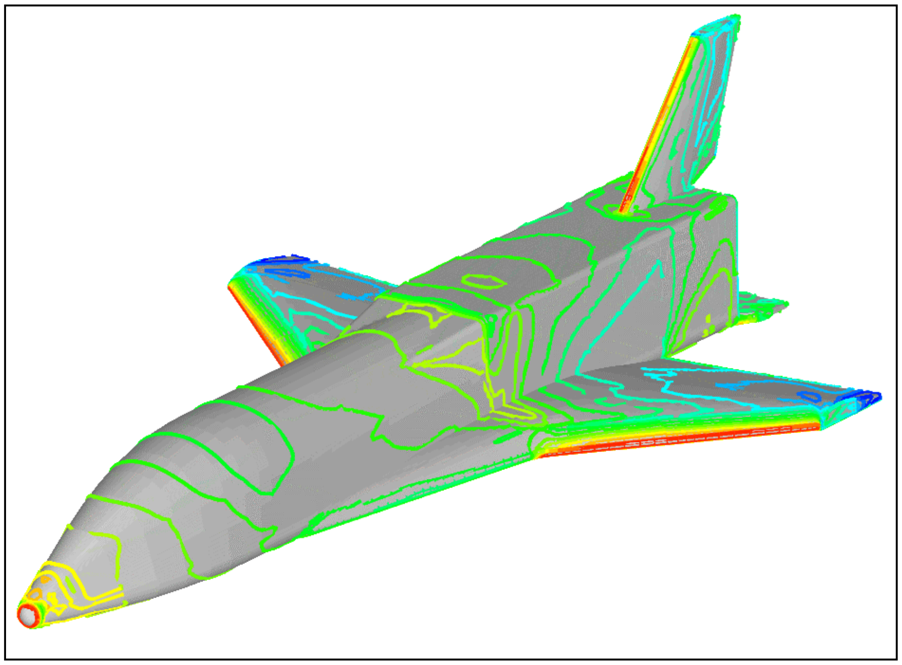
PAB3D v4.0

User Manual

Document Number: AS&M-0307-PAB3D



Analytical Services & Materials, Inc.

107 Research Drive, Hampton, VA 23666

Phone: (757) 865-7093; Fax: (757) 865-7309

URL: www.asm-usa.com

Multi-Block CFD Code for Complex Aerodynamic Configurations PAB3D© v4.0 User Manual

Document Number: AS&M-0105-PAB3D

Dr. Alaa Elmiligui
Senior Scientist

ANALYTICAL SERVICES & MATERIALS, INC.
107 Research Drive, Hampton, VA 23666
Phone: (757) 865-7093; Fax: (757) 865-7309

Printed in the United States of America on July 20, 2001

Table of Contents

Section 1.0 Welcome

Welcome to PAB3D User Manual, published by Analytical Services & Materials, Incorporated (AS&M). PAB3D was developed by Dr. Khaled S. Abdol-Hamid who is currently pursuing his research work at NASA Langley Research Center. This manual is for users of PAB3D v4.0 software. We have divided this manual into five (5) Sections and four (4) Appendices.

Section 2 introduces PAB3D and its salient features, including the names and details of Input/Output (I/O) files and the latest developments in PAB3D.

Section 3 describes the contents of Solver Control File. We have shown the input lines bold-faced and italicized. Sample cases follow the content description. As far as possible, we have ensured that the variable names precede the values in this section.

Section 4 describes the contents of User Control File. We have shown the input lines bold-faced and italicized. As far as possible, we have ensured that the variable names precede the values in this section.

Section 5 describes the Solution Procedure. This section describes the commands needed to execute PAB3D.

Appendix A provides a description of the real gas models used in PAB3D to calculate Gamma (γ), the ratio of internal energy to local enthalpy.

Appendix B provides the performance data of PAB3D under a cluster of distributed computers (MPI implementation).

Appendix C describes the procedure for solving an example problem using PAB3D.

