NASA Technical Memorandum **108985**

THE THREE-DIMENSIONAL MULTI-BLOCK ADVANCED GRID GENERATION SYSTEM (3DMAGGS)

STEPHEN J. ALTER

KENNETH J. WEILMUENSTER

MAY 1993

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Contents

Abstract

As the size and complexity of three dimensional volume grids increases, there is a growing need for fast and efficient 3D volumetric elliptic grid solvers. Present day solvers are not necessarily memory bound, but limited by computational speed. In addition, current solvers do not have all the capabilities such as interior volume grid clustering control, viscous grid clustering at the wall of a configuration, truncation error limiters and convergence optimization residing in one code. A new volume grid generator, 3DMAGGS (Three Dimensional Multi-block Advanced Grid Generation System), which is based on the 3DGRAPE[1] code written by Reese L. Sorenson at NASA Ames Research Center, has evolved to meet these needs. The system encompasses different options for a variety of volume grid generation needs.

The 3DMAGGS code proves to be the fastest volumetric elliptic grid generator available to the public domain sector and offers many of the capabilities of GRIDGEN3D.[2] 3DMAGGS utilizes the vectorized code 3DGRAPE for computational speed while adding state-of-the-art volume grid controls. Overall, 3DMAGGS is at least 18 times faster than GRIDGEN3D's anti-biasing routines, with state-of-the-art grid control capabilities activated and still as fast as its parent version of 3DGRAPE.

This is a manual for the usage of 3DMAGGS and consists of five sections. They include the motivations and usage, a GRIDGEN interface, a grid quality analysis tool, a sample case for verifying correct operation of the code and a comparison section to both 3DGRAPE and GRIDGEN3D. Since it was derived from 3DGRAPE, this paper should be used in conjunction with the 3DGRAPE manual. With this in mind, it is strongly recommended that the 3DGRAPE manual be consulted on 3DGRAPE type options and usage.

Acknowledgments

The authors would like to acknowledge Dr. Jamshid S. Abolhassani. Discussions with Dr. Abolhassani have been most helpful in understanding the intricacies of volume grid generation and implement state-of-the-art techniques into 3DMAGGS.

List of Symbols

Symbol	Description
abc	Exponential decay rate for a block/face in 3DMAGGS and 3DGRAPE.
A	Area of a face of one volumetric cell.
AR	Aspect ratio of a volumetric cell.
EXPO	Exponential decay rate for a grid block system in GRIDGEN3D.
$ORTHO_{\xi\eta}$	Orthogonality function of intersecting grid lines in the ζ=constant parametric plane.
$ORTHO_{\eta\zeta}$	Orthogonality function of intersecting grid lines in the ξ=constant parametric plane.
$ORTHO_{\xi\zeta}$	Orthogonality function of intersecting grid lines in the η=constant parametric plane.
P	Forcing function in physical space that is in the direction of the first parametric index.
Q	Forcing function in physical space that is in the direction of the second parametric index.
R	Forcing function in physical space that is in the direction of the third parametric index.
α	Soni blending coefficient in ξ direction.
β	Soni blending coefficient in η direction.
γ	Face number used in file name construction.
δ	Block number used in file name construction.
μ	Percentage of a function's contribution of a neighboring surface grid point to a surface interior grid point.
ξ	First parametric index of the computational domain.
η	Second parametric index of the computational domain.
ζ	Third parametric index of the computational domain.
ω	Relaxation parameter.
σ_i	Percentage of a function's contribution of an edge point to a surface interior grid point.
$\theta_{\xi\eta}$	Angle of intersecting grid lines in the ζ=constant parametric plane.
$\theta_{\eta\zeta}$	Angle of intersecting grid lines in the ξ=constant parametric plane.
$\theta_{\xi\zeta}$	Angle of intersecting grid lines in the η=constant parametric plane.
Δs	Distance from a surface grid point to the next volume interior grid point.
Φ	Parametric space forcing function in the direction of ξ.
Ψ	Parametric space forcing function in the direction of η.
Ω	Parametric space forcing function in the direction of ζ.
$\vec{r}_\xi$	Vector matrix of $[X \;\; Y \;\; Z]^T$ representing the first derivative of position in the ξ direction.
$\vec{r}_\eta$	Vector matrix of $[X \;\; Y \;\; Z]^T$ representing the first derivative of position in the η direction.
$\vec{r}_\zeta$	Vector matrix of $[X \;\; Y \;\; Z]^T$ representing the first derivative of position in the ζ direction.

Introduction

Two publicly available grid generation packages have been created for the construction of large structured volume grids, over the past 20 years. The EAGLE[3] code has proven to be successful and versatile when constructing surface and volume grids for missiles and similar configurations. The EAGLE code was originally designed for the CRAY-XMP and relied on extensive external file "I/O" usage. When ported to state-of-the-art supercomputers such as the CRAY-YMP series, this code's structure is responsible for excessively large CPU time requirements when generating grids requiring 1-4 million grid points.

The GRIDGEN3D[2]code works well with large and small numbers of grid point volumes but is only first order accurate in boundary condition formulations. GRIDGEN3D was designed to run on a CRAY-XMP with limited memory, employing very little vectorization. Although it is faster than EAGLE, GRIDGEN3D is still slow and cumbersome to operate. GRIDGEN3D does not take complete advantage of supercomputer technology. Furthermore, both EAGLE and GRIDGEN3D lack the application of cell sizes at a boundary based on apriori CFD calculations and both codes use mathematical relationships to calculate these cell sizes.

To make use of current supercomputing technology and current grid generation techniques, a task was undertaken to create a fast and efficient solver. Instead of generating a complete new code, an existing solver, the 3DGRAPE code created by Reese L. Sorenson at NASA Ames Research Center, was chosen as the starting point. This code was modified to include state-of-the-art volumetric grid controls. The 3DMAGGS code is an extension of 3DGRAPE, but has been given a new name to differentiate the nuances of the improvements.

The 3DGRAPE code was chosen specifically due to ease of control deck generation as well as its efficiency. This version of 3DGRAPE runs at approximately 6.5 micro-seconds per iteration per grid point.

The 3DGRAPE code embodies the modularity required for future development, which facilitated the development of 3DMAGGS. Little difficulty was encountered in the linking of necessary routines. The 3DMAGGS code extends the capabilities of the 3DGRAPE code to include:

[1] Initialization via three dimensional trans-finite interpolation (3DTFI).[4]

[2] Thomas & Middlecoff Volume grid clustering controls[5](TMCs).

[3] Hybrid controls of Sorenson and Thomas & Middlecoff.[6]

[4] Optimized grid point movement.

[5] Incorporation of an outside source definition for the cell-size specification Δs, for calculating Poisson forcing function controls. This is done through a file, residing outside of 3DMAGGS.

[6] Truncation-error limiting due to low precision machines.

The 3DMAGGS code does not employ a Graphical User Interface, since it is intended to be a batch volume grid generator. With little effort 3DMAGGS could be incorporated into the GRAPEVINE[7] code, the interactive version of 3DGRAPE.

Finally, 3DGRAPE was chosen because the first derivatives are second order accurate. With this added accuracy, surface contours become more pronounced in the grid volume

near the configuration's surface. The resulting grids tend to better model the flow near the wall.

To accompany the 3DMAGGS code, two other support codes have been created. The first code is the conversion program from GRIDGEN2D to 3DMAGGS, named PREMAGGS. PREMAGGS utilizes the boundary conditions file ([file].bnda) and the surface grid distributions file ([file].mlga) required by GRIDGEN3D to generate the input decks for 3DMAGGS. The code uses a limited number of inputs from the user to generate all command controls for the 3DMAGGS code including the following:

[1] Automatic conversion of the forcing function decay rates from a block face with orthogonality controls activated.

[2] Generation of all necessary batch process files and input files for executing 3DMAGGS.

[3] Re-assigning the limits to the parameter statement in 3DMAGGS for efficient use of available memory.

[4] The calculation of the cell size specification for the elliptic solver via the Local ARc-length Cell Sizing (LARCS[8]) as well as the two dimensional trans-finite interpolation (2DTFI)[4] methods for calculating the Poisson forcing function controls.

The availability of the PREMAGGS code enables the user to opt for a more efficient three dimensional volume grid generator. PREMAGGS enables the use of GRIDGEN2D as a surface modeler/grid generator to define the distributions on non-matching faces of a multi-block grid system.

The second support code is 3DVOLCHK. This program has been created to evaluate the

quality of the volume grid generated by the 3DMAGGS program. This code will locate any negative volumes, as well as output five grid quality measures. These measures include cell volume, aspect ratio, and three skewness factors for each point. The skewness factor is a number between zero and 1, a skewness factor of 1.0 (100%) represents total orthogonality and the factor can decrease to 0.0 (i.e., 0% orthogonality).

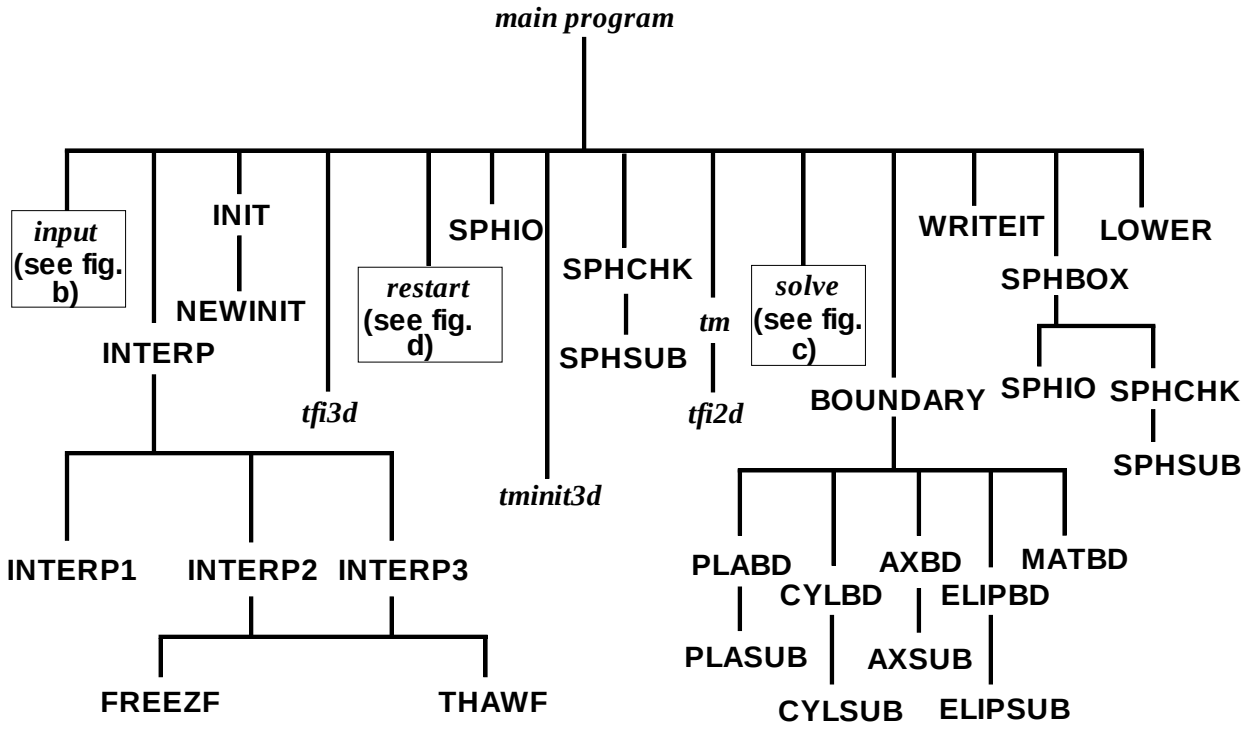
This paper consists of five sections. The first section describes the capabilities added to 3DGRAPE, the input formats and the effects of each modification. The second section is a description of the PREMAGGS code, including input and output. The usage of the 3DVOLCHK code and its inputs and outputs make up the third section, which also includes an explanation of the output variables. Section four is comprised of a simple case to verify the correct operation of the code, and the fifth section contains comparisons to GRIDGEN3D as well as 3DGRAPE.

The 3DMAGGS Code Structure

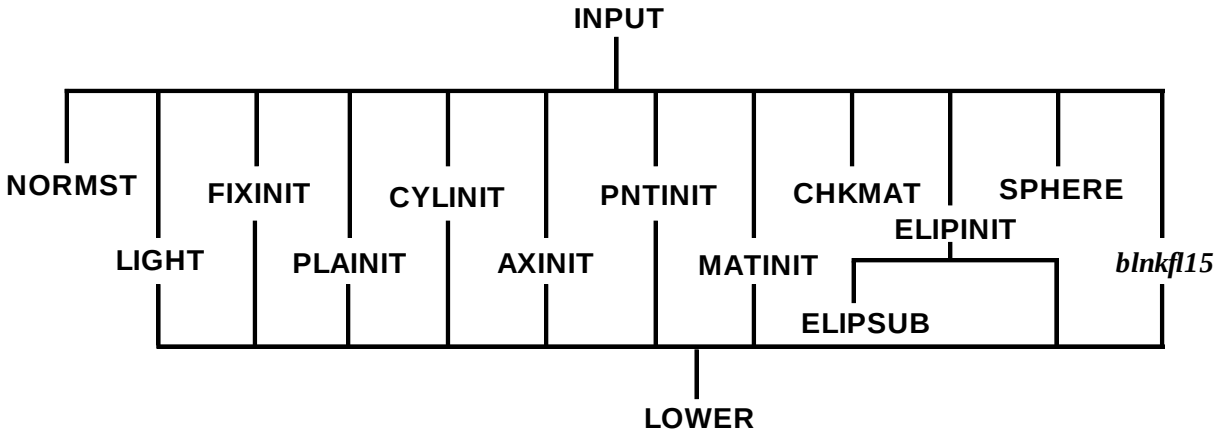
The source code of 3DMAGGS contains 60 subroutines, 8 of which enable the added capabilities. Figure 1 is a series of charts describing the routines used inside the 3DMAGGS code, where the lower-case italicized ones required modification or are added routines. The following 3DGRAPE routines were modified as follows:

[1] FIXINIT: has a new counter which returns the number of points read.

