WL-TM-91-312

..P.L.U.T.O.....

POISSON APLACE FI RTHOGONAL

3-D Grid Generator

User's Manual

Howard T. Emsley
WL/FIMM
Wright-Patterson AFB, Ohio
45433-6553

Aerodynamic Methods Group
Aerodynamics and Airframe Branch
Aeromechanics Division

June 1991

Flight Dynamics Directorate
Wright Laboratory
Wright-Patterson AFB, Oh 45433-6553

FOREWORD

This report was prepared under Work Unit 240410A1 entitled "Aerodynamic Design and Analysis Methods" by Howard T. Emsley of the Aerodynamic Methods Group, Aerodynamics and Airframe Branch, Aeromechanics Division, Flight Dynamics Directorate, Wright Laboratory, Wright-Patterson Air Force Base, Ohio.

The report documents PLUTO (Poisson Laplace U TFI Orthogonal), a three dimensional gridding and smoothing program, which was developed in-house between July 1987 and May 1991. The coding was primarily performed by Capt. David Amdahl and Mr. William Strang of the Aerodynamic Methods Group.

This document has been reviewed and approved.

Howard T Emsley

Howard T. Emsley
Aerospace Engineer

Dennis Sedlock

Dennis Sedlock
Chief, Aerodynamics and Airframe Branch

TABLE OF CONTENTS

```
 PPPPPPP   LL        UU    UU  TTTTTTTT   000000
 PP    PP  LL        UU    UU     TT     00    00
 PP    PP  LL        UU    UU     TT     00    00
 PPPPPPP   LL        UU    UU     TT     00    00
 PP        LL        UU    UU     TT     00    00
 PP        LL        UU    UU     TT     00    00
 PP        LLLLLLLL  UUUUUU       TT     000000
```

INTRODUCTION

PLUTO (Poisson Laplace U TFI Orthogonal), is a three-dimensional gridding and smoothing program developed in-house at WL/FIMM. Its purpose is to: (1) generate three-dimensional grids from the six two-dimensional faces of a computational grid, (2) check grid quality, (3) smooth grids to improve spacing, smoothness and orthogonality, and (4) output grids in user selected ascii or binary formats.

Initial grids are established by PLUTO with a three-dimensional algebraic transfinite interpolation (TFI) scheme. For many applications, where simple block shapes and spacings are used, an algebraic grid may be sufficiently smooth and orthogonal for use. As a check, QBERT[1] (Quantitative Block ERror Tester) is run on each block to generate diagnostics about the grid's spacing, skewness and right-handedness.

If the user is unsatisfied with the initial grid, Laplace and/or Poisson smoothing techniques are available, and orthogonality constraints can be enforced separately on each block face. By handling each grid block independently the user is afforded great flexibility and control over the finished grid. For example, complex block shapes whose initial algebraic grid are often crossed, can be smoothed with the Laplace technique to uncross the grid.

Once a finished grid block is generated, PLUTO also assists in outputting the blocks in a final format. This includes the capabilities to cut large

blocks into smaller blocks (this is used to subdivide large computational blocks to run on machines with limited memory), and/or output the grid in one of nine different formats.

BACKGROUND

Development of PLUTO has been somewhat of an evolutionary process. In 1987, the Aerodynamic Methods Group (WL/FIMMA) began a concerted effort to develop rapid geometric modeling and grid generation capabilities for the Air Force. Programs such as I3G/VIRGO[2], PLUTO and GRIDGEN[3] were the results of in-house and contracted efforts between 1987 and 1991. Each code has its specific applications where it excels. I3G[4] was originally developed under contract as surface modeller and then it was modified in-house (VIRGO) to support grid generation needs. PLUTO was then developed to generate three-dimensional grids as discussed above. Concurrently, GRIDGEN (GRIDBLOCK, GRIDGEN2D,and GRIDGEN3D) was developed under contract to generate grids around predefined surface databases. GRIDGEN3D performs the same function in the GRIDGEN grid generation process as PLUTO does for I3G/VIRGO.

PLUTO is not a revolutionary piece of coding, but its useability, flexibility, and inclusion of diagnostics (QBERT) are the things which make this code noteworthy and in high use. PLUTO incorporates the work of a variety of people in the grid generation field. The TFI method used is attributed to Dr. Soni, the Poisson control functions are from Dr. Joe Thompson's work[5], and the orthogonal control functions are a variation of those found in Sorenson's GRAPE Method[6].

PLUTO is written primarily in standard FORTRAN-77 however a few routines have been written in C. The code was originally written to operate on a Cray X-MP, however the version documented here was tailored for operation on an Iris workstation. Portability to other machines is expected to be straight forward, however, memory constraints and machine speed should be considered when porting to a less advanced host.

