Helsinki University of Technology
CFD-group / Laboratory of Applied Thermodynamics

MEMO No CFD/TERMO-6-95 DATE: December 17, 1995

TITLE

Automatic CFD input decomposition for massively parallel systems

AUTHOR(S)
Esa Salminen

ABSTRACT

This memorandum describes a computer program which divides CFD grids into smaller blocks and updates the boundary condition information and the flow solver control data accordingly. The program is designed to be used in conjunction with FINFLO flow solver.

MAIN RESULT

Computer program **divp3d**

PAGES
22

KEY WORDS
CFD input, massively parallel systems

APPROVED BY

Timo Siikonen December 17, 1995

Contents

1 Introduction

Numerical flow simulations on massively parallel systems require computation domain decomposition. In simple cases this can be done by hand, but when the complexity of the simulation model increases, decomposition must be automated.

This memorandum describes a computer program which divides CFD grids into smaller blocks and updates the boundary condition information and the flow solver control data accordingly. The program is designed to be used in conjunction with FINFLO flow solver. FINFLO accepts grid files in multi-block PLOT3D format.

The program can be used in two different modes. In the manual mode, the user specifies explicitly the division nodes for each block and each coordinate direction. In the automatic mode, the user specifies only the desired sub-block edge length and the program tries to extract this size blocks from the original grid.

The program has been written in FORTRAN-77 and consists of 3400 lines. The program was compiled and tested on a Silicon Graphics Indigo^2 running IRIX 5.3.

2 Grid splitting

A very simple algorithm is used in the automatic grid splitting mode. The splitting strategy is as follows: If the block edge dimension is smaller than the desired one, no cutting will take place. If the block edge dimension is larger than the desired one, but smaller than twice the desired size, a cutting line from middle of the block face will be chosen. (This could be improved by trying to leave the larger block on the solid wall side. This would improve the behavior of turbulence models). If the block edge dimension is larger than twice the desired size, but cannot be equally distributed, the smaller block will be cut from middle of the original block, i.e. the block next to the possible solid wall is always as large as possible.

The block is always fully cut, both in the manual mode and in the automatic mode. In the manual mode the block boundaries (i=1, i=imax, j=1, j=jmax, k=1, k=kmax) are automatically included into the cutting point arrays. User can omit these points, but giving them does not make any harm either.

The order of the extracted blocks is the same as the order of the cells in the grid, i.e., the sub-block cutter runs fastest in the I-direction, then in the J-direction and slowest in the K-direction.

In the current version the whole grid is first read in and then splitted. It would be possible to handle the grid block by block as well. This modification may be needed for systems with limited main memory.

3 Boundary condition patch splitting

The boundary condition patch splitting is a complex task in comparison to the grid block splitting. This is true especially in cases where two connected blocks will not be cut at the same locations. That is why the algorithm divides the boundary patches using cutting lines from both blocks. First the limits of the new boundary condition pathes will be computed. As a separate step, the connectivity information will be updated. Connectivity means here also cyclic and sliding mesh boundary conditions.

According to the FINFLO's naming and numeration conventions the block faces are numbered as shown in Fig. 1.

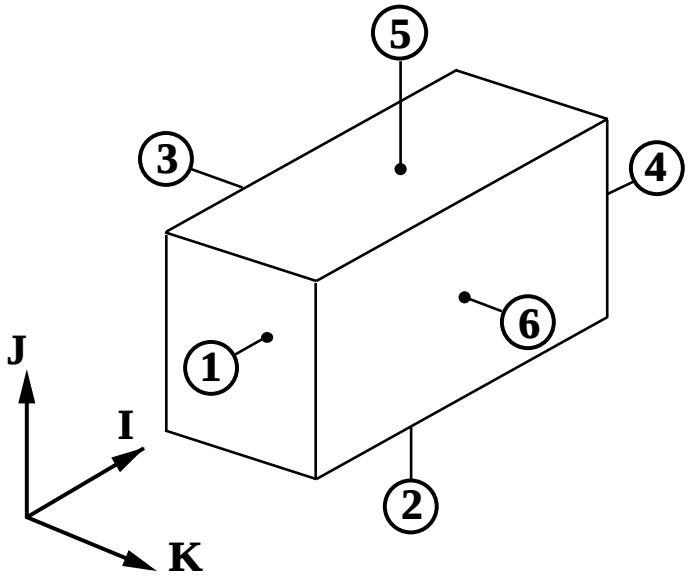

Fig. 1: FINFLO's numeration of block faces.

A do-loop for determining boundaries for a splitted BC patch is shown below. The additional cutting lines from the connected patches increase the complexity of the loop. Arrays `IPTS()`, `JPTS()`, and `KPTS()` contain cutting lines. Arrays `IPTR()`, `JPTR()`, and `KPTR()` contain the original cutting lines, i.e. the cutting lines only from the current block.

```
IB = ICON(2,IP)  ! Current block number
JC = 0
DO I=1,IB-1
  JC = JC + LMAX(I)*MMAX(I)*NMAX(I)
ENDDO

IF(ICON(3,IP) .EQ. 1) THEN  ! Face No. 1
  II = 1
  KK = 1
  DO K=2,KPTS(1,IB)
    IF(KPTR(KK+1,IB) .EQ. KPTS(K,IB)) KK = KK + 1
    JJ = 1
    DO J=2,JPTS(1,IB)
      IF(KPTR(JJ+1,IB) .EQ. KPTS(J,IB)) JJ = JJ + 1
      JB = JC + (KK-2)*LMAX(IB)*MMAX(IB)+(JJ-2)*LMAX(IB)+II
      IXLO = MAX0(ICON(4,IP),JPTS(J,  IB)  )
      IXUP = MIN0(ICON(5,IP),JPTS(J+1,IB)-1)
      IYLO = MAX0(ICON(6,IP),KPTS(K,  IB)  )
      IYUP = MIN0(ICON(7,IP),KPTS(K+1,IB)-1)
      IF(IXUP-IXLO .GE. 0 .AND. IYUP-IYLO .GE. 0) THEN
         IXLO = IXLO - JPTR(JJ,IB) + 1
         IXUP = IXUP - JPTR(JJ,IB) + 1
         IYLO = IYLO - KPTR(KK,IB) + 1
         IYUP = IYUP - KPTR(KK,IB) + 1
         JNBCS = JNBCS + 1
         JCON( 1,JNBCS) = ICON(1,IP)
         JCON( 2,JNBCS) = JB
```