The number is used to determine if the entire face was "read-in-fixed" for use with the 3DTFI initialization option.

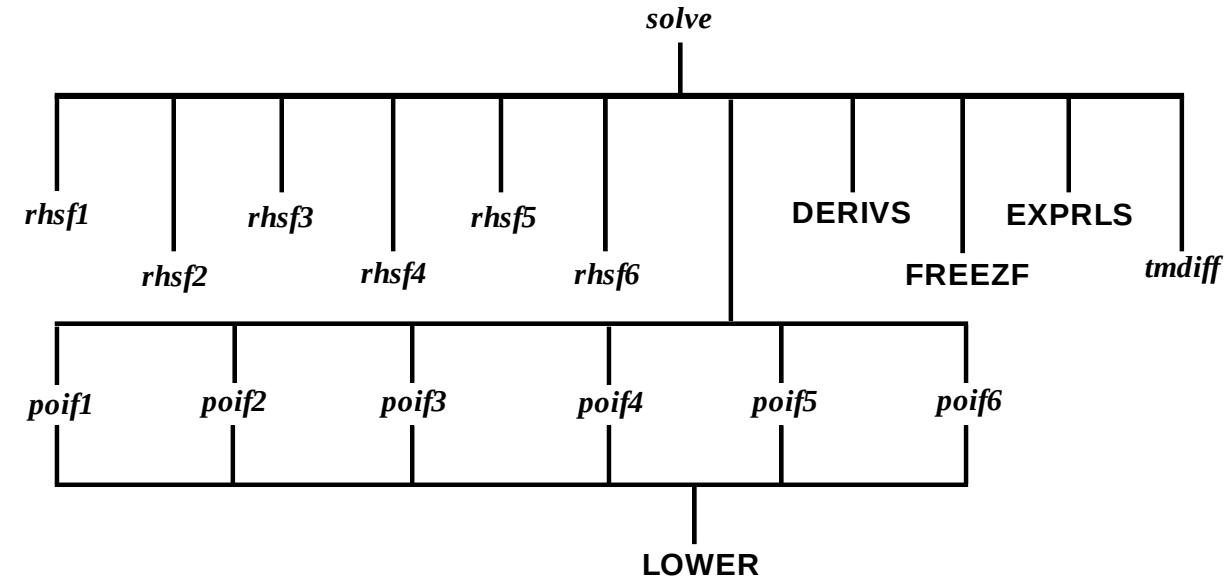


(a) Main program and subroutine structure.

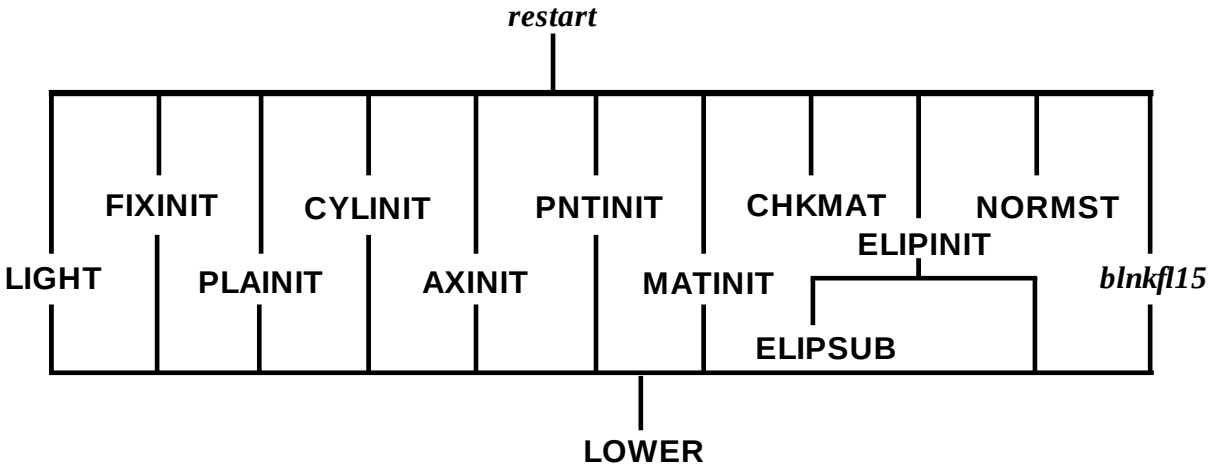


(b) Input subroutine structure and associated routines.

Figure 1: 3DMAGGS code structure and routines employed for 3D volume grid generation.



(c) Solve subroutine structure and associated routines.



(d) Restart subroutine structure and associated routines.

Figure 1: 3DMAGGS code structure and routines employed for 3D volume grid generation.(concluded)

[2] INPUT: now uses a blank filter for file names to prevent the opening of files with trailing or leading blanks.

[3] MAIN: embodies the new TMCs routines, called once at the beginning of each iteration part, where appropriate.

[4] POIF[1-6]: routines now have the added option of assigning the Δs based on read in values from an external file. The read will occur once in the beginning of the iteration cycles.

[5] RHSF[1-6]: routines now have the block numbers and iteration part number sent to them to be used to determine if the new values for P, Q and R need to be differenced with the TMCs for the hybrid controls.

[6] SOLVE: has the grid-point movement optimization added to the vectorized loop, with the option not to change the current block and the resetting of the face P, Q and R via the differencing of TMCs for the hybrid controls. This routine now returns the limits of the optimized relaxation parameter to the main program.

In addition, the following routines were added to 3DGRAPE to extend its capabilities:

[1] BLNKFL15: returns a given file name, consisting of 15 characters, with all blanks removed.

[2] LARCS: is an alternate code used for distributing a function on the edges of a surface onto the interior of that surface using the LARCS[8] method.

[3] TCPLT: writes out the TMCs calculated by the TM routine for each calculation or re-calculation of the TMCs for use with TECPLOT^{TM}[9].

[4] TFI2D: distributes a function on the edges of a surface based on two- dimensional trans-finite interpolation (2DTFI), using Soni's blending coefficients.

[5] TFI3D: initializes the volume grid of a block based on all 6 boundary faces providing that all faces were "read-in-fixed."

[6] TM: calculates the TMCs in parametric space, converts them to the physical domain and uses TFI2D to distribute them throughout the volume.

[7] TMDIFF: differences the TMCs on a per face basis to formulate the hybrid controls with Sorenson's orthogonality. This is done before and after the block is elliptically solved for one iteration.

[8] TMINIT3D: initializes the forcing functions on the faces to those calculated for the TMCs, providing a better initial guess for the orthogonality forcing functions.

The 3DMAGGS code has the same characteristics as 3DGRAPE for execution on a workstation as well as a super computer such as the CRAY. Refer to the 3DGRAPE manual for the conversion between machines. However, the 3DMAGGS code does have 6 additional common blocks and 4 more parameters in the parameter statement. The additional parameters can be found in Table 1.

Table 1. Parametric Limits.

Parameter:	Supplied Value:	Meaning:
imx	200	Maximum number of grid points in the ξ direction.
jmx	100	Maximum number of grid points in the η direction.
kmx	100	Maximum number of grid points in the ζ direction.
ijkmx	200	Maximum vector length of any of the three directions (ξ,η,ζ).

The extensions to 3DGRAPE found in 3DMAGGS have increased CPU memory requirements by 170% while decreasing the run time by a factor of 5. As stated, the 3DMAGGS code has the same input and output files as 3DGRAPE. Therefore, refer to the 3DGRAPE manual for the structure of these files.

3DMAGGS INPUT and OUTPUT

Input to 3DMAGGS is very similar to the 3DGRAPE code. Only the modifications made to the input decks for initial start and restart of the elliptic solver are outlined. In order to be consistent with the capabilities added to the 3DGRAPE to make 3DMAGGS, this section of the document is broken down to six components. Each part discusses the operation of one new capability, along with the effects of exercising the option.

Volume Grid Initialization

The code has the capability of generating the initial volume grid via the original 3DGRAPE method using linear, equal spacing distributions between each opposing face or through three dimensional trans-finite Interpolation (3DTFI). Specification of the method to be used is on the line following the relaxation parameter. Its construct can be found in Table 2.

Table 2. Volume Grid Initialization Instruction Format.

Line no.:	Field no.:	Column nos.:	Datum type:	Description:
	1	1-15	k	"initialization="
	2	16-27	c	either "keep-default" or "trans-finite"
	3	28-39	k	"-iterations="
	4	40-42	i	Number of iterations needed for optimization of initial grid

```
initialization=keep-default-iterations=  0

initialization=trans-finite-iterations= 20
```

Placing the string, "trans-finite" in columns 16-27, instructs the 3DMAGGS code to

construct the volume grid via 3DTFI. The 3DTFI code is based on volume optimization[4] of arc-length distributions from each defined boundary. This technique requires a number of iterations to be utilized, and this number is specified in columns 40-42. Typically, the technique requires 16 to 20 iterations to obtain optimized grid distributions but, it can be user specified.

The initialization, via the 3DTFI option, requires that all faces are read in as fixed. All matching face boundaries can still be read in as fixed, but, as the solver progresses, the original matching face will change. A better original distribution of the grid is used as the starting point of the elliptic solution. Current studies have shown that by using this method those faces with orthogonality specified have grid line distributions closer to orthogonality. The number of iterations used by the solver to obtain the correct forcing functions for enforcing orthogonality can be reduced by 75%. This reduction in itself tends to make the convergence of the solution quicker.

Volume Grid Clustering Control

Volume grid clustering controls are made possible by using the Thomas and Middlecoff forcing functions. There are two places in the control deck where volume interior control functions can be specified. First, there is global control available in the line(s) specifying the number of iterations and orthogonality control for those number of iterations. The new construct of this line is illustrated in Table 3.

Table 3. Thomas & Middlecoff Global Clustering Controls Instruction Format.

Line no.:	Field no.:	Column nos.:	Datum type:	Description:
	1	1-11	k	"iterations="
	2	12-14	i	number of iterations in this part
	3	15-23	k	"-control="
	4	24-25	c	global switch on control, either "ye", "no", or "fz"
	5	26-38	k	"-coarse/fine"
	6	39-42	c	"coar" or "fine"
	7	43-61	k	"-thomas-middlecoff="
	8	62-68	c	"none", "initial", "only", or "hybrid"

```
iterations=100-control=no-coarse/fine=coar-thomas-middlecoff=none

iterations=100-control=ye-coarse/fine=fine-thomas-middlecoff=initial

iterations=100-control=ye-coarse/fine=fine-thomas-middlecoff=only

iterations=100-control=ye-coarse/fine=fine-thomas-middlecoff=hybrid
```

The four options available for this type of control allow the user to globally set the type of resulting forcing functions used by the elliptic solver. The effects of each are as follows:

[1] "none" - Thomas and Middlecoff controls are disabled for entire set of iterations.

[2] "initial" - Thomas and Middlecoff controls are calculated only to initialize the elliptic solver forcing functions. Thus, the solver does not start with the forcing functions set to zero. Rather, they start with a "best guess".

[3] "only" - Thomas and Middlecoff forcing functions are used to generate the volume grid with no guarantee of orthogonality on any boundary.

[4] "hybrid" - Thomas and Middlecoff controls in conjunction with the Sorenson orthogonality control at a boundary are used to generate the volume grid utilizing a background forcing function technique.[6] In this case, the grid exhibits the volume grid clustering controls as well as having orthogonality at a specified boundary or boundaries.

Interior volume-grid-clustering controls are calculated only once, at the beginning of each iteration sequence, where the Thomas and Middlecoff controls are activated. The Thomas and Middlecoff controls are smoothed utilizing a weighted averaging algorithm, equation 1. This smoothing is continued for 5 cycles with ω equal to .2, in equation 2, to increase the window of influence. The larger window of influence simply adds dependence of a point's control function value on adjacent points. The larger window of influence for smoothing reduces the effects of discontinuities on the six defining surfaces, preventing those effects from affecting the interior grid clustering.

$$P'(\eta,\zeta) = \frac{1}{7}(\mu_1 P(\eta+1,\zeta) + \mu_2 P(\eta-1,\zeta) + \mu_3 P(\eta,\zeta+1) + \mu_4 P(\eta,\zeta-1) + 4P(\eta,\zeta)) \quad (1)$$

$$P(\eta,\zeta) = P'(\eta,\zeta)(1-\omega) + P(\eta,\zeta) \quad (2)$$

where,

$$\mu_j = 1.0 - \frac{\Delta s_j}{\sum_{i=1}^{4} \Delta s_i} \tag{3a}$$

and for j=1 (corresponds to η),

$$\Delta s_1 = \sqrt{(x(\eta+1,\zeta) - x(\eta,\zeta))^2 + (y(\eta+1,\zeta) - y(\eta,\zeta))^2 + (z(\eta+1,\zeta) - z(\eta,\zeta))^2} \tag{3b}$$

The forcing functions are interpolated onto the interior volume using a modified two-dimensional trans-finite interpolation (2DTFI) method. The interpolation method uses 50% of the sum from the first four terms and neglects the corner point corrections (equation 4), maintaining edge and corner values on the parametric surface.

$$P(\xi,\eta) = \frac{1}{2}(\sigma_1 P(\xi_{min},\eta) + \sigma_2 P(\xi_{max},\eta) + \sigma_3 P(\xi,\eta_{min}) + \sigma_4 P(\xi,\eta_{max})) \tag{4a}$$

where,

$$\sigma_1 = 1.0 - \alpha \tag{4b}$$

$$\sigma_2 = \alpha \tag{4c}$$

$$\sigma_3 = 1.0 - \beta \tag{4d}$$

$$\sigma_4 = \beta \tag{4e}$$

$$\tag{4f}$$

and,

$$\alpha = \frac{t1(\eta) + s1(\xi)(t2(\eta) - t1(\eta))}{1.0 - (s2(\xi) - s1(\xi))(t2(\eta) - t1(\eta))} \tag{4g}$$

$$\beta = \frac{s1(\eta) + t1(\xi)(s2(\eta) - s1(\eta))}{1.0 - (s2(\xi) - s1(\xi))(t2(\eta) - t1(\eta))} \tag{4h}$$

The "s" and "t" are the normalized arc lengths along the ξ and η directions. The "1" and "2" designate the minimum or maximum values of the respective parametric coordinates. The change in the 2DTFI equation was implemented because if the forcing functions at the corner points were different than the trends in the middle of the edges, resulting interpolated forcing functions are amplified on the interior. The differing trends are typically due to cell spacing gradients and surface curvature at the corners. By using the modified 2DTFI equations with Soni's[4] blending coefficients, better behaved forcing functions will result.