PROGRAM EXECUTION

When PLUTO is executed, the user is prompted to provide four different files: Input Data File, Output Data File, Input Grid File and the Output Grid File. The input files are existing files that the user will provide and the output files are new files the program will create.

The Input Data File is the equivalent of a job file on a Cray. It provides the program with all the flags necessary to direct the needed operations. Everything from block dimensions to convergence residuals are specified in this file. (See examples)

The Output Data File is the file where QBERT will write diagnostics about the blocks in the grid. The user should examine this file to determine the results of the run. At a minimum, this file will contain each block's dimensions and information on the worst cells of the block. Additional information is provided when required

The Input Grid File is the file on which PLUTO will operate. This file can be in ascii or binary format and will provide either the six starting faces of each block or a three-dimensional grid for each block. (See examples for further details)

The Output Grid File will contain the new three-dimensional grid. The format of this file is controlled with flags located in the Input Data File.

INPUT DATA FILE

PLUTO can be used to perform a wide variety of different functions for the user. Control of the program is handled almost exclusively in the Input Data File. A general explanation of this file follows, but often the examples included may provide the best guide.

Lines 1 and 2 of the file consist of a comment line explaining the inputs, a value for smoothing relaxation (values between 0. and 2.), the maximum number of iterations the smoother will run, and how often the screen will display the convergence information (see below). The solution can be over or under relaxed by varying the relaxation range.

```
**RELAXATION     NO. OF ITERATIONS      PRINT OUT/ITERATIONS     (line 1)
     .8               50                      5                (line 2)
```

Lines 3 and 4 consist of a comment line labeling the input, a flag for the convergence (average change of the block or maximum change of any one point), and the residual value for a converged solution.

```
**CONVERGENCE(0=AVER, 1=MAX. )      FINAL RESIDUAL              (line 3)
            1                          .000001                 (line 4)
```

Lines 5 thru 10 contain four comment lines and two lines containing form and format flags for the grid files. File form refers to ascii or binary format, and file format refers to the geometric format of the input and output files. PANAIR[7] and BLOCK formats provide only the six faces of each block while GRIDGEN, MERCURY[8], TEAM[9] and PLOT3D[10] expect a full three-dimensional grid. The user is directed to each code's manuals for format specifics.

```
**FORM      (1=BINARY,2=ASCII)                                    (line 5)
**FORMATS   (1=PANAIR,2=BLOCK,3=GRIDGEN,4=MERCURY,5=TEAM,6=PLOT3D)  (line 6)
**INPUT FILE FORM      INPUT FILE FORMAT (1,2, AND 3 ARE ASCII ONLY)(line 7)
      2                        1                                  (line 8)
**OUTPUT FILE FORM     OUTPUT FILE FORMAT (ONLY 4,5, AND 6)        (line 9)
      1                        6                                 (line 10)
```

Lines 11 thru 16 contain information for the orthogonality conditions. The columns correspond to P, Q, and R while the rows correspond to sides 1 through 6. The user need not modify these values, unless the

solution will not converge. If this occurs, the user is instructed to raise the values of all P, Q and R values and lower the relaxation value on line 2. The values shown below are used as exponential exponents, therefore values less than or equal to zero should not be used. (Enforcing orthogonality in three dimensions is a complicated task, and obtaining a converged solution may require a healthy dose of experimentation. In general, there is no standard "fix" to obtain convergence.)

.65	.25	.25	(line 11)
.65	.25	.25	(line 12)
.25	.65	.25	(line 13)
.25	.65	.25	(line 14)
.25	.25	.65	(line 15)
.25	.25	.65	(line 16)

Lines 17 through 20 contain three comment lines and the number of blocks the program should be reading from the Input Grid File. Starting on line 21, the Input Data File should contain information about the blocks in question. A variety of different inputs can appear after line 20 depending on the application desired. An unrealistic combination of options is shown below to illustrate the kinds of information that may be required for each block.

In general, PLUTO will require the block dimensions, the mode and a set of orthogonality flags for each block. The mode selected will determine the operation that will be performed on the block, and the orthogonality flags will determine which sides will have orthogonality imposed (0 = no orthogonality, 1 = orthogonality). These flags will only be taken into account under modes 2, 3 and 4. Additional information will be required for blocks the user wishes to cut into two or more smaller blocks.

When the grid is to be used on machines with limited memory, block

cutting is important to subdivide large computational grid blocks into smaller blocks. To cut a block, a ten should be added to the mode selected (e.g. mode 2 becomes mode 12). Following the mode line, the Input Data File should provide the number of new blocks and lines containing the looping indices of each new block (see block 3 below). If a block is being smoothed and cut, the block will not be cut until after the smoothing has been completed. By performing the functions in this order, slope continuity of the grid will occur between the newly cut blocks.

The first block (lines 21, 22 and 23) has dimensions of 94 103 35 and will be generated with a Transfinite Interpolation scheme (mode 1). This format is generally used to generate the initial grid from the six block faces. The orthogonality flags in this case are of no consequence.