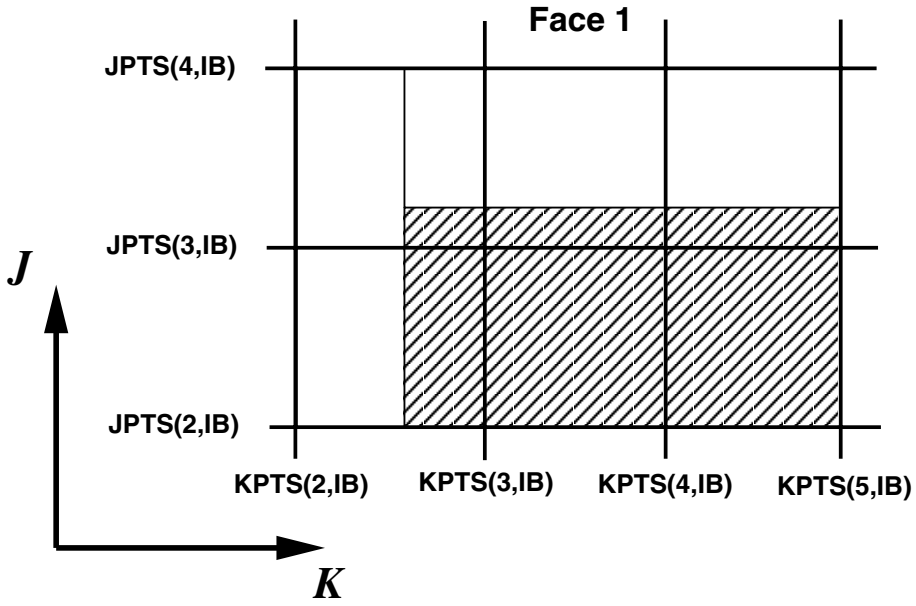

Fig. 2: BC patch splitting.

```
            JCON( 3,JNBCS) = ICON(3,IP)
            JCON( 4,JNBCS) = IXLO
            JCON( 5,JNBCS) = IXUP
            JCON( 6,JNBCS) = IYLO
            JCON( 7,JNBCS) = IYUP
            JCON( 8,JNBCS) = 0
            JCON( 9,JNBCS) = 0
            JCON(10,JNBCS) = 0
            IF(ICON(1,IP) .EQ. 6) THEN
              DO L=1,9
                BBB(L,JNBCS) = AAA(L,IP)  ! Cyclicity matrix
              ENDDO
            ENDIF
            IF(ICON(1,IP) .EQ. 3 .OR. ICON(1,IP) .EQ. 5) THEN
              BOUNDG(JNBCS) = BOUNDF(IP)  ! INL/OUT file name
            ENDIF
          ENDIF
        ENDDO
      ENDDO
    ENDIF

C ... Faces 2, 3, 4, 5, and 6 accordingly
```

The most challencing task in the BC patch splitting process is the determination of the additional cutting lines from connected patches. When we calculate these lines, we must consider which faces are connected and what is the orientation between the blocks. An additional difficulty comes from the relative position of the blocks. Since there are six faces on both blocks and four possible orientations, we have $6 \times 6 \times 4 = 144$ combinations. The right case can be found by computing a magic number

`MAGIC = IORI+(JFACE-1)*4+(IFACE-1)*24+1`,

where the face numbers can have values from one to six and the orientation can have values from zero to three.