In addition to the global control command line, the TMCs can be activated or deactivated by block for an entire run of multiple iteration sets. For blocks where the TMC's are not necessarily needed, the use of the control functions can be eliminated without affecting other surrounding blocks. This command is found in the last line of the "block" information section and before any face information is defined (Table 4).

Table 4. Thomas & Middlecoff Local Clustering Controls Instruction Format.

Line no.:	Field no.:	Column nos.:	Datum type:	Description:
	1	1-18	k	"thomas-middlecoff="
	2	19-20	c	local Thomas & middlecoff clustering controls, either "ye" or "no"
	3	21-35	k	"iterate-block="
	4	36-37	c	local control on solving individual block, either "ye" or "no"

```
thomas-middlecoff=ye-iterate-block=ye

thomas-middlecoff=no-iterate-block=ye

thomas-middlecoff=ye-iterate-block=no

thomas-middlecoff=no-iterate-block=no
```

In columns 19-20, the string "ye" or "no" tells the elliptic solver whether or not to utilize the Thomas and Middlecoff controls for this block. By deactivating the TMC's with a "no", the solver is prevented from using any form of the TMC's. This does not affect the operation on the rest of the blocks. In addition to the local Thomas and Middlecoff controls, the option to let the elliptic solver run this block is also available within the control line. In columns 36-37, the "ye" instructs the program to solve the elliptic equations with the defined controls. A "no" in this field would tell the solver to completely ignore this block. Because this line is located once in each block, the local controls hold for all parts of the iteration schedule.

Freezing Forcing Function Corrections

Users of grid generation and Computational Fluid Dynamics (CFD) codes are switching from the super computers like the CRAY to modern and efficient workstations. While these fast workstations provide speed, the users are finding the reduced precision from 64 bit to 32 bit precision induces problems due to truncation error. With grid generation, the forcing functions necessary to enforce orthogonality end up being determined before the grid-point movement has a chance to converge. To counteract this problem, the option of freezing the updates to the control functions has been added to 3DMAGGS. The option is located in the "iterations" line of the control deck as illustrated in Table 4.

Placing the "fz" in columns 24-25 instructs the solver to set the relaxation parameter to the update of the forcing functions in the solver to zero. This means the forcing functions are still calculated, but their values will not change. The routines are vectorized enough to be executed quickly with insignificant impact on performance. The grid-point movement is the only quantity allowed to change. The use of this option gives the user an improved control on grid-point convergence, leading towards a better solution when using 32-bit hardware.

Grid Point Movement Optimization

As stated in the 3DGRAPE document, the code uses a PSOR method. The document states that this method does vectorize well, but the relaxation parameter is user defined and fixed for the entire grid generation process. In order to enhance convergence, the method in 3DMAGGS employs the same PSOR, but the relaxation factor can be determined by Ehrilch's[10] method. The implementation is simple because there was no change to the relaxation parameter command line as illustrated in Table 5.

Table 5. Grid Point Movement Optimization Instruction Format.

Line no.:	Field no.:	Column nos.:	Datum type:	Description:
	1	1-17	k	"relaxation-parameter="
	2	18-29	c/f	"keep-default", $+\omega$ or $-\omega$ relaxation factor
	3	30-54	k	"-forcing-function-output="
	4	55-56	c	"ye" or "no"

```
relaxation-param=keep-default

relaxation-param=0.3500000000

relaxation-param=-.7500000000
```

To activate the optimization, input a negative number for the relaxation parameter. This number is the percentage of the optimum the user wishes. The recommended percentage is 75% (-.75) because full optimization tends to make the solver unstable. The method of 3DGRAPE is retained in the specification of the relaxation parameter by using a positive number or "keep-default".

The vectorization algorithm employed in 3DMAGGS only uses those points that are surrounding the point in the vector, not the points directly before and after (Figure 2). The open points are not used because of the increased magnitudes of eigenvalues in the vectorized direction. The previous point is already at the next solution in time. So, if the relaxation parameter is calculated at the next consecutive point based on the previous point, the optimum relaxation parameter would be consistently one solution ahead of the previous point. The points along the vector are neglected when calculating the optimum relaxation

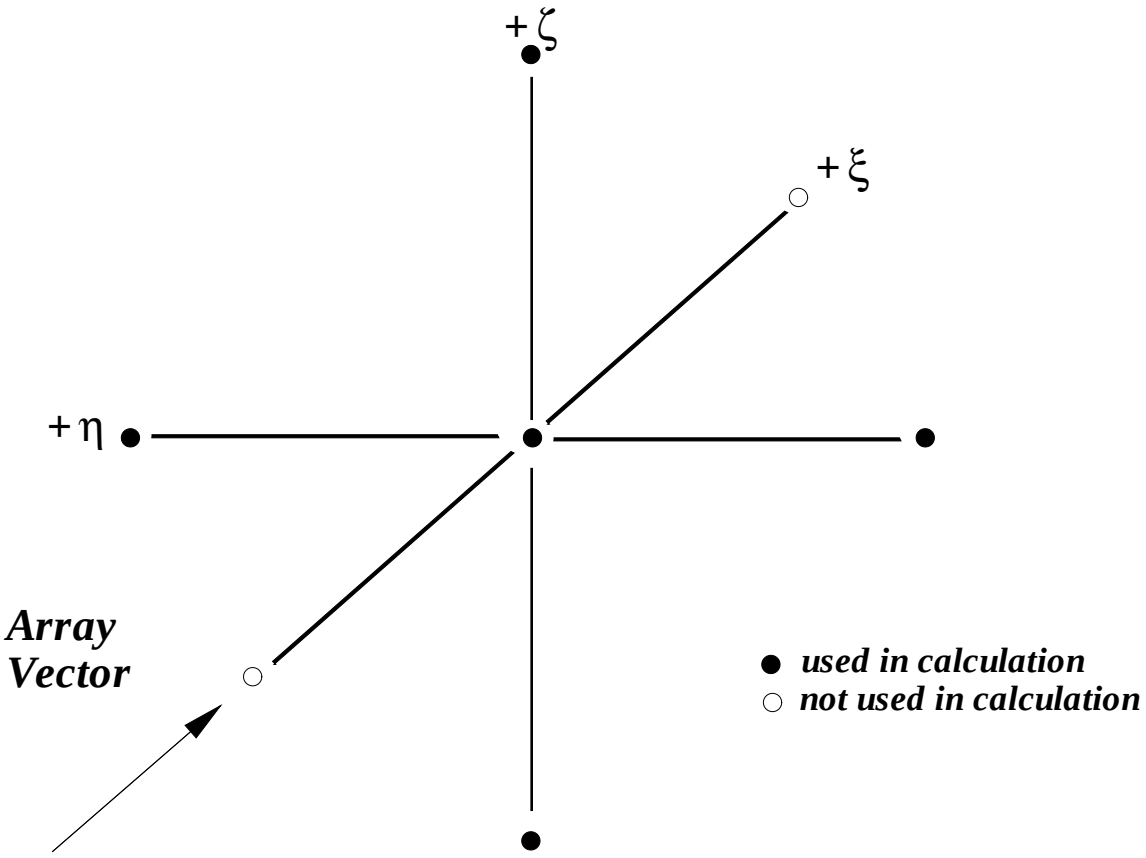


Figure 2: Grid points used in the calculation of the optimum relaxation parameter.

parameter. Although the PSOR is still used, the optimum relaxation parameter provides for a rapidly converged solution.

External Cell Size Specification

To calculate the forcing functions for orthogonality control in the elliptic equations, a cell size (Δs) has to be specified. The cell size represents the distance to the first point from a volume block face boundary. Although the 3DGRAPE code offers two ways to enter or calculate the cell size, there is not enough flexibility for complex block faces. The 3DMAGGS code incorporates a "flag" in the cell size specification of the face definition line, as illustrated in Table 6, to read these values from an external file.

Table 6. External Cell Size Specification Instruction Format.

Line no.:	Field no.:	Column nos.:	Datum type:	Description:
	1	1-5	k	"face"
	2	6	i	face number
	3	7-16	k	"-sections="
	4	17-18	i	number of sections comprising the face
	5	19-26	k	"-normal="
	6	27-38	c/f	"uncontrolled", "n-i-stations", cell height or "<peripheral>"
	7	39-43	k	"-abc="
	8	44-55	c/f	"keep-default" or stretching parameter
	9	56-68	k	"-light/tight="
	10	69-70	c	"ye" or "no"

```
face-2-sections= 1-normal=uncontrolled-abc=keep-default-light/tight=no

face-2-sections= 1-normal=123456789.12-abc=keep-default-light/tight=no

face-2-sections= 1-normal=<peripheral>-abc=keep-default-light/tight=no
```

The location of the cell size "flag" to 3DMAGGS is in field 6 and is given as "<peripheral>". 3DMAGGS looks for a file containing the necessary cell sizes for each point on the block face. This file has a naming construct to make them unique to 3DMAGGS. The file names take on the form:

DSIBLK δ[δ]Fγ.DAT

where,

$\delta[\delta]$ represents the block number for the system

and γ represents the face number of the specified block,

which follows 3DGRAPE conventions.

If block 3, face 4 is to have an orthogonality boundary condition under this method, the file would be named DSIBLK3F4.DAT. Because some blocking strategies utilize more than 9 blocks, the second optional β is a legal construct. Note that this method allows for the generation of as many blocks as the user desires. The user must provide the correct naming convention if any other methods are used besides the one outlined in PREMAGGS.

Utilizing this format, an outside program can be used to generate the cell sizes necessary for the calculation of the orthogonality controls. These cell sizes can be computed or extracted from a given CFD solution enabling apriori information to be incorporated into the volume grid. This form of cell size specification gives the user more flexibility in grid-point clustering and volume-grid structure.

Forcing Function Output

Due to the methods used in the interpolation of forcing functions, it became apparent that the user may wish to view these values calculated for volume control. An output "flag" was added to 3DMAGGS to output a TECPLOT^{TM} ASCII formatted file of the forcing functions. These forcing functions will only be written when the controls of Thomas and Middlecoff are formulated. The option, illustrated in Table 5, outputs three different TECPLOT^{TM} files. All files end the same way. They are configured such that the block number and the most current version of the file are in the file name. The beginning file name

is one of the following:

[1] "initPQR" - Initial derivatives calculated and interpolated onto the six defining faces of the grid block. This file also contains the parametric space forcing functions for those faces where they are directly calculable.

[2] "smthPQR" - Parametric space forcing functions after being smoothed with a weighted averaging algorithm.

[3] "fnlPQR" - Final forcing functions used on the defining six grid block faces, converted to physical space.

The file naming conventions account for multiple versions of the same file. For example, if block 12 was being written for a second time and there were 8 previous file names for block 12 after initialization, the initial forcing function data will be in "initPQRb12f10.dat". The number after "b" represents the block number, and the number after "f" is the file's version number. The creation of new files will be added in a consecutive order. Any pre-existing files will not be overwritten. The user must be sure the number of versions of a file do not exceed 99, because 3DMAGGS will stop if no more file versions are available. TECPLOT^{TM} can then be used to evaluate the magnitudes and to understand the forcing functions and their effects.

GRIDGEN Interface

In order to use the 3DMAGGS code, just as the 3DGRAPE code, a definition of the configuration's surface is needed. To make 3DMAGGS a complete system, a volume grid block and 2D parametric block-face grid generator are required. To provide this information, PREMAGGS, an interface code, was created to link GRIDGEN2D to 3DMAGGS.

The PREMAGGS code uses its own input file and GRIDGEN3D input data to generate the 3DMAGGS control decks. Other data required to run 3DMAGGS are provided by PREMAGGS, including:

[1] UNIX shell scripts for 3DMAGGS and 3DVOLCHK.

[2] Generation of 3DMAGGS control decks (files 10, 11 and 16).

[3] Δs cell sizing for Sorenson forcing function controls
using either 2DTFI or LARCS.

[4] Source code re-dimensioning of 3DMAGGS.

The PREMAGGS input file has the form shown in Table 7.