The second block (lines 24, 25 and 26) with dimensions of 65 60 27 will be read in and written out with no manipulation of the grid. This assumes that the grid has already been generated as done in the previous block or possibly by another 3-D grid generator (such as GRIDGEN3D). This option can also be used if the user wants to rewrite existing blocks in another form or format without smoothing (see lines 5 thru 10 for further details). Again the orthogonality flags are of no consequence.

The third block (lines 27 to 29) with dimensions of 33 44 27 will be iterated on by a Poisson smoother for the number of iterations specified in line 2. Line 29 contains the six flags that can be set to impose orthogonality conditions at any or all of the block's six faces (0 = no orthogonality, 1 = orthogonality).

The fourth block (lines 30 to 35) with dimensions of 33 4 27 will be iterated on by a Laplacian smoother with orthogonality imposed on faces 3, 5 and 6. After the defined iterations (line 2) or convergence, the

block will be cut into two smaller blocks at i=10. In this case, line 30 contains the current dimensions of the block, line 31 indicates the smoother mode (2) and that a cut will take place (+10), line 32 indicates number of new blocks, lines 33 and 34 provide the new dimensions, and line 35 provides the orthogonality flags (used before cut).

```
**NUMBER OF BLOCKS                                          (line 17)
**DIMENSION                                                 (line 18)
**MODE (0=READ/WRITE,1=TFI,2=LAPL,3=POIS,4=POIS w/CURV)     (line 19)
   4                                                        (line 20)
    94 103  35                                              (line 21)
     1                                                      (line 22)
     0 0 0 0 0 0                                            (line 23)
    65  60  27                                             (line 24)
     0                                                      (line 25)
     0 0 0 0 0 0                                            (line 26)
    33  44  27                                             (line 27)
     3                                                      (line 28)
     1 1 0 0 0 0                                            (line 29)
    33   4  27                                             (line 30)
     12                                                     (line 31)
     2                                                      (line 32)
      1  10   1   4   1  27                                 (line 33)
     10  33   1   4   1  27                                 (line 34)
     0 0 1 0 1 1                                            (line 35)
```

MODES

PLUTO provides the user with 5 modes of operation:

0=READ/WRITE
1=TFI
2=LAPLACE
3=POISSON
4=POISSON with CURVATURE

The simplest of these is READ/WRITE which is appropriately named. This mode will cause the program to read a block in and write it back out without modifying the original block. It can be used quite effectively to transform one file format or form into another.

TFI (Transfinite Interpolation) is an algebraic grid generator, which is often used to generate the original grid from the six block faces. For many applications, algebraic grids will be sufficiently smooth and no smoothing will be required.

The Laplace mode tends to equalize the volume of the cells in a block. It can also be used to uncross algebraic grids that may be crossed due the block shape. Crossed grids must be uncrossed before the other smoothing modes are used.

The Poisson and Poisson with curvature modes are also used to elliptically smooth the grid. In these modes, Thomas-Middlecoff control functions (based on the boundary arclengths) are linearly interpolated across the grid to control the spacing. In the latter mode, the control functions are modified by the curvatures of the adjacent boundaries.

Only the three smoothers (modes 2,3 and 4) allow the user to control orthogonality at the block faces.

Experimentation and practice will provide the user a much clearer understanding of each modes applications. Two dimensional forms of these smoothers are used in I3G/VIRGO, and the user is directed there to experiment and view the different results that can occur.

EXAMPLES

The following three example have been included to illustrate a variety of different Input Data Files.

EXAMPLE 1

In this example, we will generate four blocks of a grid from the six faces of each block in an ascii PANAIR formatted file. The three-dimensional grid will be output in a binary PLOT3D format and the grid will be generated algebraically with TFI. The Input Data File is as follows.

```
**RELAXATION    NO. OF ITERATIONS     PRINT OUT/ITERATIONS
     .8              50                    5
**CONVERGENCE(0=AVER, 1=MAX. )     FINAL RESIDUAL
             1                        .000001
**FORM     (1=BINARY,2=ASCII)
**FORMATS  (1=PANAIR,2=BLOCK,3=GRIDGEN,4=MERCURY,5=TEAM,6=PLOT3D)
**INPUT FILE FORM      INPUT FILE FORMAT (1,2, AND 3 ARE ASCII ONLY)
       2                      1
**OUTPUT FILE FORM     OUTPUT FILE FORMAT (ONLY 4,5, AND 6)
       1                      6
              .65      .25      .25
              .65      .25      .25
              .25      .65      .25
              .25      .65      .25
              .25      .25      .65
              .25      .25      .65
**NUMBER OF BLOCKS
**DIMENSION
**MODE (0=READ/WRITE,1=TFI,2=LAPL,3=POIS,4=POIS w/CURV)
   4
```

```
94 103  35
 1
 0 0 0 0 0 0
65  60  27
 1
 0 0 0 0 0 0
33  44  27
 1
 0 0 0 0 0 0
33   4  27
 1
 0 0 0 0 0 0
```