The cutting point arrays are merged in the subroutine `AP`. The right arrays and the right order of the arrays are established in the subroutine call. The parameters `IR` and `JR` define if the cutting point distributions must be reversed or not. The last two parameters `M` and `N` determine the relative position of the connected patches.

```
      IB    = ICON(2,IP)  ! block number
      IFACE = ICON(3,IP)  ! face number
      JB    = ICON(2,JP)  ! block number - connected block
```

```
      JFACE = ICON(3,JP)  ! face number - connected block

      IF(IORI .EQ. 0) THEN
        IR = 0
        JR = 0
      ELSEIF(IORI .EQ. 1) THEN
        IF(IFACE .EQ. 1 .OR. IFACE .EQ. 3 .OR. IFACE .EQ. 5) THEN
          IR = 0
          JR = 1
        ELSE
          IR = 1
          JR = 0
        ENDIF
      ELSEIF(IORI .EQ. 2) THEN
        IR = 1
        JR = 1
      ELSEIF(IORI .EQ. 3) THEN
        IF(IFACE .EQ. 1 .OR. IFACE .EQ. 3 .OR. IFACE .EQ. 5) THEN
          IR = 1
          JR = 0
        ELSE
          IR = 0
          JR = 1
        ENDIF
      ENDIF

      IF(MOD(IFACE+JFACE+IORI,2) .EQ. 0) THEN
        M = ICON(6,JP) - ICON(4,IP)
        N = ICON(4,JP) - ICON(6,IP)
        IF(IR .EQ. 1) M = ICON(7,JP) + ICON(4,IP)
        IF(JR .EQ. 1) N = ICON(5,JP) + ICON(6,IP)
      ELSE
        M = ICON(4,JP) - ICON(4,IP)
        N = ICON(6,JP) - ICON(6,IP)
        IF(IR .EQ. 1) M = ICON(5,JP) + ICON(4,IP)
        IF(JR .EQ. 1) N = ICON(7,JP) + ICON(6,IP)
      ENDIF

      MAGIC = IORI+(JFACE-1)*4+(IFACE-1)*24+1

      GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
     &      22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,
     &      40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,
     &      58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,
     &      76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,
     &      94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,
     &      109,110,111,112,113,114,115,116,117,118,119,120,121,
     &      122,123,124,125,126,127,128,129,130,131,132,133,134,
     &      135,136,137,138,139,140,141,142,143,144),MAGIC

    1 CONTINUE
        CALL AP(JPTS(1,IB),KPTS(1,IB),KPTS(1,JB),JPTS(1,JB),0,0,M,N)
        RETURN
    2 CONTINUE
        CALL AP(JPTS(1,IB),KPTS(1,IB),JPTS(1,JB),KPTS(1,JB),0,1,M,N)
        RETURN
    3 CONTINUE
        CALL AP(JPTS(1,IB),KPTS(1,IB),KPTS(1,JB),JPTS(1,JB),1,1,M,N)
        RETURN
    4 CONTINUE
        CALL AP(JPTS(1,IB),KPTS(1,IB),JPTS(1,JB),KPTS(1,JB),1,0,M,N)
        RETURN

        .
        .
        .

  143 CONTINUE
        CALL AP(IPTS(1,IB),JPTS(1,IB),JPTS(1,JB),IPTS(1,JB),1,1,M,N)
        RETURN
  144 CONTINUE
        CALL AP(IPTS(1,IB),JPTS(1,IB),IPTS(1,JB),JPTS(1,JB),0,1,M,N)
        RETURN
```

4 Solver control data

FINFLO's control file can be divided into two parts. The upper part contains general information like iteration control, Courant numbers, Reynolds number, Mach number, etc. The lower part contains information specific for each grid block.

In the input decomposition process the upper part is kept unchanged. In the block specific part the number of blocks and the block dimensions are updated and all other information is copied from original blocks to the corresponding extracted sub-blocks. The only parameter that requires slightly more attention, is the number of multi-grid levels. Since the block is divided, the number of available multi-grid levels is probable smaller than in the original block. The number of multigrid-levels is obtained by

```
        read( 13,*) mgrid  ! MGRID from the original INPUT file
          do mg = 1,5
            if(mod(idmn(1),2**(mg-1)) .eq. 0 .and.
     &         mod(jdmn(1),2**(mg-1)) .eq. 0 .and.
     &         mod(kdmn(1),2**(mg-1)) .eq. 0) mgn = mg
          enddo
          mgrid = min0(mgrid,mgn)
```

A typical FINFLO control file is shown below.

```
Ahlstrom impeller (Viscous flow; New BC; Wall 4 ROT)
      1       2      1        0     |   IOLD    LEVEL     ITURB    NSCAL
/afrodite/tmp/kpelkone/finflo/ahlviscous/reunaehdot/AHL_KITKA.GRID
/afrodite/tmp/kpelkone/finflo/ahlviscous/reunaehdot/AHL_KITKA.BC
  'ROE'   'YES'  'NO'  'NO'   'NO'  |   IFLX   RESTART  STRESC  FULLFC SOURC
  'NO'    'NO'   'NO'               |   TIMEC   COORC  PRESC
  2.5  2.5     1.e-8  64368.  1.0   |   CFL-NUMBERS   DROLIM  TMAX    DT1
  1305     100        5000          |   ICMAX         KPRINT    MPRINT
  0.000   0.000    00.0    0.0      |   RMACH   ALPHA  BETA    ROTAT
  0.      0.72    0.90               |   RE   PRANDTLIN LUVUT(PR;PRT)
  1       'YES'                      |   ISTATE    DEFAULT PERF.GAS
  288.15    1.2249       85.07      |   FRSTEM    FRSDEN   FRSVEL
    0.001   0.01         5000.      |   TURBFRS   RMUFRS    TURLIM
    0.001   1.00   0.1   0.2   5.   |   TURBINI   RMUINI    CMGK CMGEPS  CSIMPS
    0       0      0     0          |   JRDIF     JRDIS   JRPRE   JRIMP
    0.1     1.                      |   CC1    CC2
  0.176521     1.00        1.0      |   AREF  CHLREF (initialisointi) GRILEN
  0.     0.    0.     0.  -1.       |   XMOM  YMOM   ZMOM  REFPRE  DIFPRE
  3    2       0                    |   NUMBER OF BLOCKS   NOF INLETS/OUTLETS
 48  112  48   -1 -1 -1  111   10   |   IMAX JMAX KMAX INTER(IJK) LAMIN BLOCK 1
 106   1  24   49  113  49     0    |   IT   IL   IK    IDI1   IDI2  IDI3  MOV
  2    1  48    1  112   1 48       |   MGRID  MIB MIT   MJB MJT   MKB MKT
 -115.8288   2                     |   OMEGA   IROTVE
 48    2  48   -1 -1 -1  111   10   |   IMAX JMAX KMAX INTER(IJK) LAMIN  BLOCK 2
 00    3  08   49  3    49     0    |   IT   IL   IK    IDI1   IDI2  IDI3  MOV
  1    1  48    1  2     1 48       |   MGRID  MIB MIT   MJB MJT   MKB MKT
 -115.8288   2                     |   OMEGA   IROTVE
 48    2  48   -1 -1 -1  111   10   |   IMAX JMAX KMAX INTER(IJK) LAMIN  BLOCK 3
 00    1  26   49  3    49     0    |   IT   IL   IK    IDI1   IDI2  IDI3  MOV
  1    1  48    1  2     1 48       |   MGRID  MIB MIT   MJB MJT   MKB MKT
 -115.8288   2                     |   OMEGA   IROTVE
```