Table 7. GRIDGEN to 3DMAGGS Interface Input File.

```
***** PRE 3DMAGGS CONTROL FILE *****
Working directory of 3DGRAPE runs  (a)  :/scr/salter/wood/sc1/
FLAGS ctd,face,dsi,3dj,3dg,3dv     (6i2):  0 0 0 0 1 0
Job Control Deck for Cray or SGI   (a)  :cray
Configuration name                 (a)  :Straight Cone #1 for UPS Study
Default file name prefix           (a)  :sc1
Block Information file (*.bnda)    (a)  :sc1.bnda
Face Information file (*.mlga)     (a)  :sc1.mlga
Number of iteration sequences      (i2) :04
   Number      Global  Coarse (0)     Thomas
of Iterations  Control   Fine(1)    & Middlecoff
     000          0         1            0
     100          0         1            1
     100          1         1            2
     300          1         1            3
Relaxation parameter              (f12.6):-0.7
Decay rates for each block/face   (f12.6):6.0
 Block    Face   Decay Rate
Number  Number    Factor
   1       1       -1.00
   1       2        0.20
   1       3        0.35
   1       4        0.35
   1       5        0.30
   1       6        0.35
Sorenson init (1); 3DTFI (2)  (f12.6):  2.
Orthogonality Control         (f12.6):  6.
 Block    Face    Interp.   Interp.  Blending  Normalized
Number  Number  indx1->3  indx2->4  Function  Arc Lengths
   1       1        2         2         0          1
   1       2        1         1         3          1
   1       3        1         1         1          1
   1       4        1         1         1          1
   1       5        1         1         3          1
   1       6        2         2         2          1
```

The file is read with formatted FORTRAN statements for those line containing the colons ":" and the rest of the information is read with free formats. Two header lines are used for understanding the input file for the iteration control sequences, orthogonality decay rates and the calculation type for determining cell size. These header lines are expected and will

be read as 80 column character strings. The description of each line is tabulated in Table 8.

Table 8. Description of PREMAGGS Input File.

Line #	Format	Description
1-2	(a)	Header for the file.
3	(41x,a)	Directory to find all data, including the source codes. Note: If the directory has a ~ in front of it, the script written will be for a C-Shell, as opposed to the default Bourne Shell.
4	(41x,6i2)	Control flags for the types of data to be produced: Flag # Description 1 Control deck generation. 2 File 11 construction for "read-in-fixed" data. 3 Cell size calculation. 4 3DMAGGS UNIX script generation. 5 3DMAGGS re-dimensioned based on grid dimensions. 6 3DVOLCHK re-dimensioned.
5	(41x,a)	Type of machine 3DMAGGS will use.
6	(41x,a)	First comment line in the 3DMAGGS control deck, typically used to label the control file for clarity.

Table 8. Description of PREMAGGS Input File. (cont.)

Line #	Format	Description
7	(41x,a)	Default name of GRIDGEN and 3DMAGGS files.
8	(41x,a)	Truncated GRIDBLOCK ascii file name.
9	(41x,a)	GRIDGEN2D block face grid definitions.
10	(41x,i2)	Number of iteration sequences to be run in the "newstart" control deck.
10a	(a)	Header for columns of following data.
10b-?	(*)	Number of iterations, activation of orthogonality controls (1->YES/0->NO), coarse or fine solution, and type of Thomas and Middlecoff activation for a given sequence. The latter option and the specified 4 type:

Option #	Description
0	No TMC's are to be used or calculated.
1	TMC's used as initial guesses for P, Q & R.
2	Only TMC's are used for forcing functions.
3	Hybrid Control of Sorenson and TMC's.

11	(41x,f12.6)	Relaxation Parameter for solver. A negative number is that percentage of the optimum value. A positive number is a constant to be used.

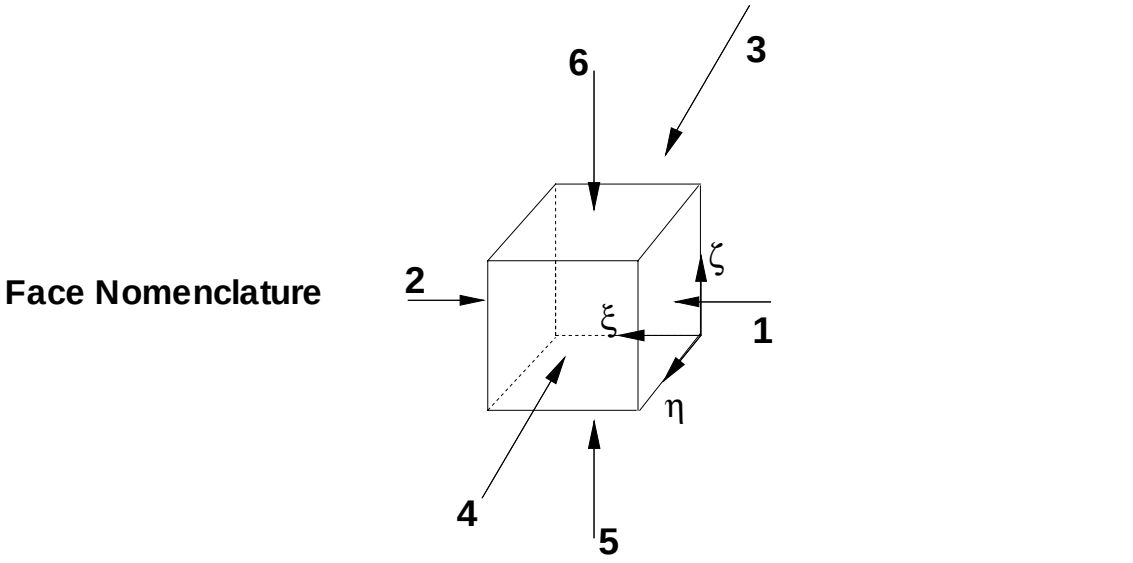
Table 8. Description of PREMAGGS Input File. (cont.)

Line #	Format	Description
12	(41x,f12.6)	Decay rate specification for the forcing functions. A negative number denotes the default. The default in the "newstart" deck is "keep-default". The default in the "restart" deck is the conversion from GRIDGEN to 3DMAGGS using the absolute value of the decay rate read as the value of the EXPO variable (equation 5), illustrated later. Otherwise a positive number is the number of block/face combinations that will use lines 12a-?.
12a	(a)	First line header for columns of following data.
12b-?	(*)	Grid block number, face number and decay rate to be used to exponentially decay the orthogonality controls into the volume. A negative number for the decay rate denotes 3DGRAPE's default for the specified block/face combination.
13	(41x,f12.6)	Volume grid initialization through Sorenson's method (#1), or optimized 3DTFI (#2).
14	(41x,f12.6)	Cell height control of each block/face combination. If the number is negative, the TEAM nomenclature and options within GRIDGEN, are used. In this case, only pole boundaries and matching faces have no control. All other face types will have orthogonality. If this number is positive, it represents the number of block/face combinations with controls to be specified in the following format:

Table 8. Description of PREMAGGS Input File. (concluded)

Line #	Format	Description
14a	(a)	First line header for columns of following data.
14b	(a)	Second line header for columns of following data.
14c-?	(*)	Block number, face number, type of interpolations for adjoining opposing faces (Figure 3) Linear (1), Elliptic (2), Hyperbolic blending of both opposing pairs, and use of normalized arc lengths for interpolations for each block/face combination.
	NOTE:	The computed cell heights/sizes are based on the LARCS method and each block/face combination will have a different file name, as mentioned in the **External Cell Size Specification** section of this paper. The PREMAGGS code outputs three files for each block/face combination, one for 3DMAGGS which utilizes the LARCS method, an alternate file that uses two-dimensional trans-finite interpolation where "tfi" is substituted for "dsi" in the file name convention and a third with a ".tcp" extension that is input to TECPLOT^{TM} to look at the cell sizes specified on a per point basis, as well as other LARCS parameters.

As mentioned in the description of the PREMAGGS input file, the default decay rate for the restart deck is the decay rate used by 3DMAGGS that emulates GRIDGEN3D's. This decay rate is easily determined by equating the method used by each code for a given face,



Opposing Face Nomenclature for LARCS Blending

ξ=Constant	η=Constant	ζ=Constant
Faces 1 & 2	Faces 3 & 4	Faces 5 & 6
Pair #1: 5 $\rightarrow$ 6	Pair #1: 5 $\rightarrow$ 6	Pair #1: 3 $\rightarrow$ 4
Pair #2: 3 $\rightarrow$ 4	Pair #2: 1 $\rightarrow$ 2	Pair #2: 1 $\rightarrow$ 2

Figure 3: Opposing Face Nomenclature.

such as ξ_{min} (equation 5).

$$P_{face_{\xi_{min}}} e^{-abc_{face_{\xi_{min}}}(\xi-1)} = P_{face_{\xi_{min}}} e^{-EXPO\frac{\xi-1}{\xi_{max}-1}} \tag{5}$$

Canceling the forcing function term, and taking the natural log of both sides results in equation 6.

$$abc(\xi-1) = EXPO\frac{\xi-1}{\xi_{max}-1} \tag{6}$$

Again, canceling the $(\xi-1)$, and substituting the EXPO in terms of "abc" yield the conversion between GRIDGEN3D and 3DMAGGS, equation 7.

$$EXPO = abc(\xi_{max}-1) \tag{7}$$

For the PREMAGGS input, the user must specify a negative EXPO value to use the GRIDGEN3D to 3DMAGGS conversion. For GRIDGEN3D, the default EXPO is 6.

Although the file seems involved and an added step, the PREMAGGS code provides a lot of flexibility in the grid-generation process. The transition between GRIDGEN and 3DMAGGS is smooth and efficient. This transition also enables the user to generate large grids easily and efficiently.

Grid Quality Parameter Calculation Tool

The second support code for the 3DMAGGS code is 3DVOLCHK. This program is designed to work with the data set from 3DMAGGS output, as well as any PLOT3D format file. The code has the same requirement on index/face nomenclature as 3DGRAPE. The 3DVOLCHK program calculates the volume of each cell, the cell's aspect ratio and three skewness parameters.

In order for 3DVOLCHK to calculate the above values, two specific elements are necessary. The user must provide the PLOT3D file name and notation of whether the PLOT3D file contains a single or multiple block grid. From this information, a "q" file, utilizing the name of the input volume grid file, is output. 3DVOLCHK incorporates the base name of the PLOT3D file to create an output file name.

The output "q" file then contains each cell's volume, the aspect ratio and the three skewness parameters, in this order. The volume is calculated by decomposing a hexahedron, to account for a cell's curvature.[11] The aspect ratio of each cell is then calculated by equation 8.

$$AR = \sum_{i=1}^{6} \frac{A_i}{6V^{\frac{2}{3}}} \tag{8}$$

Each area is calculated from the cross products of the vectors tangent to a given face, as in equation 9.

$$A_{\zeta=constantface} = \frac{1}{2}\|\vec{\mathbf{r}}_\xi \times \vec{\mathbf{r}}_\eta\| \tag{9}$$

Finally, the skewness of the grid lines can be determined by standard dot products. For example, looking at the plane where ζ is constant, skewness of grid lines are determined by

calculating the arc-cosine of the dot product for vectors $\vec{\mathbf{r}}_\xi$ and $\vec{\mathbf{r}}_\eta$ (equation 10.)

$$\theta_{\xi\eta} = \cos^{-1}\left[\frac{\vec{\mathbf{r}}_\xi \cdot \vec{\mathbf{r}}_\eta}{\sqrt{(\vec{\mathbf{r}}_\xi \cdot \vec{\mathbf{r}}_\xi)(\vec{\mathbf{r}}_\eta \cdot \vec{\mathbf{r}}_\eta)}}\right] \tag{10}$$

For the plane where ξ is constant, equation 11 is used. And for the plane where η is constant, equation 12 is used.

$$\theta_{\eta\zeta} = \cos^{-1}\left[\frac{\vec{\mathbf{r}}_\eta \cdot \vec{\mathbf{r}}_\zeta}{\sqrt{(\vec{\mathbf{r}}_\eta \cdot \vec{\mathbf{r}}_\eta)(\vec{\mathbf{r}}_\zeta \cdot \vec{\mathbf{r}}_\zeta)}}\right] \tag{11}$$

$$\theta_{\xi\zeta} = \cos^{-1}\left[\frac{\vec{\mathbf{r}}_\xi \cdot \vec{\mathbf{r}}_\zeta}{\sqrt{(\vec{\mathbf{r}}_\xi \cdot \vec{\mathbf{r}}_\xi)(\vec{\mathbf{r}}_\zeta \cdot \vec{\mathbf{r}}_\zeta)}}\right] \tag{12}$$

The skewness data written last in the "q" file, measures from zero to one, a skewness factor of 1.0 (100%) represents total orthogonality and the factor can decrease to 0.0 (i.e., 0% orthogonality). Equations 10, 11 and 12 are normalized by using equations 13, 14 and 15, to obtain the measures written.

$$ORTHO_{\xi\eta} = 1 - \frac{\left|\frac{\pi}{2} - \theta_{\xi\eta}\right|}{\frac{\pi}{2}} \tag{13}$$

$$ORTHO_{\eta\zeta} = 1 - \frac{\left|\frac{\pi}{2} - \theta_{\eta\zeta}\right|}{\frac{\pi}{2}} \tag{14}$$

$$ORTHO_{\xi\zeta} = 1 - \frac{\left|\frac{\pi}{2} - \theta_{\xi\zeta}\right|}{\frac{\pi}{2}} \tag{15}$$

The above measures were generated specifically due to the cosine function. The function produces a scale that has two places of no orthogonality and only one location of true orthogonality. Using the above formulation, the orthogonality exists at a value near one, and no orthogonality exists at zero. Orthogonality parameters become more meaningful and

easier to interpret.