EXAMPLE 2

Now that a three-dimensional grid exists we will smooth blocks 2, 3 and 4 for a maximum of 150 iterations while leaving block 1 as is. The Input Grid File is now a binary PLOT3D file, and the Grid Output File is an ascii MERCURY file. No orthogonality is specified.

```
**RELAXATION     NO. OF ITERATIONS      PRINT OUT/ITERATIONS
     .8              150                     5
**CONVERGENCE(0=AVER, 1=MAX. )      FINAL RESIDUAL
            1                          .000001
**FORM      (1=BINARY,2=ASCII)
**FORMATS   (1=PANAIR,2=BLOCK,3=GRIDGEN,4=MERCURY,5=TEAM,6=PLOT3D)
**INPUT FILE FORM       INPUT FILE FORMAT (1,2, AND 3 ARE ASCII ONLY)
      1                        6
**OUTPUT FILE FORM      OUTPUT FILE FORMAT (ONLY 4,5, AND 6)
      2                        4
             .65       .25       .25
```

```
              .65       .25       .25
              .25       .65       .25
              .25       .65       .25
              .25       .25       .65
              .25       .25       .65
**NUMBER OF BLOCKS
**DIMENSION
**MODE (O=READ/WRITE,1=TFI,2=LAPL,3=POIS,4=POIS w/CURV)
  4
   94 103   35
    0
    0 0 0 0 0 0
   65  60   27
    2
    0 0 0 0 0 0
   33  44   27
    3
    0 0 0 0 0 0
   33   4   27
    2
    0 0 0 0 0 0
```

EXAMPLE 3

We will now assume that the grid is fully smoothed. Block 1 however is too big for our solver therefore it must be cut into three pieces. The Input Grid File is now an ascii MERCURY file, and the Grid Output File will be an ascii MERCURY file.

```
**RELAXATION     NO. OF ITERATIONS      PRINT OUT/ITERATIONS
     .8               50                      5
```

```
**CONVERGENCE(0=AVER, 1=MAX. )      FINAL RESIDUAL
             1                          .000001
**FORM      (1=BINARY,2=ASCII)
**FORMATS   (1=PANAIR,2=BLOCK,3=GRIDGEN,4=MERCURY,5=TEAM,6=PLOT3D)
**INPUT FILE FORM       INPUT FILE FORMAT (1,2, AND 3 ARE ASCII ONLY)
       2                        4
**OUTPUT FILE FORM      OUTPUT FILE FORMAT (ONLY 4,5, AND 6)
       2                        4
               .65       .25       .25
               .65       .25       .25
               .25       .65       .25
               .25       .65       .25
               .25       .25       .65
               .25       .25       .65
**NUMBER OF BLOCKS
**DIMENSION
**MODE (0=READ/WRITE,1=TFI,2=LAPL,3=POIS,4=POIS w/CURV)
   4
    94 103  35
    10
    3
     1  94   1  30   1  35
     1  94  30  60   1  35
     1  94  60 103   1  35
    0 0 0 0 0 0
    65  60  27
     0
     0 0 0 0 0 0
    33  44  27
     0
     0 0 0 0 0 0
    33   4  27
     0
     0 0 0 0 0 0
```

REFERENCES

1. Strang, W. Z., "QBERT: A Grid Evaluation Code", AFWAL-TM-88-193.

2. Emsley, H. T., "I3G/VIRGO, Interactive Graphics for Geometry Generation and Visual Interactive Rapid Grid Generation, User's Manual", WRDC-TM-90-317.

3. WRDC-TR-90-3022, "The GRIDGEN 3D Multiple Block Grid Generation System".

4. LaBozzetta, W. F., "Configuration Data Management System", AFWAL-TR-87-3064 (Vols. I and II).

5. Thompson, Joe F., Numerical Grid Generation, North-Holland, 1985.

6. Sorenson, Reese L., "Three-Dimensional Zonal Grids About Arbitrary Shapes by Poisson's Equation", Numerical Grid Generation In Computational Fluid Mechanics '88, 1988, pp. 75-84.

7. Magnus, Alfred E., "PAN AIR -A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows About Arbitrary Configurations Using a Higher Order Panel Method", NASA Contractor Report 3251.

8. Strang, W. Z., "MERCURY User's Manual", AFWAL-TM-88-217.

9. Raj, P., "Three-dimensional Euler/Navier-Stokes Aerodynamic Method (TEAM)", AFWAL-TR-87-3074.

10. Buning, Pieter G., "PLOT3D User's Manual", NASA TM 101067, 1989.