Appendix D provides a bibliography of all the articles referred to in developing and implementing various features of the PAB3D code since 1988.

Section 2.0 General Overview

2.1 Introduction

PAB3D flow solver is widely used in the U.S. aerospace industry for propulsion component design, aircraft system analysis, and environmental quality studies including jet engine acoustics. The solution methodology embodied in the PAB3D code can be applied to a much wider range of problems for general industry and academia as a research tool or a teaching aid. The original development and improvement of PAB3D [1–5] was under contracts from the NASA Langley Research Center and the GEAE Company. This code has been used to simulate complex aerodynamic flow configurations and is currently being used in several national programs such as:

1. High Speed Research (HSR)
2. Advanced Subsonic Technology (AST)
3. Large Engine Technology (LET)
4. CFD-Based Aero-Acoustic Prediction Method
5. CFD-Based Model Mixer Design and Analysis

PAB3D has several built-in timesaving routines including grid sequencing and customized computer memory requirements that permit the user to quickly obtain a converged solution. PAB3D uses advanced turbulence models to determine the Reynolds Stress terms [6–9] in the governing equations. There are several state-of-the-art two-equation and algebraic Reynolds Stress turbulence models implemented in the PAB3D code. PAB3D is also capable of simulating different gases (species) simultaneously for Real Gas simulations. The species concentrations are used to evaluate equivalent thermodynamic and viscous parameters in the flow governing equations. All scalar equations (turbulence and species concentration) are solved uncoupled from the mean flow governing equations. This approach keeps the scheme partially implicit with a reduction in computational time.

The PAB3D code has an option for space marching scheme [10–12]. The interface flux in the streamwise direction is determined by separate terms, depending on the quantities on the left (upstream) and the right (downstream) sides of the interface. The downstream term, which is a significant part of any elliptic problem, has a value of zero for hyperbolic or supersonic problem and can be ignored without introducing significant flow solution for parabolic problem. By ignoring the downstream dependence terms in the Roe scheme, the solver becomes the space-marching scheme. Under the modified scheme, a solution is obtained plane by plane from upstream to downstream by carrying out a sufficient number of implicit iterations in each plane. A solution for the entire computational domain is established in a single sweep. When the space-marching option is used for jet flow computations as conditions permit, the computer time is less than one-twentieth of the time required for obtaining a time marching solution with the same flow condition. The error introduced due to these different solvers is practically indistinguishable.

PAB3D uses either natural or specified location to transition the flow from laminar to turbulent. Turbulent calculations do not require any special initialization procedure for stable computation. The code uses a flexible mesh sequencing procedure. Typical solutions will require 800 iterations on a twice-coarsened mesh level, 400 iterations on a once coarsened mesh level, and 200 iterations on the finest mesh level. For example 1,000,000 grid points require 25-30 hours using an SGI R10000/195 MHz workstation.

Advanced turbulence models [3–5] are needed to get accurate representation of the aerodynamic characteristic of the dual separate flow nozzles under investigation. PAB3D uses the two-equation turbulence model and the more advanced Algebraic Stress Models (ASM). To compare the PAB3D, NPARC, and WIND codes; John H. Glenn Research Center recently used algebraic turbulence models to predict multi-stream flow characteristic. The ASM [5] produces the best prediction of jet/plume mixing compared to experimental data and to the standard two-equation turbulence models. PAB3D is one of the few codes available that uses advanced turbulence models in the simulation of complex compressible three-dimensional flows.

PAB3D requires less memory than most other three-dimensional codes. The memory requirement for using a two-equation turbulence model is less than **23-words/grid point**. There is no additional memory required for the algebraic stress models implemented in the PAB3D code. The code speed (based on 1,000,000 grid points on a single processor C90 computer) varies with the selected solver: two-factor, three-factor (block) and three-factor (scalar).