From these parameters, the quality of the volume grid can be determined easily. Grid quality can be evaluated at the users discretion by utilizing 3DVOLCHK. There is no requirement to operate the entire 3DMAGGS code just to obtain grid quality information. Although there are some volume quality analysis tools available in visualizing software such as the Flow Analysis Software Tool[12](FAST), 3DVOLCHK adds the capability of employing TECPLOT^{TM} to view volume grids and quality parameters.

Example Problem

This portion of the manual has been included to enable the user to get a feel for the correct operation of the 3DMAGGS code along with its support codes PREMAGGS and 3DVOLCHK. The section consists of one example that extensively illustrates the full use of options added to 3DMAGGS.

The example used for all three codes is a sphere-cone-cylinder-cone configuration with an elliptical cross-section, Figure 4. This configuration was constructed using a quasi-axisymmetric grid generator. The configuration's dimensions are shown in Figure 5.

The method of volume grid generation typically used in conjunction with the 3DMAGGS code is the following:

[1] Construct or obtain the surface of a configuration.

[2] Load the surface geometry definition into GRIDBLOCK of the GRIDGEN code.

[3] Construct the grid-blocking structure to be used, as well as setting CFD boundary conditions and face-matching definitions.

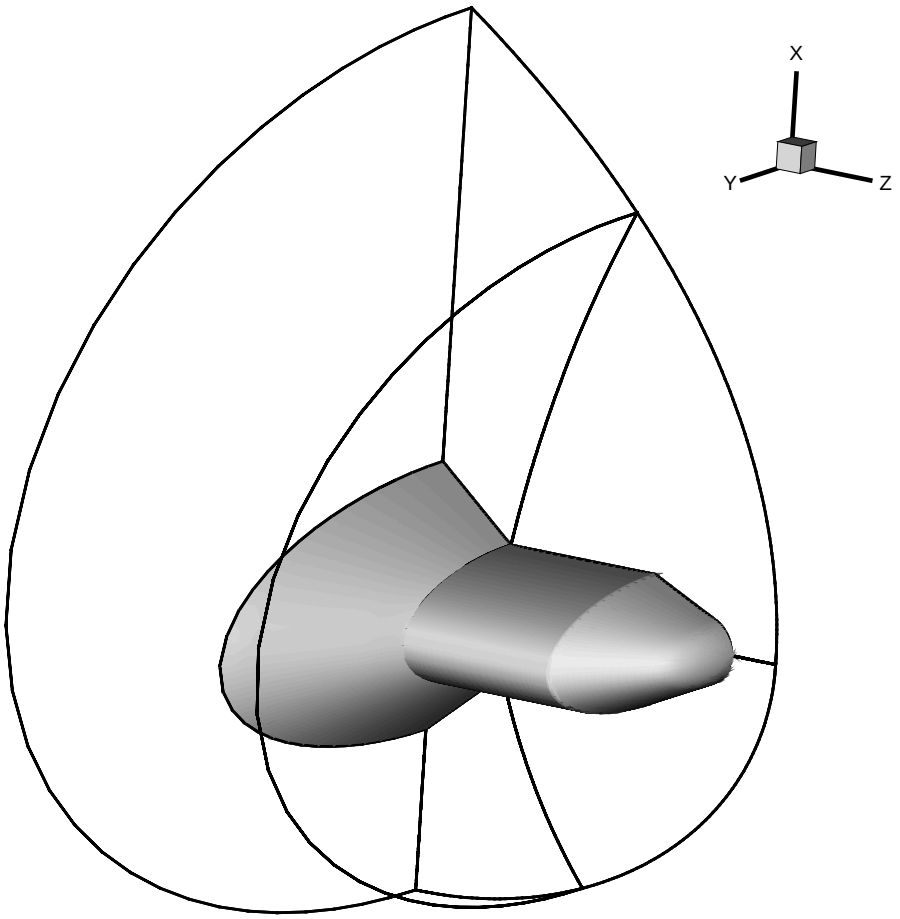


Figure 4: Example Sphere-Cone-Cylinder-Cone configuration.

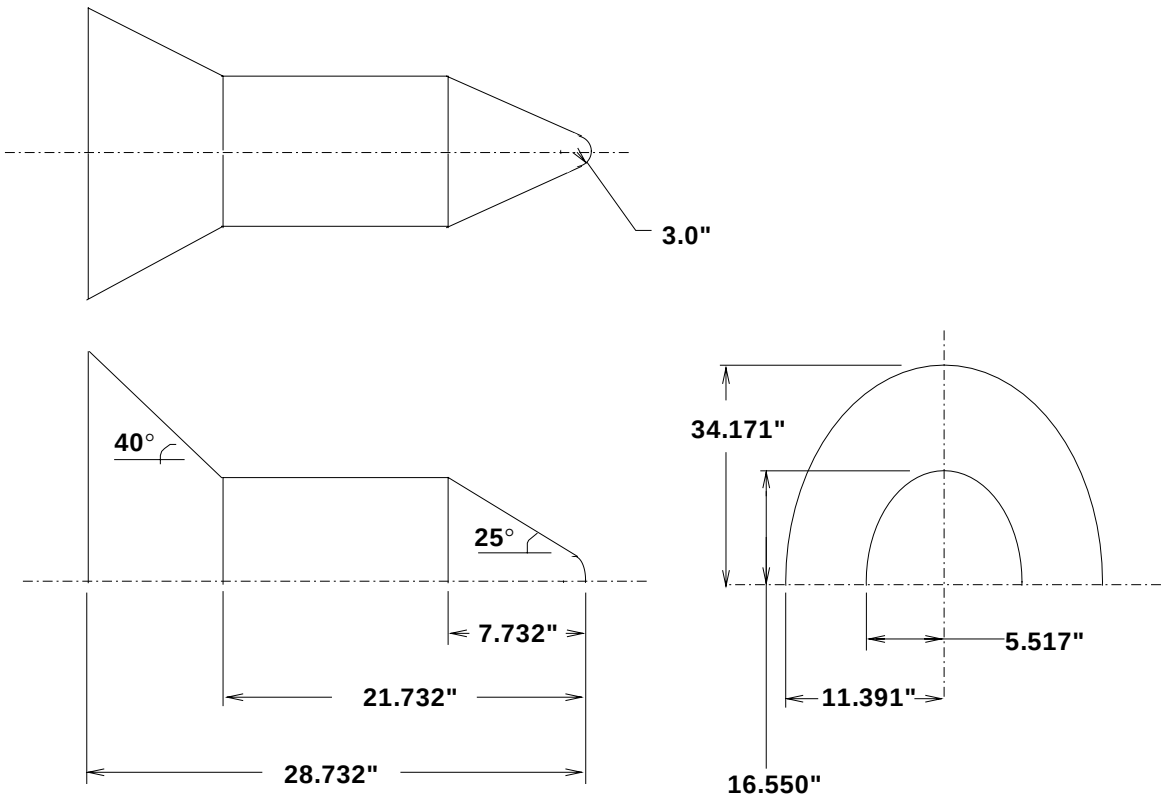


Figure 5: Scaled drawing of example configuration.

[4] Load the GRIDBLOCK output into GRIDGEN2D and create all defining faces of the grid-block structure. Note, there are six faces for each block.

[5] Output the face grid distributions (also referred to face definitions) and the boundary conditions to load into the GRIDGEN3D code.

[6] Set up the input file for PREMAGGS and run it with the input [file].bnda and [file].mlga files usually read by GRIDGEN3D.

[7] Compile, link and execute the 3DMAGGS code for the geometry to convergence or until the grid structure meets the needs of the user.

[8] Execute the 3DVOLCHK code to evaluate grid quality, and to determine if further iterations with the 3DMAGGS code is necessary.

The user may have to repeat the last 6 steps of the above method to obtain good grid distributions, or better parametric dimensional limits.

For the sample case, the face definitions, illustrated in Figure 6, were set up so the methods used for matching block boundaries were accounted and easily obtainable from the elliptic solver. Then the PREMAGGS input was set up, shown in A of the Appendix. PREMAGGS then generated the input decks used for the operation of 3DMAGGS and the UNIX scripts, shown in Appendix B, C, D and E.

By viewing the input, based on PREMAGGS assumptions, the matching boundary will have orthogonality activated. The 3DMAGGS code, in conjunction with the operation of the Thomas and Middlecoff controls, does not handle matching faces very well. This arises from the Thomas and Middlecoff controls calculated once at the beginning of each iteration

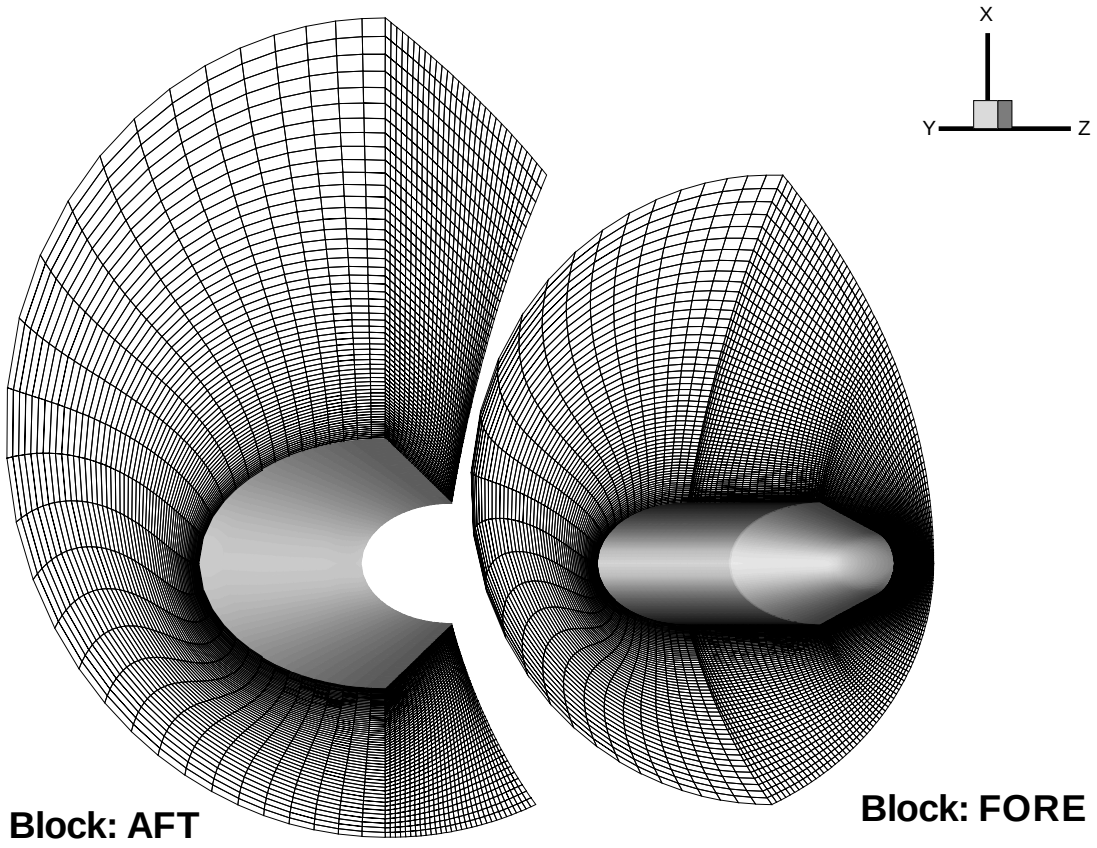


Figure 6: Block face definitions for the Example case with ζ limit faces and $\xi = 1$ face of AFT block not shown, for clarity.

group. Due to the methods used for matching boundaries, the surface that is shared by both blocks will be different than the one used to originally formulate the Thomas and Middlecoff controls. This difference in surface tends to cause instability in 3DMAGGS and can be extreme enough to prevent convergence. For matching boundaries, the orthogonality is activated. The decay rate of the forcing function from that face onto the interior of the volume grid is extremely fast. This causes little effect of the orthogonality but does result in a reasonable slope continuity across a matching boundary.

For the example case, Figure 7 illustrates the convergence of the root-mean squared residual grid-point movement, and Figure 8 shows the elliptic solver residuals, both on a per iteration basis. Just as expected, the solution proceeds smoothly towards convergence. One special point to note is that, in the beginning of the solution, the root mean squared grid-point movement oscillates very little. This is indicative of starting the solution at a guess determined by Thomas and Middlecoff, as opposed to stating the forcing function terms at zero. This effect combined with the grid lines tending to be more indicative of the final grid, through the use of 3DTFI initialization, tends to make the 3DMAGGS solver more stable and faster.

3DMAGGS does not handle matching boundaries with the Thomas and Middlecoff controls activated. This problem can be alleviated by using orthogonality at the matching boundary, with a rapid decay rate of the required forcing functions, onto the interior of the grid. Utilizing 3DMAGGS for the volumetric grid generation is not only faster but can give the user the necessary control to enable the formation of the Thomas and Middlecoff controls to be computed once.

Comparisons to GRIDGEN3D and 3DGRAPE

A primary motivation for extending the capabilities of 3DGRAPE to 3DMAGGS was the lethargy of GRIDGEN3D in elliptic volume grid generation. At the time of 3DMAGGS' conception, several complex aerodynamic configurations were being studied, and GRIDGEN3D was unable to deliver a usable volume grid efficiently. The efficiency was also effected by the lack of state-of-the-art controls within the 3DGRAPE code. Hence, the state-of-the-art volumetric grid-generation controls of GRIDGEN3D were incorporated into the most efficient available elliptic solver, 3DGRAPE. This section illustrates the advantages of using 3DMAGGS over both 3DGRAPE and GRIDGEN3D by comparing to the example case in this document.

For GRIDGEN3D, the boundary conditions used in the comparison were:

[1] The matching face boundary was set to have orthogonality in both blocks.