When handling the control file, the program assumes that the comments at the end of the input lines do not start earlier than at the 36th column.

5 Sample runs

This first example illustrates program's capability to handle complex connections between blocks. This two-block grid is purely fictive and does not present any reasonable CFD geometry. The original grid and the splitted grid are shown in Fig. 3.

Note that in order to increase the complexity of the splitting task, the larger block is rotated about the y-axis so that its K-direction coincides with the smaller block's I-direction.

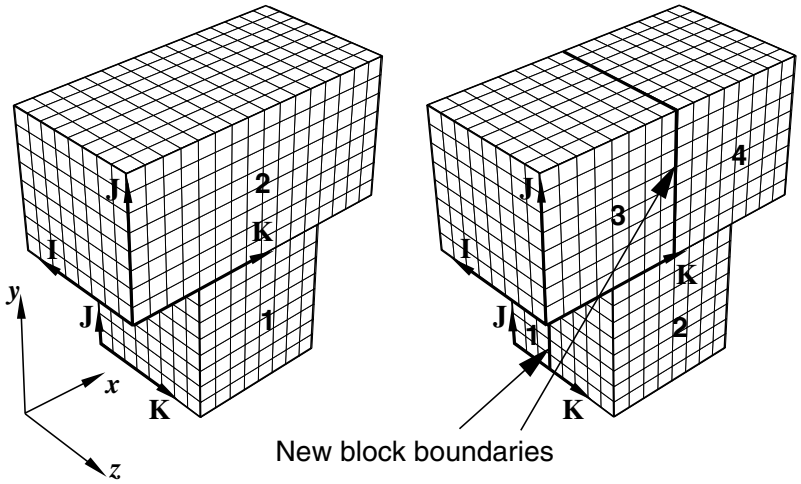

Fig. 3: Original two-block grid and the splitted four-block grid.

Both blocks will be divided in the K-direction at one location. In the larger block this location is the 9th node and in the smaller block the 3rd node. However, due to the rotation of the larger block, these cuttings are not parallel but perpenticularly crossing. Further, the 9th node in the larger block's K-direction coincides with the 5th node in the smaller block's I-direction and the smaller block's 3rd node in the K-direction coincides with the larger block's 5th node in the I-direction. This is why the connective patch on both blocks must be divided into four pieces. A screen copy of the program run, the original BC data file, and the decomposed BC data file are shown in appendix A.

This second example illustrates program usage in automatic mode. The grid used in this test is a one-block O-O type $128 \times 64 \times 64$ delta wing. The grid is shown in Fig. 4.

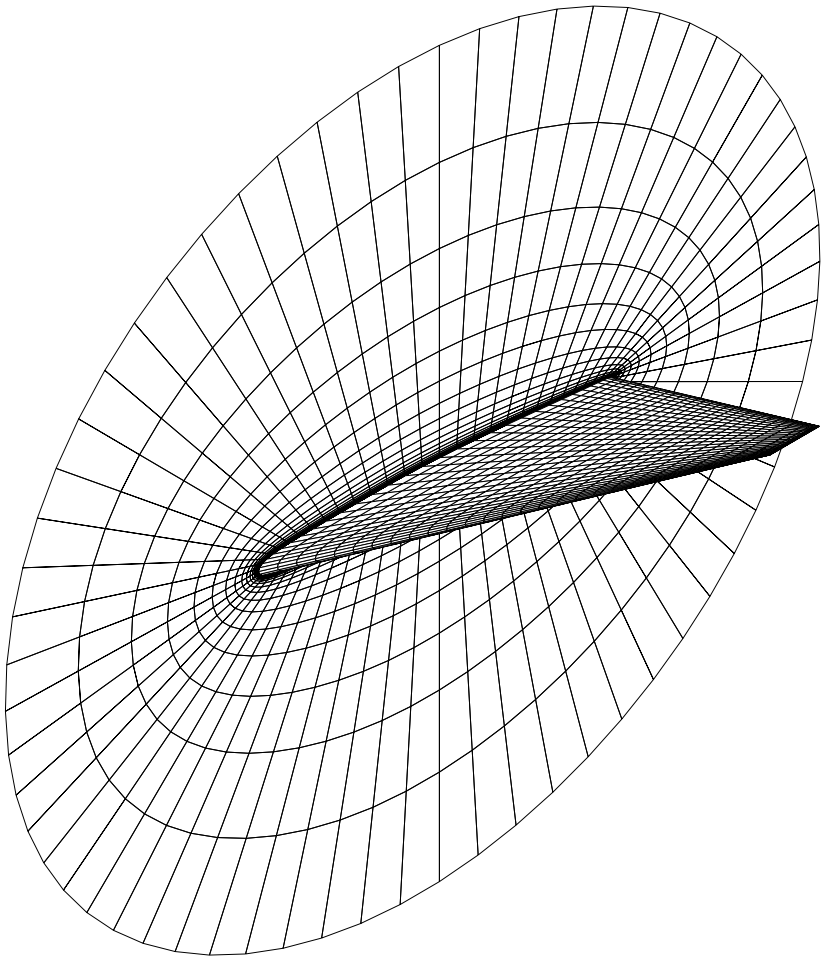

Fig. 4: Delta wing. (2nd finest grid level, symmetry plane grid drawn only partly.)