1. Two-factor 17 µs/iteration/grid point

2. Three-factor (block) 19 µs/iteration/grid point

3. Three-factor (scalar) 7 µs/iteration/grid point

The code uses moderate size temporary arrays, which are quite efficient on workstations: three-factor (block) on SGI/R10000/ 190MHz runs at 110 µs/iteration/grid point.

Over 50 publications describe the use of the advanced turbulence models in the PAB3D code to simulate complex 3D flows (see http://www.asm-usa.com). The code is widely used by NASA Langley Research Center, NASA Glenn Research Center, NASA Dryden Flight Research Center, General Electric (GEAE), Pratt & Whitney (P&W), and Boeing for simulating complex aircraft configurations, engines, and inlets. Those simulations require the use of advanced turbulence models.

Users have used PAB3D to simulate several different three-dimensional mixed-flow nozzles [6–8]. Each of these configurations required around 3,000,000 grid points to resolve 25 diameters downstream from the nozzle exit and included the entire spreading jet in the radial directions. The solutions were used for aerodynamic and acoustic prediction. Results were in excellent agreement with the experimental data. On a single processor SGI R10000, each case required approximately 72 hours of CPU time.

There are several ways to reduce elapsed time:

1. One approach is to use distributed computers or a multiprocessor computer. The MPI (Message Passing Interface) version of PAB3D was used to produce a solution for an equivalent nozzle-exhaust problem with CHEVRON (noise suppression device). It used 6 HP 9000 computers and got a converged solution in approximately 12 hours.

2. Another approach is to solve the nozzle (less than 500,000 grid points) using time marching, (which requires less than six hours on a single CPU) technique followed by using a space marching technique to complete the jet/plume solution. John H. Glenn Research Center was able to get a space marching solution for an equivalent of 1,000,000 grid points in less than two CPU hours on a 333 MHz Intel PC which is 80% the speed of SGI R10000/195 MHz workstation.

The user may refer to other publications relevant to the usage and/or the development of PAB3D [13–64].

2.2 Salient Features

This manual describes the latest version of the PAB3D code. In this version, we have attempted to transform PAB3D to be increasingly user-friendly. We have also reduced the number and size of the files.

The following are some of the features of the current version (v4.0) of PAB3D:

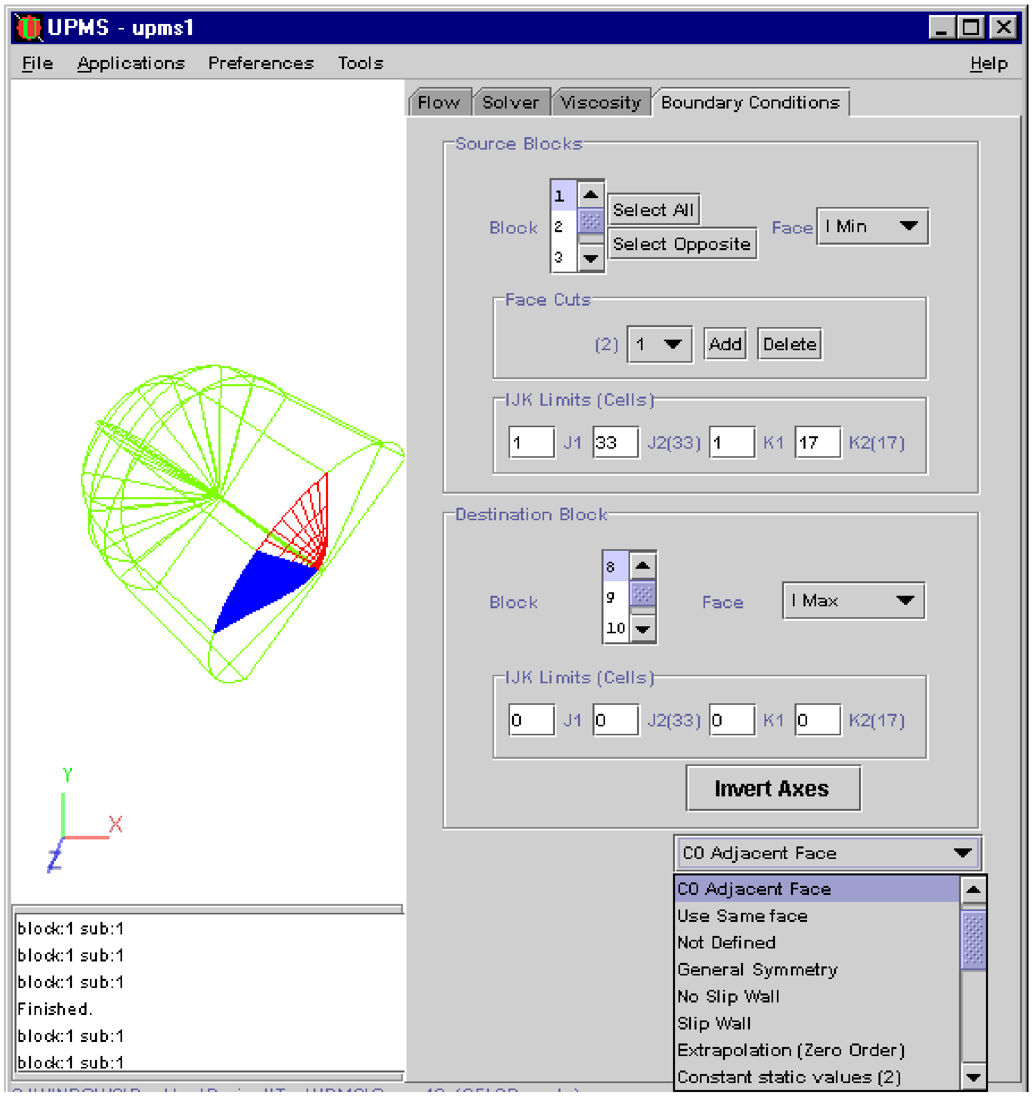
- allocates memory for the PAB3D and its utilities dynamically through Dynamic Memory Allocation routines. This feature allows the user to compile the programs only once for each computer platform. also uses a more efficient scaled dynamic memory for distributed computer simulations.

- allocates temporary memories for the blocks, which turned off then releases them when using distributed computing mode.

- uses first- or second-order time-accurate scheme for unsteady flow simulations

- allows the use pf a set of generalized harmonic, jet boundary conditions for total pressure and velocity is provided

- contains Wilcox K-ω also among other turbulence models
- allows the use of any two-equation turbulence models at block level
- allows the use of different flux approximation variables at block level
- creates Restart files for each sequence level. This cuts down the expense in written a full grid restart files for a coarse grid solution.
- calculates the diffusion terms in all three directions with the following two exceptions:
  - This option should only be used with 3-factor or diagonalization approach
  - I-direction is not available for multi-species simulation
- contains independent and in code post-processing utility using Dynamic Memory Allocation
- maintains a convergence history for each plane when the Space Marching Scheme is used
- uses less number of I/O files compared to other CFD codes.



Source Blocks: Block 1 (2, 3); Face I Min; Face Cuts (2) 1; IJK Limits (Cells): 1 J1, 33 J2(33), 1 K1, 17 K2(17)

Destination Block: Block 8 (9, 10); Face I Max; IJK Limits (Cells): 0 J1, 0 J2(33), 0 K1, 0 K2(17)

C0 Adjacent Face

```
block:1 sub:1
block:1 sub:1
block:1 sub:1
Finished.
block:1 sub:1
block:1 sub:1
```

Figure 1. GUI for PAB3D

- offers compatibility with several Operating System platforms:
- Unix Computers: Cray, DEC, HP, IBM, SGI and SUN
  - Linux Computers: DEC and Intel
  - PCs: Win95/98, WinNT and Mac

A new program "AutoG3d" developed by Dr. Paul Pao (757-864-3044) at NASA Langley Research Center is now a part of PAB3D. This program reads the grid file and generates control and map files with information about patching. In addition, we have just released a Graphical User Interface (GUI), called "Universal Process Management System (UPMS)". UPMS can be used to generate control files for several CFD codes including PAB3D, see **Fig. 1**.