[2] Orthogonality was imposed on the symmetry planes, the exit plane and the configuration's wall.

[3] Thomas and Middlecoff controls were set at the beginning of the solution.

[4] An optimum relaxation factor of 75% was used for grid point movement.

[5] Default decay rates were applied to all boundaries with orthogonality activated, except the matching boundary. This capability is not available in GRIDGEN3D, so an EXPO was chosen to be more indicative of the grid point clustering on boundaries with orthogonality activated.

[6] 1000 iterations were to be performed.

Under these conditions, the results from using GRIDGEN3D (input in Appendix F) and 3DMAGGS, Figure 9 illustrates the convergence of the root mean squared residual grid point movement to the iteration. While both seem to perform similarly, they are different in execution time (see Figure 10).

Inspecting the GRIDGEN3D output, the circled regions in Figure 10, illustrate an important difference between 3DMAGGS and GRIDGEN3D. GRIDGEN3D does not save the elliptic solver forcing functions or residuals for a restart. GRIDGEN3D tries to recalculate the controls at each restart. 40% of the time necessary to obtain a converged solution could be lost by GRIDGEN3D. Due to capabilities in restart and execution time, 3DMAGGS is a more efficient solver.

To remain consistent with GRIDGEN3D, similar matching boundary conditions were set, as noted earlier. The matching boundary shown in Figure 11, in GRIDGEN3D tends to produce a wave-type phenomenon in the grid. This wave-like structure arises from the lack of source-term controls within GRIDGEN3D. By comparison, 3DMAGGS produces a slope-continuous transition across the boundary.

Source-term decay in 3DMAGGS from a surface with activated orthogonality is different than GRIDGEN3D. The 3DMAGGS code uses a unique value for the rate of decay per grid point from a surface of orthogonality and is illustrated in equation 16.

$$\begin{aligned} P = & P_{face_{\xi_{min}}} e^{-abc_{face_{\xi_{min}}}(\xi-1)} + P_{face_{\xi_{max}}} e^{-abc_{face_{\xi_{max}}}(\xi_{max}-\xi)} \\ & + P_{face_{\eta_{min}}} e^{-abc_{face_{\eta_{min}}}(\eta-1)} + P_{face_{\eta_{max}}} e^{-abc_{face_{\eta_{max}}}(\eta_{max}-\eta)} \\ & + P_{face_{\zeta_{min}}} e^{-abc_{face_{\zeta_{min}}}(\zeta-1)} + P_{face_{\zeta_{max}}} e^{-abc_{face_{\zeta_{max}}}(\zeta_{max}-\zeta)} \end{aligned} \tag{16}$$

The "abc" is a user specified or default decay rate on a per face basis. By utilizing this formulation, greater control is available for near-boundary grid clustering. By comparison, GRIDGEN3D uses parametric limits in the formulation and a constant decay rate per entire grid block system (equation 17).

$$
\begin{aligned}
P = & P_{face_{\xi_{min}}} e^{-EXPO\frac{\xi-1}{\xi_{max}-1}} + P_{face_{\xi_{max}}} e^{-EXPO\frac{\xi_{max}-\xi}{\xi_{max}-1}} \\
& + P_{face_{\eta_{min}}} e^{-EXPO\frac{\eta-1}{\eta_{max}-1}} + P_{face_{\eta_{max}}} e^{-EXPO\frac{\eta_{max}-\eta}{\eta_{max}-1}} \\
& + P_{face_{\zeta_{min}}} e^{-EXPO\frac{\zeta-1}{\zeta_{max}-1}} + P_{face_{\zeta_{max}}} e^{-EXPO\frac{\zeta_{max}-\zeta}{\zeta_{max}-1}}
\end{aligned} \tag{17}
$$

3DMAGGS eliminates any biasing in the formulation of non-zero source terms for orthogonality control. A face, whose parametrically orthogonal direction has a larger number of points relative to surrounding faces, will not necessarily be the dominant controlling force for the volume grid appearance. Thus, 3DMAGGS provides the user with a greater degree of control over the forcing function controls on the boundary and interior volume grid, as compared to GRIDGEN3D.

GRIDGEN3D offers only one other capability not offered by 3DMAGGS, the Fixed-Grid computation. This capability exists primarily to smooth the results of algebraic grid generation. To account for the lack of a Fixed-Grid capability in 3DMAGGS, the controls over the forcing functions are more numerous than GRIDGEN3D. These controls include the determination of the decay rate of forcing functions into the interior of a volume grid. The effective smoothing of Thomas and Middlecoff controls at the time of their formulation, reduces "kinked" grid production. Hence, with the use of small relaxation parameters for grid-point movement and with a larger variety of grid clustering controls, 3DMAGGS does

not need the Fixed-Grid controls.

In comparison with 3DGRAPE (input in Appendix G and H), the optimization, the 3DTFI initialization, and the initial guess of source terms based on Thomas and Middlecoff controls afford better convergence of 3DMAGGS over 3DGRAPE (Figure 12). With the Thomas and Middlecoff controls deactivated for the volumetric control, the 3DMAGGS code uses available grid generation time efficiently. This efficiency is due to the grid points being moved based on their optimum relaxation as opposed to some fixed relaxation rate. With the state-of-the-art capabilities of 3DMAGGS, time to convergence is still maintained (Figure 13).

Conclusion

In summary, the 3DMAGGS code is more versatile and robust than GRIDGEN3D and 3DGRAPE. The extensions to 3DGRAPE found in 3DMAGGS offer a larger variety of elliptic solver enhancements without sacrificing execution time. These extensions provide precision to the user for controlling both volumetric and near surface grid clustering. The robust nature of 3DMAGGS over GRIDGEN3D, as well as faster execution of the code, results in a reduction in time and resources spent on grid generation.

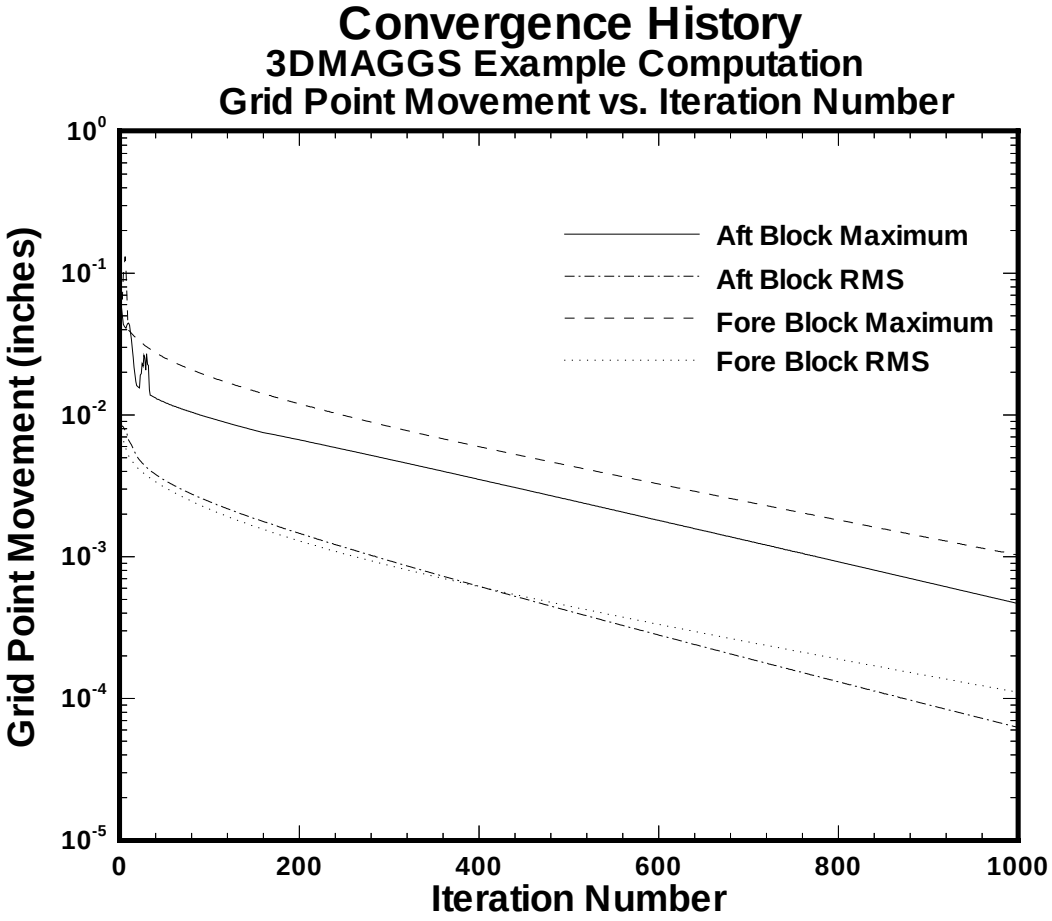


Figure 7: 3DMAGGS convergence history of grid point movement per iteration.

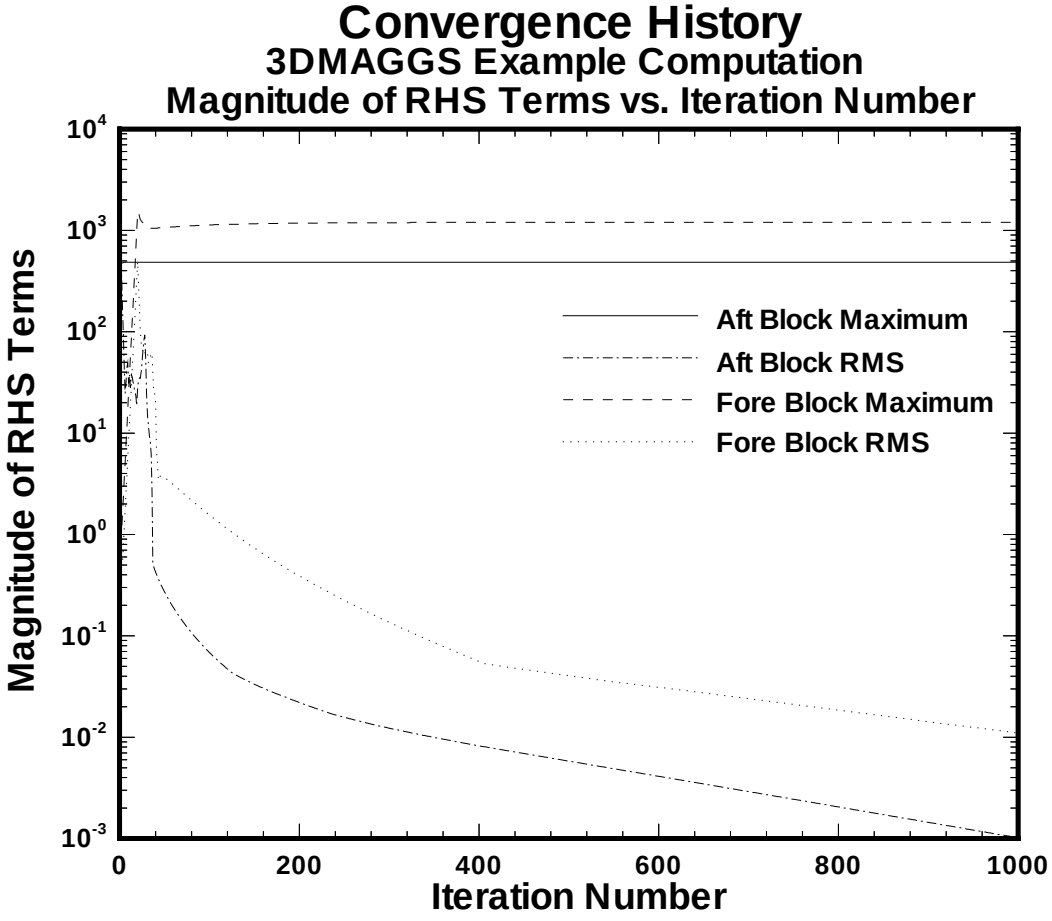


Figure 8: 3DMAGGS convergence history of elliptic solver's Right Hand Side terms per iteration.

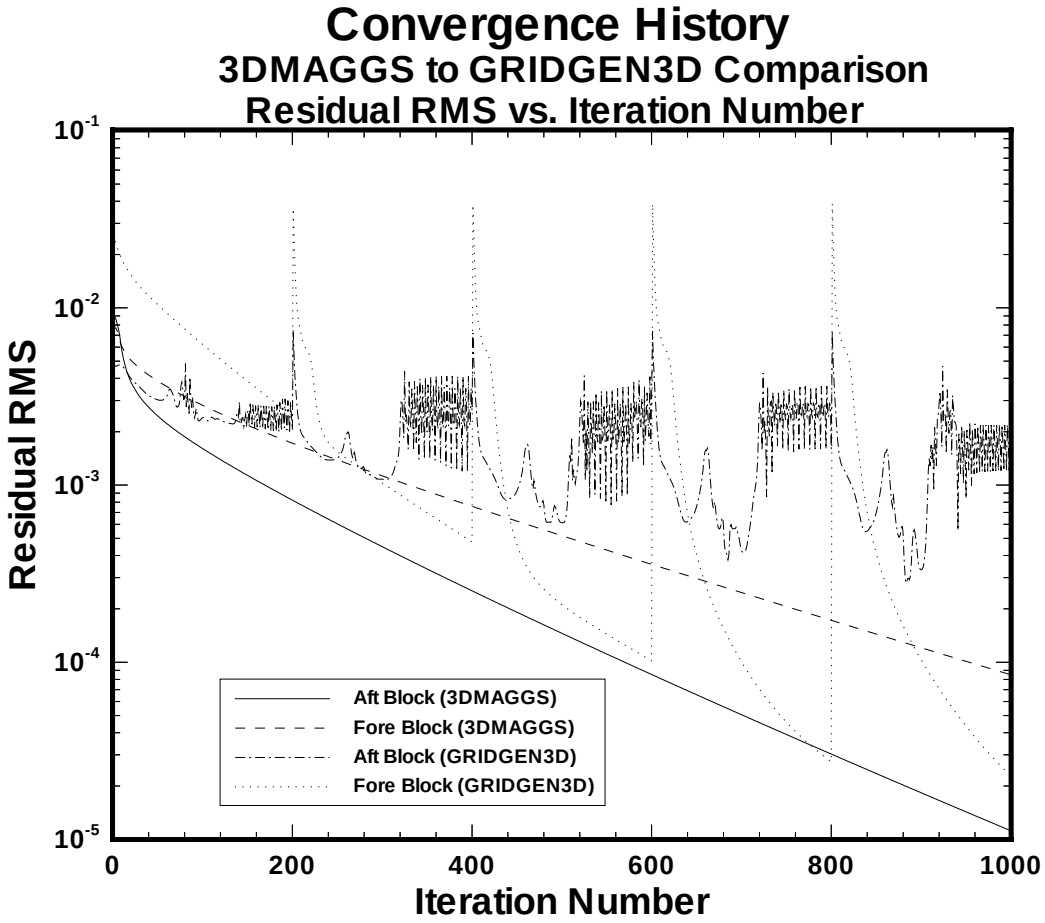


Figure 9: Comparison of 3DMAGGS and GRIDGEN3D convergence history per iteration.