In the first run we give 32 as a desired sub-block edge length. The grid size is ideal for this type of partitioning. As a result we get sixteen $32 \times 32 \times 32$ blocks. In the second run we give 48 as a desired sub-block edge length. Now the program fails to extract any $48 \times 48 \times 48$ blocks but gives eight $48 \times 32 \times 32$ blocks and four $32 \times 32 \times 32$ blocks.

Screen copies from both runs, the original BC data file, and the decomposed BC data files are shown in appendix B.

6 Concluding remarks

A CFD input decomposition software has been written. The current version can handle correctly typical FINFLO inputs. However, there is still room for improvements.

The program should write a log how the grid was splitted so that after completing the flow simulation, the grid and the flow solution could be merged into the original form.

The automatic grid block splitting algorithm could be improved. It is probable, however, that the manual mode will be sufficient for most applications in the near future. During the testing process it was seen that the grid was easy to split using the manual mode even in cases where the desired number of blocks was several hundreds.

Overlapping grid blocks may need some special treatment in the future. This will be seen as soon as Chimera type grids become more common in FINFLO simulations.

Splitting of a sliding mesh patch in the circumferential direction is not allowed. This limitation is not implemented in program's current version. Therefore, it is recommended that all grids containing sliding patches should be splitted using program's manual mode.

Handling of boundary value files, i.e. special files used in conjunction with INLET- and OUTLET-blocks, should be implemented.

A Sample run I/O - fictive grid

Sample run screen copy - fictive grid

```
POSEIDON> divp3d

 ***************************************************
 ***************************************************

 DIVP3D   Version 1.0  Nov, 1995  running on POSEIDON

 ***************************************************
 ***************************************************


 Please enter the input grid file name:
RUBIK.GRID

 Please enter the input boundary condition file name:
RUBIK.BC

 Please enter the original INPUT file name:
INPUT

 Please enter the output grid file name:
SPLITTED.GRID

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter the output boundary condition file name:
SPLITTED.BC

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter file name for the updated INPUT:
SPLITTED.INPUT

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2


 DIVP3D:  STARTING GRID:

          Number of grid blocks NB =            2

          IDMN(  1) =   9, JDMN(  1) =   9, KDMN(  1) =   9
          IDMN(  2) =   9, JDMN(  2) =   9, KDMN(  2) =  17


 DIVP3D:  Reading original BC data: RUBIK.BC

 DIVP3D:  Boundary conditions include INL and/or OUT.


 MANUAL MODE (=0) OR  AUTOMATIC MODE (=1) ?
0

 Block No.  1
 Dimensions of this block are (NI,NJ,NK) =     9    9    9
 Give the cutting points (nodes) for i-direction
1 9
 Give the cutting points (nodes) for j-direction
1 9
 Give the cutting points (nodes) for k-direction
1 3 9

 Block No.  2
 Dimensions of this block are (NI,NJ,NK) =     9    9   17
 Give the cutting points (nodes) for i-direction
1 9
 Give the cutting points (nodes) for j-direction
1 9
```

```
 Give the cutting points (nodes) for k-direction
1 9 17


 DIVP3D:  DIVIDED GRID:

          Number of grid blocks NC =            4

          IDMN(  1) =   9, JDMN(  1) =   9, KDMN(  1) =   3
          IDMN(  2) =   9, JDMN(  2) =   9, KDMN(  2) =   7
          IDMN(  3) =   9, JDMN(  3) =   9, KDMN(  3) =   9
          IDMN(  4) =   9, JDMN(  4) =   9, KDMN(  4) =   9

 DIVP3D:  Starting grid NNODE =         2106
 DIVP3D:  Target   grid NNODE =         2268

 DIVP3D:  Requesting        52488 bytes of memory

 DIVP3D:  Memory successfully allocated ...

 DIVP3D:  Reading original grid: RUBIK.GRID

 DIVP3D:  Writing splitted grid: SPLITTED.GRID

 DIVP3D:  Writing splitted BC data: SPLITTED.BC


 DIVP3D:  Reading the original INPUT: INPUT
  ... and writing the splitted INPUT: SPLITTED.INPUT

POSEIDON>
```

Original BC data - fictive grid

```
Two-block test grid
2
------------------------------ BLOCK 1 ---------------------------------
8 8 8
1 1
SOL
2 1
SOL
3 1
SOL
4 1
SOL
5 1
CON 1 8 1 8 2 2 2
6 1
SOL
------------------------------ BLOCK 2 ---------------------------------
8 8 16
1 1
SOL
2 3
SOL 1 8  1  4
CON 1 8  5 12 1 5 1
SOL 1 8 13 16
3 1
INL inlet.dat
4 1
MIR
5 1
EXT
6 1
SOL
```