To maintain the backward compatibility of PAB3D, the control file maintains the following inputs. The inputs are active in earlier versions of PAB3D while they remain inactive in the current version.

1. **`'init.d','user.cont':`** PAB3D v4.0 will use fixed name init.d and user.cont. Users do not have to run Ginitmn, because we have replaced it with Gibc, which is activated by PAB3D.
2. **`Igrid`, `Irisio`,** and **`inorm`:** PAB3D v4.0 does not use these routines, because users can get PLOT3D output using POST.
3. **`iperf1`:** PAB3D v4.0 does not use this routine, because the performance is handled differently using POST.

2.2.1 Input

The input to PAB3D consists of four (4) unformatted and two (2) ASCII files, which the users need to keep in the working directory. The four unformatted files are:

1. ***grid file***: This file contains all the information about the grid, connectivity, and memory. The user must create this file.
2. ***restart file(s)***: PAB3D will create a directory, which contains the information about solution variables and boundary condition.
3. ***'init.d'***: This file contains the initial conditions, which will be created by the user.cont section of the User Control File.
4. ***'initb.d'***: This file contains the boundary conditions, which will be created by the user.cont section of the User Control File.

Users need to create two ASCII (solver and user control) files that will be used to generate at least 75% of the data required to run PAB3D. This manual describes the options associated with these files.

2.2.2 Output

The PAB3D output contains many files. We have provided the names of important output files and their contents in **Table 1**. In future versions, we will attempt to remove all unwanted output files.

In this release, PAB3D does not have the capability to output files in PLOT3D format. However, we offer a post-processing utility program 'POST' along with PAB3D to enable users to visualize the results. A separate documentation fully describes the functionality of POST. PAB3D users can conveniently view the species concentration and gas property results using POST. POST can write any PAB3D variable and many additionally calculated quantities to PLOT3D and binary TECPLOT formats in either cell-centered or edge-point formats. POST currently contains most of the scalar field functions found in

Table 1. Important Output File Names and their contents

Fixed Name	File Content
resid.out	"restart directory"/resid.out monitors the total residual for the NS and Turbulent transport equations.
Diag.out	Within 1 global, PAB3D will output the locations of the maximum NS residuals for each block and iteration.
header.dat	"restart directory"/header.dat Contains control parameters for the specific run

FAST and PLOT3D, with additional support for spatially varying real gas constant (R) and ratio of specific heats (γ), due to the presence of multiple species. The interface is text based with extensive defaults and many shortcuts. Batch processing is available via automatically generated script files. Users can write the results in PLOT3D or binary Tecplot format. POST also supports functions that are defined only on a subset of the domain, such as boundary layer parameters or forces. Currently, in POST, y+ is the only sub-domain function available, but in future releases this area will be greatly expanded to provide an engineering toolbox of specialized functions not usually found in general purpose plotting packages.

Section 3.0 Solver Control File `tpab3d.cont`

3.1 General Requirements for All Blocks

In this part of the control file, general information such as file names, solver scheme specifications, number of iterations, and global sets are specified. A single line label for identification precedes each group of input.

3.1.1 Header Version Control

This is a subset of 1.50 release of the PAB3D code. The syntax is

```
"!PAB150"
```

3.1.2 Geometry file

This is one of the two main unformatted files. The syntax is

```
"Grid File"
gpab3d.d
```

This geometry file `"gpab3d.d"` contains the following segments (see **Fig. 2**):