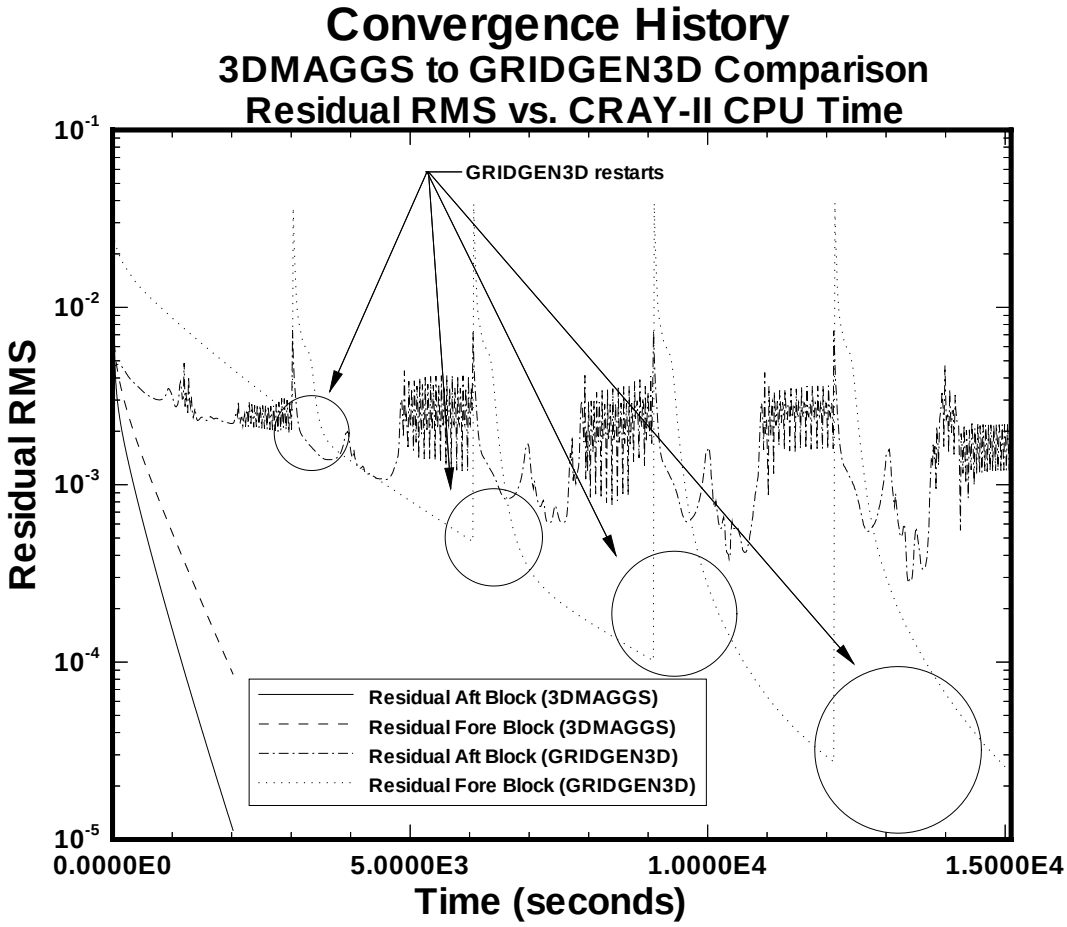
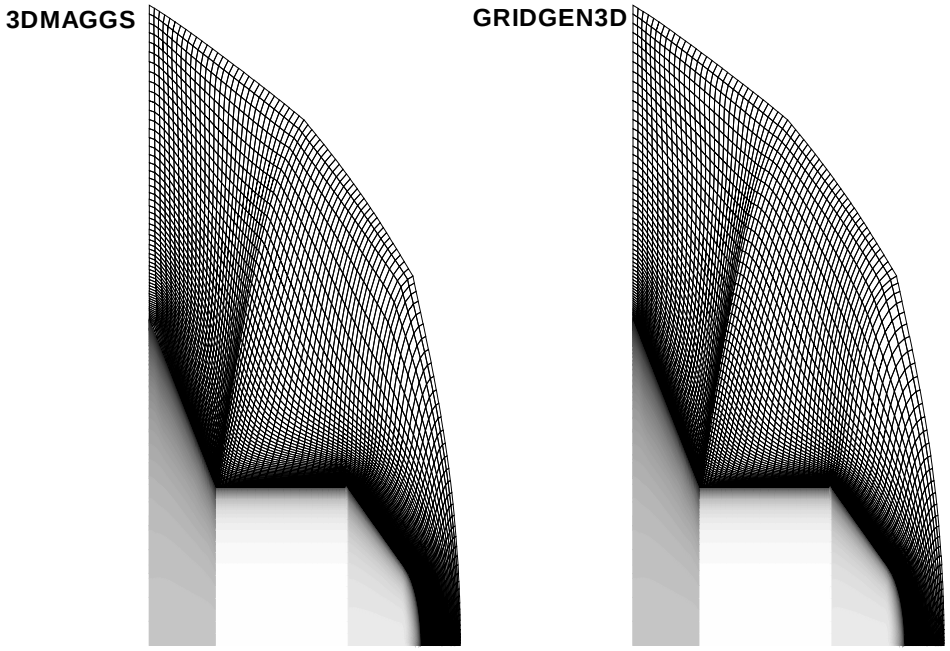
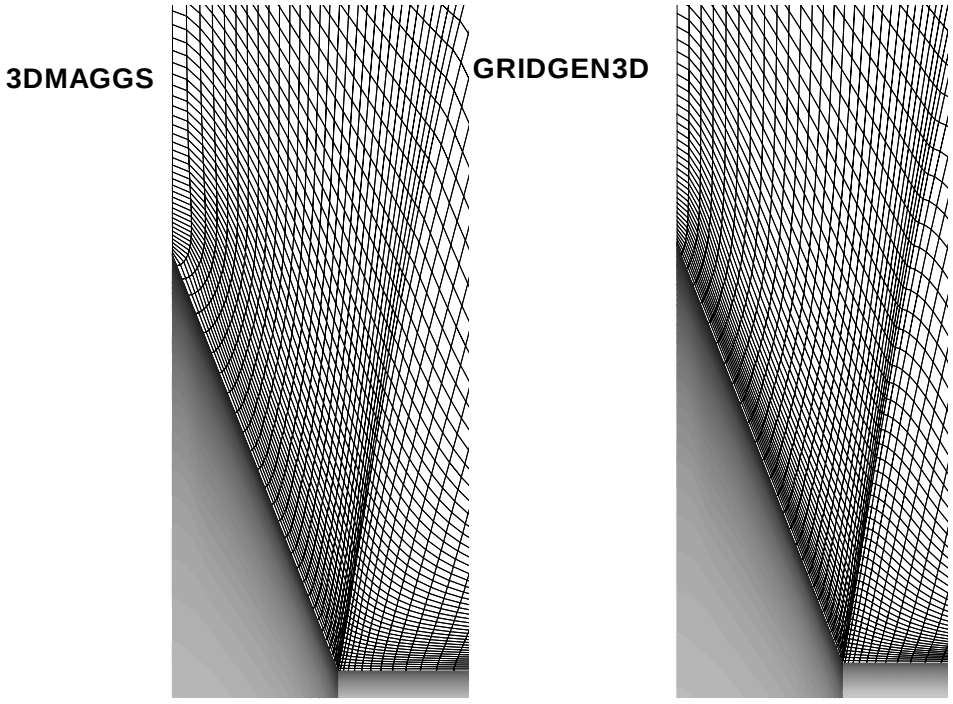


Figure 10: Comparison of 3DMAGGS and GRIDGEN3D convergence history per second.



(a) 3D solver influence of volume grid.



(b) Expanded view of actual boundary.

Figure 11: Comparison of 3DMAGGS and GRIDGEN3D matching boundary conditions.

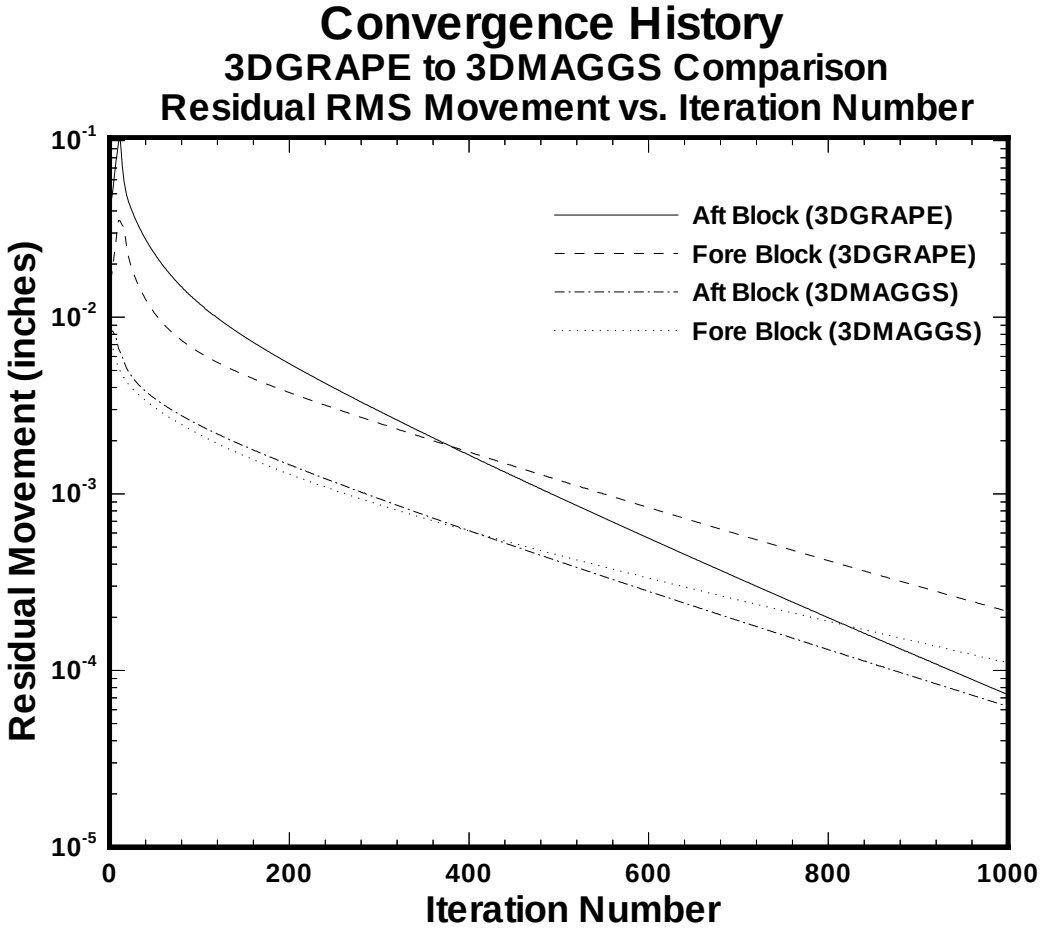


Figure 12: Comparison of 3DMAGGS and 3DGRAPE convergence history in iterations.

