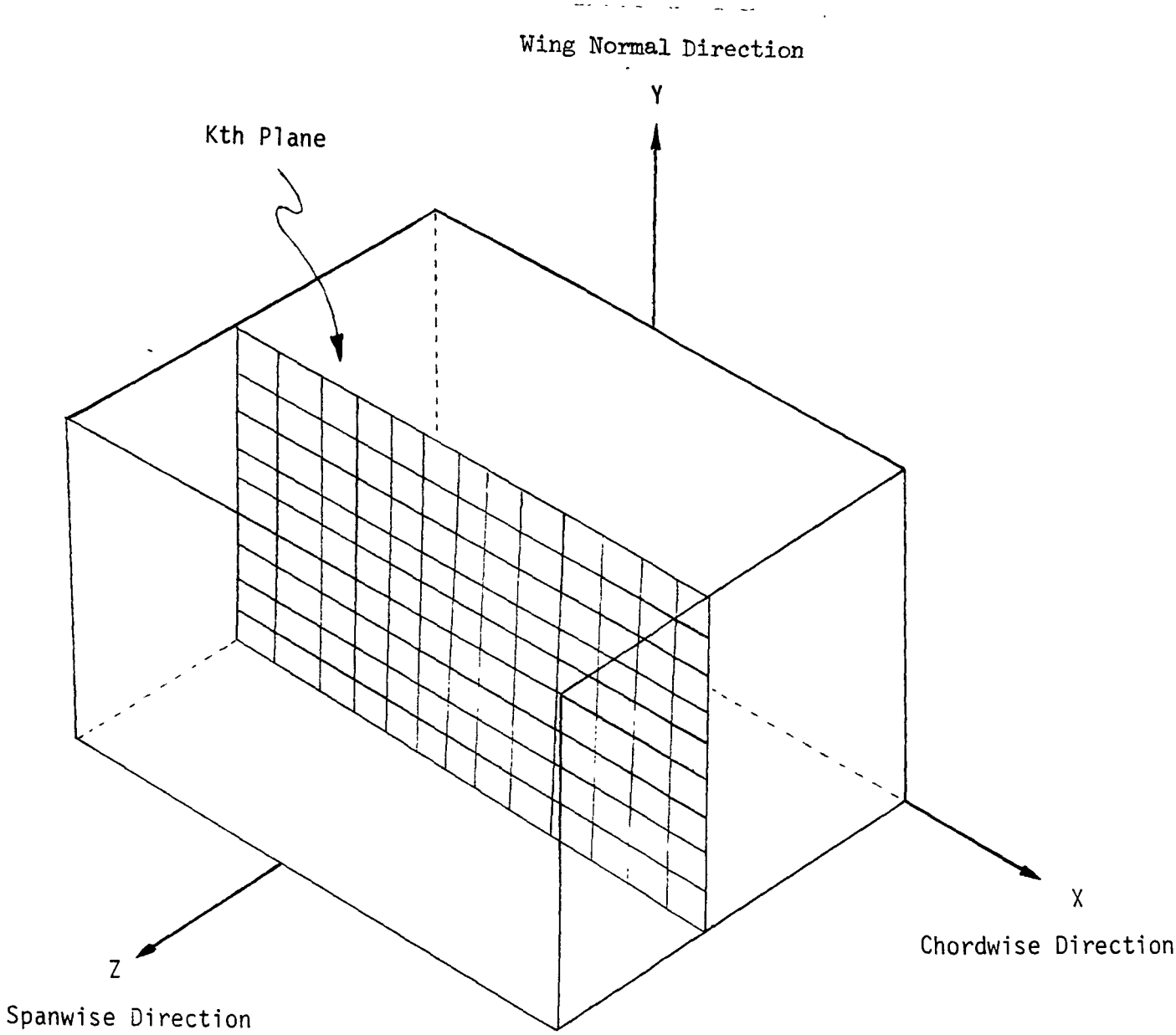

Figure 2 - Computational Domain

REFERENCES

1. Jameson, Antony; Caughey, D. A.; Newman, P. A.; and Davis, R. M.: A Brief Description of the Jameson–Caughey NYU Transonic Swept–Wing Computer Program FLO–22; NASA TMX–73996, 1976.

2. Jameson, Antony; Caughey, D. A.: Numerical Calculation of the Transonic Flow Past a Swept–Wing. ERDA Research and Development Report COO–3077–140, 1977.

3. Jameson, Antony: Transonic Flow Calculations Presented at the Lecture Series on Computational Fluid Dynamic, March 15–19, 1976, Von Karman Institute, Rhode St. Genese, Belgium. (Lecture notes to be published).

4. Control Data Corporation: "Control Data STAR–100 Computer Hardware Reference Manual," Publication Number 60156000, December 1975.

5. Lambiotte, J. J.; Voigt, R. G.: The Solution of Tridiagonal Linear Systems on the CDC STAR–100 Computer. ACM Transactions on Mathematical Software, Vol. 1, No. 4, December 1975.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-78665		

4. Title and Subtitle	5. Report Date
A VECTORIZATION OF THE JAMESON-CAUGHEY NYU TRANSONIC SWEPT-WING COMPUTER PROGRAM FLO-22-V1 FOR THE STAR-100 COMPUTER	March 6, 1978
	6. Performing Organization Code
7 Author(s) R. E. Smith, J. I. Pitts, and J. J. Lambiotte	8. Performing Organization Report No.
	10. Work Unit No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No
	13. Type of Report and Period Covered
12 Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D. C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The computer program FLO-22 for analyzing inviscid isentropic transonic flow past 3-D swept-wing configurations has been modified to use vector operations and run on the STAR-100 computer. The "vectorized version" described herein is called FLO-22-V1. Vector operations have been incorporated into Successive Line Over-Relaxation in the transformed horizontal direction. Vector relational operations and control vectors are used to implement upwind differencing at supersonic points. A high speed of computation and extended grid domain are characteristics of FLO-22-V1. The new program is not the optimal vectorization of Successive Line Over-Relaxation applied to transonic flow; however, it proves that vector operations can readily be implemented to increase the computation rate of the algorithm.

17 Key Words (Suggested by Author(s)) (STAR category underlined)	18 Distribution Statement
	Unclassified - Unlimited

19 Security Classif. (of this report)	20. Security Classif. (of this page)	21 No of Pages	22 Price*
Unclassified	Unclassified	30	$4.00

*Available from { The National Technical Information Service, Springfield, Virginia 22151 / STIF/NASA Scientific and Technical Information Facility, P.O. Box 33, College Park, MD 20740