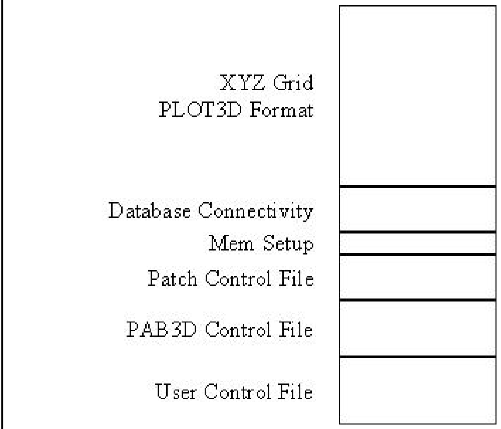


Figure 2. Segments of Geometry File

- Grid file in PLOT3D format & Database information
- Memory information needed for the future release of PAB3D dynamic memory
- Patch control file
- Last PAB3D solver and user files used in solving the present case.

3.1.3 Restart File

PAB3Dv1.5 uses a restart file name but PAB3Dv4.0 uses a restart directory name. User can specify up to 20 characters for the file/directory name. If PAB3D is unable to find the restart file, it will use the `"Init File"` to fill up the dependent variables arrays (ρ, ρu, ρv, ρw, and ρe). The syntax is

```
"Restart File"
restart.d
```

The restart file(s) directory `"restart.d"` contains a set of grid and solution files. Each represents an active block and names BlocN.q. Where is N is the block number in three digit format, 001 for example. Each of these files has the following segments (see **Fig. 3**):

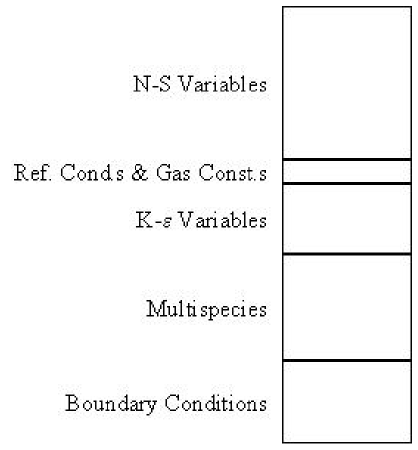


Figure 3. Segments of Restart file

- N-S '5' conservative variables in plot3d format for each block
- Turbulence variables for each block
- Species densities for each block

- h/e (Energy ratio)
- Specific heat ratio, γ, for each block
- N-S, Turbulence model and Species boundary condition variables for each block

The first two lines of the restart file are in this form:

```
Ngrid
(idim, jdim, kdim, ib=1,Ngrid),
5+Ntrv+Nspv, (Nseq, ib=1,Ngrid), Nspv
```

where,

- `Ngrid` - Number of blocks
- `idim` Number of grid points in the I-direction
- `jdim` Number of grid points in the J-direction
- `kdim` Number of grid points in the K-direction
- `Nseq` Grid sequence level for each block used in the last solution
- `Ntrv` =2 if two-equation model option is utilized
- `Nspv` Number of species variables

3.1.4 Init and user files

The syntax of this line is

```
"INIT File"
'init.d','user.cont'
```

`init.d` specifies initial/boundary conditions values (ρ, ρ*u, ρ*v, ρ *w, and e). Using the `Gibc.f` utility, a simple input specification generates this file.

`user.cont` deals with all the special options available in using the PAB3D code (see Section 4). The INIT FILE names are ignored in ***all versions of PAB3D released after 2.0*** because, in these versions, `init.d` and `user.cont` are fixed names.

3.1.5 Turbulence Initialization Flag and Generic Solver Specifications

The syntax of this statement is

```
nte year 'F hh:mm'
0 97 '1 00:00'
```

The variables **`nte`, `year`, `'F hh:mm'`** are defined as:

- `nte` is the *k*-ε variable initialization flag, where:
- <0 = initialization by iterating twice
- 0 = no
- `year` is the release year of pab3d code '97'
- **`F hh:mm`**
  - 0 = select the hour and minute (hh:mm 07:21 is 7 am and 21 minutes) for the code to stop and write restart file
  - 1 = select the number of hours and minutes (clock time). For example, 19:20, the code will continue running for 19 hours and 20 minutes.