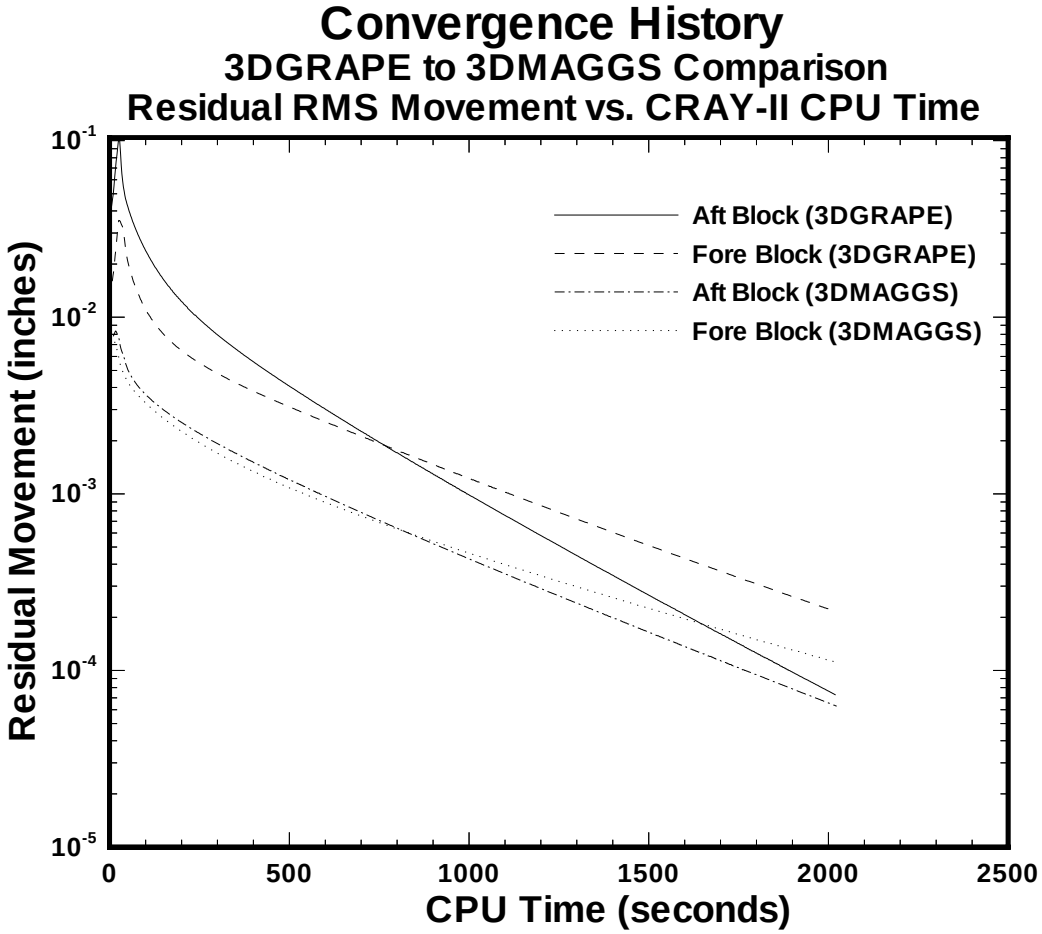


Figure 13: Comparison of 3DMAGGS and 3DGRAPE convergence history in CPU time.

References

[1]R. L. Sorenson, "The 3DGRAPE Book: Theory, Users' Manual, Examples," NASA TM 102224, July 1989.

[2]J. P. Steinbrenner, J. R. Chawner, and C. L. Fouts, "The GRIDGEN 3D Multiple Block Grid Generation System," Wright Research and Development Center Report WRDC–TR–90–3022, October 1989.

[3]J. F. Thompson, "A Composite Grid Generation Code for General 3D Regions-the EAGLE Code," *AIAA Journal*, vol. 26, pp. 1–10, March 1988.

[4]B. K. Soni, "Two– and Three–Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics," AIAA Paper 85–1526, 1985.

[5]P. D. Thomas and J. F. Middlecoff, "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations," *AIAA Journal*, vol. 18, pp. 652–656, June 1979.

[6]J. P. Steinbrenner, J. R. Chawner, and D. A. Anderson, "Enhancements to the GRIDGEN System for Increased User Efficiency and Grid Quality," AIAA Paper 92–0662, January 1992.

[7]R. L. Sorenson and K. McCann, "GRAPEVINE: Grids About Anything by Poisson's Equation in a Visually Interactive Networking Environment," NASA Conference Publication 3143, April 1992.

[8]S. J. Alter and K. J. Weilmuenster, "Cell Volume Control at a Surface for Three-Dimensional Grid Generation Packages," in *Software Systems for Surface Modeling and Grid Generation* (R. E. Smith, ed.), vol. CP-3143, pp. 273–298, NASA, 1992.

[9]I. Amtec Engineering, "Tecplot: version 5 User's Manual," Amtec Engineering publica-

tion V5.0/92-14, January 1992.

[10]L. W. Ehrlich, "An Ad Hoc SOR Method," *Journal of Computational Physics*, vol. 44, pp. 31–45, March 1981.

[11]W. Kordulla and M. Vinokur, "Efficient Computation of Volume in Flow Predictions," *AIAA Journal*, vol. 21, pp. 917–918, June 1983.

[12]P. P. Walatka, J. Clucas, R. K. McCabe, T. Plessel, and R. Potter, *FAST User Guide*. Mountain View, CA.: NASA Ames Research Center, first ed., 1992.

Appendix A
Input File to PREMAGGS

```
***** PRE 3DMAGGS CONTROL FILE *****
 Working directory of 3DGRAPE runs  (a)  :/scr/salter/3dmaggs/ex/
 FLAGS ctd,face,dsi,3dj,3dg,3dv     (6i2):  0 1 1 0 0 1
 Job Control Deck for Cray or SGI   (a)  :cray
 Configuration name                 (a)  :CONE for test case of 3DMAGGS
 Default file name prefix           (a)  :exn
 Block Information file (*.bnda)    (a)  :exn.bnda
 Face Information file (*.mlga)     (a)  :exn.mlga
 Number of iteration sequences      (i2) :01
    Number      Global   Coarse (0)     Thomas
of Iterations  Control    Fine(1)    & Middlecoff
     400          1          1            3
 Relaxation parameter             (f12.6):  1.
  Block    Face   Decay Rate
 Number  Number    Factor
    1       1        2.00
    1       2       -1.00
    1       3        0.35
    1       4        0.35
    1       5        0.30
    1       6        0.35
    2       1       -1.00
    2       2        2.00
    2       3        0.35
    2       4        0.35
    2       5        0.30
    2       6        0.35
 Decay rates for each block/face  (f12.6):-1.
 Sorenson init (1); 3DTFI (2)     (f12.6):  2.
 Orthogonality Control            (f12.6):  8.
```

Block Number	Face Number	Interp. indx1->3	Interp. indx2->4	Blending Function	Normalized Arc Lengths
1	1	2	2	3	1
1	2	1	1	3	1
1	3	1	1	1	1
1	4	1	1	1	1
1	5	1	1	3	1
1	6	2	2	3	1
2	1	2	2	0	1
2	2	1	1	3	1
2	3	1	1	1	1
2	4	1	1	1	1
2	5	1	1	3	1
2	6	2	2	3	1

Appendix B
File 10 Data for Sample 3DMAGGS Comparison to GRIDGEN3D

```
run-comment          CONE for test case of 3DMAGGS
run-comment          Initialization file
number-of-blocks= 2-number-of-parts-in-iteration-schedule= 1
iterations=999-control=ye-coarse/fine=fine-thomas-middlecoff=hybrid
filename-11-input=exn1.face      -filename-12-output=
filename-14-grid-output=exn1-1.vol     -form=plot3d
write-for-restart=ye-filename-15-output=exn1.res
relaxation-param=-.7500000000-forcing-function-output=no
initialization=trans-finite-iterations= 16


block- 1-comment    aft
dimension-j= 31-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31
```

```
block- 2-comment    foremid
dimension-j= 71-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=uncontrolled-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31
```

Appendix C
File 16 Data for Sample 3DMAGGS Comparison to GRIDGEN3D

```
run-comment          CONE for test case of 3DMAGGS
run-comment          Restart file
filename-17-input=exn1.res
number-of-parts-in-iteration-schedule= 1
iterations=400-control=ye-coarse/fine=fine-thomas-middlecoff=hybrid
filename-11-input=exn1.face      -filename-12-output=
filename-14-grid-output=exn2.vol       -form=plot3d
write-for-restart=ye-filename-15-output=exn2.res
relaxation-param=keep-default-forcing-function-output=no


block- 1-comment    aft
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31
```

```
block- 2-comment    foremid
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=uncontrolled-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31
```

Appendix D
UNIX script for 3DMAGGS on CRAY-II

```
# user=login pw=nobusiness        # username and password on cray
# qsub-r 3dmaggs                  # request name
# qsub-o 3dmaggs1.out             # output file name
# qsub-lt 3600                    # time limit
# qsub-lm 8mw                     # memory limit
# qsub-eo                         # send stderr to stdout
# qsub-s /bin/csh                 # use a c shell
#
#
#   Job file to execute 3DMAGGS
#
#     ################################################
#     ##   The user should only change the input &  ##
#     ##   output files below this line.  Remember  ##
#     ##   this is a BETA release of 3DMAGGS!       ##
#     ################################################
#
#
#.......change to working directory
cd /scr/salter/3dmaggs/ex/
#
#.......issue the 3dmaggs execution command
#
time ./3dgs < 3dg1s.inp >> 3dg1s.out
#
#.......check volume grid for negative volumes.
#
./3dvchk < 3dv1.inp >> 3dg1s.out
```

Appendix E
Input Information to 3DVOLCHK

```
exn1.vol
s
```

Appendix F
Input File Data for GRIDGEN3D

```
   $assem
          iassem = 1,
          fnboci = 'exn0.bnda',
          fnsurf = 'exn.mlga',
          pathtmp = '/tmp',
          fnvoli = 'exn0.vol',
   $end
   $init
          initial = 2*2,
   $end
   $ellip
          maxit = 200,
          relax = .5,-.7,
          icon = 2*2,
          igrape(1,1) = 2,
          igrape(1,2) = 2,
          igrape(1,3) = 2,
          igrape(1,4) = 1,
          igrape(1,5) = 2,
          igrape(1,6) = 2,
          igrape(2,1) = 2,
          igrape(2,2) = 2,
          igrape(2,3) = 0,
          igrape(2,4) = 2,
          igrape(2,5) = 2,
          igrape(2,6) = 2,
          expo = 24,
          imulti = 0,
          ivec = 1,
   $end
$out
       iwrite = 1,
       ivue = -1,
       istyle = 2,
       iflo = 0,
       iasc = 0,
       fnboco = 'exn0.bnda',
       fnvolo = 'exn1.vol',
       fnvue = 'exn1.vue',
       fnflo = 'exn1.flo',
$end
```

Appendix G
File 10 Data for Sample 3DMAGGS Comparison to 3DGRAPE

```
run-comment          CONE for test case of 3DMAGGS
run-comment          initialization file; comparison to 3dgrape
number-of-blocks= 2-number-of-parts-in-iteration-schedule= 1
iterations=999-control=ye-coarse/fine=fine-thomas-middlecoff=initial
filename-11-input=exn1.face      -filename-12-output=
filename-14-grid-output=exn1-4.vol     -form=plot3d
write-for-restart=ye-filename-15-output=exn1.res
relaxation-param=   -.7000000-forcing-function-output=no
initialization=trans-finite-iterations= 16


block- 1-comment    aft
dimension-j= 31-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31
```

```
block- 2-comment    foremid
dimension-j= 71-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=ye-iterate-block=ye

face-1-sections= 1-normal=uncontrolled-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31
```

Appendix H
File 10 Data for 3DGRAPE

```
run-comment          CONE for test case of 3DGRAPE
run-comment          initialization file
number-of-blocks= 2-number-of-parts-in-iteration-schedule= 1
iterations=999-control=ye-coarse/fine=fine-thomas-middlecoff=none
filename-11-input=exn1.face      -filename-12-output=
filename-14-grid-output=exn1-3.vol     -form=plot3d
write-for-restart=ye-filename-15-output=exn1-3.res
relaxation-param=keep-default-forcing-function-output=no
initialization=keep-default-iterations= 0


block- 1-comment    aft
dimension-j= 31-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=no-iterate-block=ye

face-1-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 31-k-from-  1-to- 31
```

```
block- 2-comment    foremid
dimension-j= 71-dimension-k= 31-dimension-l= 61
handedness=r-initcond=l-cart/sph=cartesian
thomas-middlecoff=no-iterate-block=ye

face-1-sections= 1-normal=uncontrolled-abc=keep-default-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-2-sections= 1-normal=<peripheral>-abc=2.0000000000-light/tight=no
read-in-fixed-xyz-k-from-  1-to- 31-l-from-  1-to- 61

face-3-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-4-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-l-from-  1-to- 61

face-5-sections= 1-normal=<peripheral>-abc=0.3000000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31

face-6-sections= 1-normal=<peripheral>-abc=0.3500000000-light/tight=no
read-in-fixed-xyz-j-from-  1-to- 71-k-from-  1-to- 31
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1993	Technical Memorandum

4. TITLE AND SUBTITLE

The Three-Dimensional Multi-Block Advanced Grid Generation System (3DMAGGS)

5. FUNDING NUMBERS

506-40-91-02

6. AUTHOR(S)

Stephen J. Alter and Kenneth J. Weilmuenster

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

NASA TM-108985

11. SUPPLEMENTARY NOTES

Alter: Lockheed Engineering & Sciences Co., Hampton, VA and Weilmuenster: Langley Research Center, Hampton, VA.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Unclassified-Unlimited

Subject Category 34

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

As the size and complexity of three dimensional volume grids increases, there is a growing need for fast and efficient 3D volumetric elliptic grid solvers. Present day solvers are limited by computational speed and do not have all the capabilities such as interior volume grid clustering control, viscous grid clustering at the wall of a configuration, truncation error limiters and convergence optimization residing in one code. A new volume grid generator, 3DMAGGS (Three-Dimensional Multi-Block Advanced Grid Generation System), which is based on the 3DGRAPE code, has evolved to meet these needs. This is a manual for the usage of 3DMAGGS and contains five sections, including the motivations and usage, a GRIDGEN interface, a grid quality analysis tool, a sample case for verifying correct operation of the code and a comparison to both 3DGRAPE and GRIDGEN3D. Since it was derived from 3DGRAPE, this technical memorandum should be used in conjunction with the 3DGRAPE manual (NASA TM-102224).

14. SUBJECT TERMS

Computational grids
Grid generation (mathematics)

15. NUMBER OF PAGES

72

16. PRICE CODE

A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		