Splitted BC data - fictive grid

```
Two-block test grid - decomposed
  4
--------   Block No.    1   --------
  8   8   2
1  1
SOL    1   8   1   2
2  1
SOL    1   8   1   2
3  1
SOL    1   8   1   8
4  1
SOL    1   8   1   2
5  2
CON    1   4   1   2      3   2   3
CON    5   8   1   2      4   2   2
6  1
CON    1   8   1   8      2   3   1
--------   Block No.    2   --------
  8   8   6
1  1
SOL    1   8   1   6
2  1
SOL    1   8   1   6
3  1
CON    1   8   1   8      1   6   1
4  1
SOL    1   8   1   6
5  2
CON    1   4   1   6      3   2   2
CON    5   8   1   6      4   2   1
6  1
SOL    1   8   1   8
--------   Block No.    3   --------
  8   8   8
1  1
SOL    1   8   1   8
2  3
SOL    1   8   1   4
CON    1   6   5   8      2   5   1
CON    7   8   5   8      1   5   1
3  1
INL inlet.dat
4  1
MIR    1   8   1   8
5  1
EXT    1   8   1   8
6  1
CON    1   8   1   8      4   3   1
--------   Block No.    4   --------
  8   8   8
1  1
SOL    1   8   1   8
2  3
CON    1   6   1   4      2   5   2
CON    7   8   1   4      1   5   2
SOL    1   8   5   8
3  1
CON    1   8   1   8      3   6   1
4  1
MIR    1   8   1   8
5  1
EXT    1   8   1   8
6  1
SOL    1   8   1   8
```

B Sample run I/O - delta wing

Sample run screen copy - delta wing - desired block edge length 32

```
POSEIDON> ../finflonkehitys/divp3d

 ***************************************************
 ***************************************************

 DIVP3D   Version 1.0  Nov, 1995  running on POSEIDON

 ***************************************************
 ***************************************************


 Please enter the input grid file name:
DELTA.GRID

 Please enter the input boundary condition file name:
DELTA.BC

 Please enter the original INPUT file name:
INPUT

 Please enter the output grid file name:
SPLITTED.GRID

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter the output boundary condition file name:
SPLITTED.BC

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter file name for the updated INPUT:
SPLITTED.INPUT

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2


 DIVP3D:  STARTING GRID:

          Number of grid blocks NB =            1

          IDMN(  1) = 129, JDMN(  1) =  65, KDMN(  1) =  65


 DIVP3D:  Reading original BC data: DELTA.BC


 MANUAL MODE (=0) OR  AUTOMATIC MODE (=1) ?
1


 Desired sub-block edge length ?
32


 DIVP3D:  DIVIDED GRID:

          Number of grid blocks NC =           16

          IDMN(  1) =  33, JDMN(  1) =  33, KDMN(  1) =  33
          IDMN(  2) =  33, JDMN(  2) =  33, KDMN(  2) =  33
          IDMN(  3) =  33, JDMN(  3) =  33, KDMN(  3) =  33
          IDMN(  4) =  33, JDMN(  4) =  33, KDMN(  4) =  33
          IDMN(  5) =  33, JDMN(  5) =  33, KDMN(  5) =  33
          IDMN(  6) =  33, JDMN(  6) =  33, KDMN(  6) =  33
          IDMN(  7) =  33, JDMN(  7) =  33, KDMN(  7) =  33
          IDMN(  8) =  33, JDMN(  8) =  33, KDMN(  8) =  33
          IDMN(  9) =  33, JDMN(  9) =  33, KDMN(  9) =  33
```

```
          IDMN( 10) =  33, JDMN( 10) =  33, KDMN( 10) =  33
          IDMN( 11) =  33, JDMN( 11) =  33, KDMN( 11) =  33
          IDMN( 12) =  33, JDMN( 12) =  33, KDMN( 12) =  33
          IDMN( 13) =  33, JDMN( 13) =  33, KDMN( 13) =  33
          IDMN( 14) =  33, JDMN( 14) =  33, KDMN( 14) =  33
          IDMN( 15) =  33, JDMN( 15) =  33, KDMN( 15) =  33
          IDMN( 16) =  33, JDMN( 16) =  33, KDMN( 16) =  33

DIVP3D:  Starting grid NNODE =       545025
DIVP3D:  Target   grid NNODE =       574992

DIVP3D:  Requesting     13440204 bytes of memory

DIVP3D:  Memory successfully allocated ...

DIVP3D:  Reading original grid: DELTA.GRID

DIVP3D:  Writing splitted grid: SPLITTED.GRID

DIVP3D:  Writing splitted BC data: SPLITTED.BC


DIVP3D:  Reading the original INPUT: INPUT
 ... and writing the splitted INPUT: SPLITTED.INPUT
```