If hh and mm are all zeros the code will run using the number of iterations specified in the next sections.

3.1.6 Zone, Check, and Factorization Options

The syntax of the statement is

```
nzone ichk ischeme
1 1 2
```

The variables **`nzone`**, `ichk` and `isch` are defined as:

- `nzone` is the number of zones; 1 (single zone) for general multiblock grids.
- **cfl scheme**: `ichk` (see **Table 2**)
- **Factorization** : `isch` (see **Table 3**)

A typical input line has values of `1, 1, 2`.

3.1.7 Global Iteration Sets and Number of Processors

The syntax of this statement is

Table 2. Definitions for **`ichk`**

Display **`cfl scheme`**	Definition	Pab3D **`Ichk`**
CFL (0)	Don't check for sudden drop density and temperature	0
CFL (ρ)	Change CFL/dt if density (ρ) exceeds 50% of the density at the previous time-step. This is very important option in the first few iterations during the transition from initial condition	1
CFL (T)	Change CFL/dt if temperature exceeds 50% of the density at the previous time-step. This is very important option in the first few iterations during the transition from initial condition	2
CFL (ρ & T)	Change CFL/dt if density and/or temperature could exceed 50% of the density at the previous time-step. This is very important option in the first few iterations during the transition from initial condition, Advances of Flow Simulation Techniques, Davis, CA 1997	3

Table 3. Definitions for **`isch`**

Display **Factorization**	Definition	Pab3D **`isch`**
2-Factor	Two-factor scheme. This scheme proved to give very fast convergence for supersonic/subsonic mixed flows. The implicit scheme is restricted to the (*j, k*) planes, AIAA-87-1113.	2
3-Factor	Three-factor scheme. It should give faster convergence especially for full subsonic flows. This is a block diagonal scheme using LU decomposition to inverse the diagonal terms, AIAA-87-1113.	3
Scalar	Scalar three-factor scheme. It should give the fastest convergence especially for attached flows. This is a block diagonal scheme using LU decomposition to inverse the diagonal terms.	4

```
ngit Isafe Iauto
5 1 2
```

The variable **`ngit`** is defined as the number of global set of iterations. At the end of each global iteration, an interim restart file (**`flow.d`**) is written. This feature provides a route for periodic examination of performance parameters, and protects the intermediate solutions from being totally lost in case of an abnormal PAB3D job termination. This is particularly important for long jobs submitted to the batch queue.

The **`Isafe`** (v1.4) option is used with MPI implementation. PAB3D uses blocking send procedure and either blocking receive (**`Isafe=0`**) or non-blocking receive (**`Isafe=1`**). The non-blocking receive produces less communication time between computer nodes. However, we found that the non-blocking option with LAM implementation of MPI does not work with the DEC Alpha computers and we have to use a blocking receive (10% slower). Additionally, the MPICH implementation of MPI does not yield correct results with non-blocking option. However, the MPICH implementation gives faster message passing time than the LAM 6.1 for both blocking and non-blocking options.

The **`Iauto`** is used to distribute blocks to different computer nodes. If **`Iauto`**

- = 0, the blocks are distributed by their numbers
- = 1, the blocks are distributed by their sizes trying to balance the load to different processors (recommended)
- = 2, user specifies load distribution through the use of MPI section in the **`user.cont`**.

3.1.8 Number of Iterations in Each Global Sets

The syntax of the statement is

```
nit
10*100
```

The variable **`nit`** is defined as the number of iterations for **`ngit`** sets. It is required to specify one value for each global set. A typical input can be 10*100, i.e., 100 iterations for each of the ten global sets. The maximum number of iterations allowed per global is 5000.

3.1.9 Repeated Specifications for Each Zone and their Blocks

In the earlier versions of the PAB3D code, the user can divide the grid into multiple zones. The multiple zone structure is designed for having multi-block / multi-zone space marching scheme in mind. In **Fig. 4**, we show the computational domain of PAB3D. In the space marching scheme, the multi-block structure in each zone is restricted to (J, K) directions only. The grid planes must be aligned in the (I direction). Hence, `idm` is the same for all the blocks in a given zone.