Sample run screen copy - delta wing - desired block edge length 48

```
POSEIDON> ../finflonkehitys/divp3d

 ************************************************
 ************************************************

 DIVP3D   Version 1.0  Nov, 1995  running on POSEIDON

 ************************************************
 ************************************************


 Please enter the input grid file name:
DELTA.GRID

 Please enter the input boundary condition file name:
DELTA.BC

 Please enter the original INPUT file name:
INPUT

 Please enter the output grid file name:
SPLITTED.GRID

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter the output boundary condition file name:
SPLITTED.BC

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2

 Please enter file name for the updated INPUT:
SPLITTED.INPUT

 File already exists! What do you want to do?
  = 1 : Re-enter filename
  = 2 : Overwrite existing file
2


 DIVP3D:  STARTING GRID:

          Number of grid blocks NB =            1

          IDMN(  1) = 129, JDMN(  1) =  65, KDMN(  1) =  65


 DIVP3D:  Reading original BC data: DELTA.BC


 MANUAL MODE (=0) OR  AUTOMATIC MODE (=1) ?
1


 Desired sub-block edge length ?
48


 DIVP3D:  DIVIDED GRID:

          Number of grid blocks NC =           12

          IDMN(  1) =  49, JDMN(  1) =  33, KDMN(  1) =  33
          IDMN(  2) =  33, JDMN(  2) =  33, KDMN(  2) =  33
          IDMN(  3) =  49, JDMN(  3) =  33, KDMN(  3) =  33
          IDMN(  4) =  49, JDMN(  4) =  33, KDMN(  4) =  33
          IDMN(  5) =  33, JDMN(  5) =  33, KDMN(  5) =  33
          IDMN(  6) =  49, JDMN(  6) =  33, KDMN(  6) =  33
          IDMN(  7) =  49, JDMN(  7) =  33, KDMN(  7) =  33
          IDMN(  8) =  33, JDMN(  8) =  33, KDMN(  8) =  33
          IDMN(  9) =  49, JDMN(  9) =  33, KDMN(  9) =  33
          IDMN( 10) =  49, JDMN( 10) =  33, KDMN( 10) =  33
          IDMN( 11) =  33, JDMN( 11) =  33, KDMN( 11) =  33
          IDMN( 12) =  49, JDMN( 12) =  33, KDMN( 12) =  33
```

```
 DIVP3D:  Starting grid NNODE =       545025
 DIVP3D:  Target   grid NNODE =       570636

 DIVP3D:  Requesting     13387932 bytes of memory

 DIVP3D:  Memory successfully allocated ...

 DIVP3D:  Reading original grid: DELTA.GRID

 DIVP3D:  Writing splitted grid: SPLITTED.GRID

 DIVP3D:  Writing splitted BC data: SPLITTED.BC


 DIVP3D:  Reading the original INPUT: INPUT
  ... and writing the splitted INPUT: SPLITTED.INPUT

POSEIDON>
```

Original BC data - delta wing

```
Delta wing
1
------------------------------  1. blokki ------------------------------
128 64 64
1 1
CON 1 1 1 1  1 4 1
2 1
SOL
3 1
MIR
4 1
CON 1 1 1 1  1 1 1
5 1
EXT
6 2
CON  1  64  1 64  1 6 2
CON 65 128  1 64  1 6 1
```

Splitted BC data - delta wing - desired block edge length 32

```
Delta wing - decomposed
 16
--------   Block No.    1   --------
 32  32  32
1  1
CON    1  32   1  32      4   4   1
2  1
SOL    1  32   1  32
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      2   1   1
5  1
CON    1  32   1  32      5   2   1
6  1
CON    1  32   1  32      9   3   1
--------   Block No.    2   --------
 32  32  32
1  1
CON    1  32   1  32      1   4   1
2  1
SOL    1  32   1  32
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      3   1   1
5  1
CON    1  32   1  32      6   2   1
6  1
CON    1  32   1  32     10   3   1
--------   Block No.    3   --------
 32  32  32
1  1
CON    1  32   1  32      2   4   1
2  1
SOL    1  32   1  32
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      4   1   1
5  1
CON    1  32   1  32      7   2   1
6  1
CON    1  32   1  32     11   3   1
--------   Block No.    4   --------
 32  32  32
1  1
CON    1  32   1  32      3   4   1
2  1
SOL    1  32   1  32
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      1   1   1
5  1
CON    1  32   1  32      8   2   1
6  1
CON    1  32   1  32     12   3   1
--------   Block No.    5   --------
 32  32  32
1  1
CON    1  32   1  32      8   4   1
2  1
CON    1  32   1  32      1   5   1
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      6   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     13   3   1
--------   Block No.    6   --------
 32  32  32
1  1
CON    1  32   1  32      5   4   1
2  1
CON    1  32   1  32      2   5   1
3  1
```

```
MIR    1  32   1  32
4  1
CON    1  32   1  32      7   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     14   3   1
--------   Block No.    7   --------
 32  32  32
1  1
CON    1  32   1  32      6   4   1
2  1
CON    1  32   1  32      3   5   1
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      8   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     15   3   1
--------   Block No.    8   --------
 32  32  32
1  1
CON    1  32   1  32      7   4   1
2  1
CON    1  32   1  32      4   5   1
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      5   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     16   3   1
--------   Block No.    9   --------
 32  32  32
1  1
CON    1  32   1  32     12   4   1
2  1
SOL    1  32   1  32
3  1
CON    1  32   1  32      1   6   1
4  1
CON    1  32   1  32     10   1   1
5  1
CON    1  32   1  32     13   2   1
6  1
CON    1  32   1  32     12   6   1
--------   Block No.   10   --------
 32  32  32
1  1
CON    1  32   1  32      9   4   1
2  1
SOL    1  32   1  32
3  1
CON    1  32   1  32      2   6   1
4  1
CON    1  32   1  32     11   1   1
5  1
CON    1  32   1  32     14   2   1
6  1
CON    1  32   1  32     11   6   1
--------   Block No.   11   --------
 32  32  32
1  1
CON    1  32   1  32     10   4   1
2  1
SOL    1  32   1  32
3  1
CON    1  32   1  32      3   6   1
4  1
CON    1  32   1  32     12   1   1
5  1
CON    1  32   1  32     15   2   1
6  1
CON    1  32   1  32     10   6   1
--------   Block No.   12   --------
 32  32  32
1  1
CON    1  32   1  32     11   4   1
```

```
2  1
SOL    1  32   1  32
3  1
CON    1  32   1  32      4   6   1
4  1
CON    1  32   1  32      9   1   1
5  1
CON    1  32   1  32     16   2   1
6  1
CON    1  32   1  32      9   6   1
--------   Block No.   13   --------
 32  32  32
1  1
CON    1  32   1  32     16   4   1
2  1
CON    1  32   1  32      9   5   1
3  1
CON    1  32   1  32      5   6   1
4  1
CON    1  32   1  32     14   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     16   6   1
--------   Block No.   14   --------
 32  32  32
1  1
CON    1  32   1  32     13   4   1
2  1
CON    1  32   1  32     10   5   1
3  1
CON    1  32   1  32      6   6   1
4  1
CON    1  32   1  32     15   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     15   6   1
--------   Block No.   15   --------
 32  32  32
1  1
CON    1  32   1  32     14   4   1
2  1
CON    1  32   1  32     11   5   1
3  1
CON    1  32   1  32      7   6   1
4  1
CON    1  32   1  32     16   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     14   6   1
--------   Block No.   16   --------
 32  32  32
1  1
CON    1  32   1  32     15   4   1
2  1
CON    1  32   1  32     12   5   1
3  1
CON    1  32   1  32      8   6   1
4  1
CON    1  32   1  32     13   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     13   6   1
```

Splitted BC data - delta wing - desired block edge length 48

```
Delta wing - decomposed
 12
--------   Block No.    1   --------
 48  32  32
1  1
CON    1  32   1  32      3   4   1
2  1
SOL    1  48   1  32
3  1
MIR    1  48   1  32
4  1
CON    1  32   1  32      2   1   1
5  1
CON    1  48   1  32      4   2   1
6  1
CON    1  48   1  32      7   3   1
--------   Block No.    2   --------
 32  32  32
1  1
CON    1  32   1  32      1   4   1
2  1
SOL    1  32   1  32
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      3   1   1
5  1
CON    1  32   1  32      5   2   1
6  1
CON    1  32   1  32      8   3   1
--------   Block No.    3   --------
 48  32  32
1  1
CON    1  32   1  32      2   4   1
2  1
SOL    1  48   1  32
3  1
MIR    1  48   1  32
4  1
CON    1  32   1  32      1   1   1
5  1
CON    1  48   1  32      6   2   1
6  1
CON    1  48   1  32      9   3   1
--------   Block No.    4   --------
 48  32  32
1  1
CON    1  32   1  32      6   4   1
2  1
CON    1  48   1  32      1   5   1
3  1
MIR    1  48   1  32
4  1
CON    1  32   1  32      5   1   1
5  1
EXT    1  48   1  32
6  1
CON    1  48   1  32     10   3   1
--------   Block No.    5   --------
 32  32  32
1  1
CON    1  32   1  32      4   4   1
2  1
CON    1  32   1  32      2   5   1
3  1
MIR    1  32   1  32
4  1
CON    1  32   1  32      6   1   1
5  1
EXT    1  32   1  32
6  1
CON    1  32   1  32     11   3   1
--------   Block No.    6   --------
 48  32  32
1  1
CON    1  32   1  32      5   4   1
2  1
CON    1  48   1  32      3   5   1
3  1
```

```
MIR    1  48   1  32
4  1
CON    1  32   1  32      4   1   1
5  1
EXT    1  48   1  32
6  1
CON    1  48   1  32     12   3   1
--------   Block No.    7   --------
 48  32  32
1  1
CON    1  32   1  32      9   4   1
2  1
SOL    1  48   1  32
3  1
CON    1  48   1  32      1   6   1
4  1
CON    1  32   1  32      8   1   1
5  1
CON    1  48   1  32     10   2   1
6  1
CON    1  48   1  32      9   6   1
--------   Block No.    8   --------
 32  32  32
1  1
CON    1  32   1  32      7   4   1
2  1
SOL    1  32   1  32
3  1
CON    1  32   1  32      2   6   1
4  1
CON    1  32   1  32      9   1   1
5  1
CON    1  32   1  32     11   2   1
6  2
CON    1  16   1  32      8   6   2
CON   17  32   1  32      8   6   1
--------   Block No.    9   --------
 48  32  32
1  1
CON    1  32   1  32      8   4   1
2  1
SOL    1  48   1  32
3  1
CON    1  48   1  32      3   6   1
4  1
CON    1  32   1  32      7   1   1
5  1
CON    1  48   1  32     12   2   1
6  1
CON    1  48   1  32      7   6   1
--------   Block No.   10   --------
 48  32  32
1  1
CON    1  32   1  32     12   4   1
2  1
CON    1  48   1  32      7   5   1
3  1
CON    1  48   1  32      4   6   1
4  1
CON    1  32   1  32     11   1   1
5  1
EXT    1  48   1  32
6  1
CON    1  48   1  32     12   6   1
--------   Block No.   11   --------
 32  32  32
1  1
CON    1  32   1  32     10   4   1
2  1
CON    1  32   1  32      8   5   1
3  1
CON    1  32   1  32      5   6   1
4  1
CON    1  32   1  32     12   1   1
5  1
EXT    1  32   1  32
6  2
CON    1  16   1  32     11   6   2
CON   17  32   1  32     11   6   1
--------   Block No.   12   --------
 48  32  32
```

```
1  1
CON    1  32   1  32     11   4   1
2  1
CON    1  48   1  32      9   5   1
3  1
CON    1  48   1  32      6   6   1
4  1
CON    1  32   1  32     10   1   1
5  1
EXT    1  48   1  32
6  1
CON    1  48   1  32     10   6   1
```