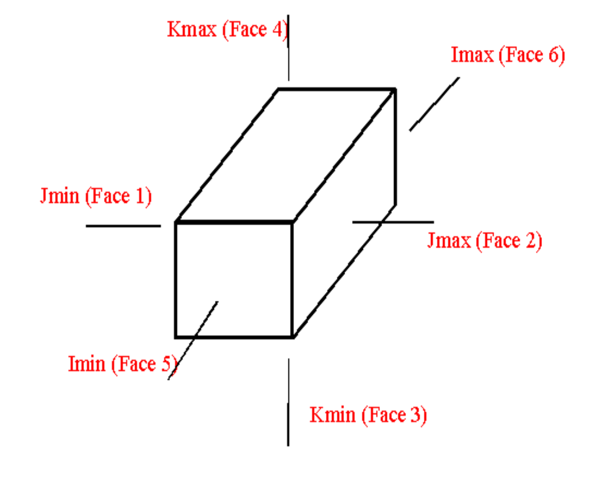


Figure 4. PAB3D Computational Domain

The general multi-block structure in the PAB3D code allows general block connectivity between blocks. Each block can have any dimension, grid topology, and/or the spatial orientation of that dimension. In this case, the entire block can be put in one zone. For each zone (generally, in a multi-block, there is only one zone), there are two lines of input: number of iterations and block dimensions specifications.

3.1.9.1 Number of Iterations for Each Zone

The syntax of this statement is

```
nitz
10*1
```

The variable `nitz` is defined as the number of zonal iterations for `ngit` sets, where:

- 1 turns zone on
- 0 turns zone off

N 1 can be used, but it is not necessary (see `nit`)

3.1.9.2 Number of Blocks and Block Dimensions

The syntax is

```
#Block | ib (Block #),
idim,jdim,kdim,nitb,nseq,dt,dtmb
5
1 2 9 81 1 444 -1.000 0.000
2 2 113 81 1 444 -1.000 0.000
3 2 121 81 1 444 -1.000 0.000
4 2 121 41 1 444 -1.000 0.000
5 2 9 41 1 444 -1.000 0.000
```

This is the input line for the number of blocks and block dimensions: `ib`, `jdim`, `kdim`, `nitb`, `nseq`, `dt`, `dtmn` repeat iblock times for each iblock, idim repeat iblock times for each block.

The variables are defined as:

- `ib`: block number (5).
- `idim`: the number of grid in I-direction for a given block.
- `jdim`: the number of grid in J-direction for a given block.
- `kdim`: the number of grid in K-direction for a given block.
- `nitb` number of iterations for this block "`ib`". `nitb=0`, it represents the block switched off. For the space marching solution, nitb is the number of iterations per plan (I-plane)
- `nseq` is defined as a three-digit integer (`InJnKn`), where one value is required for each global set of iterations (see *ngit*).
- `In` is the grid reduction factor in the I-direction.

- `Jn` is the grid reduction factor in the J-direction.
- `Kn` is the grid reduction factor in the K-direction.

When the variable `nseq = 0`, it is equivalent to `In = Jn = Kn = 1`, or `111` (full grid). Other examples include:

A grid reduction by a factor of 2 in `(I, J, K)` is `222`.

A grid reduction by a factor of 2 in `(I, J)`, with a full grid count in `K`, is `221`.

- `dt`: `CFL/dt` of this block if `nprc1` or when using space marching scheme.
- `Dtmb`: `CFL` number allowed if `nprc1` or Order of Magnitude Reduction per plan when using space marching scheme.

This input line is in very simple form as compared with the last release of the PAB3D code.

3.2 Specifications for Each Block

3.2.1 Options for Flux Splitting Schemes, Iteration Type, Viscosity

The syntax of this statement is

```
ivfxj ivflux irst ivisc kturb ibs ibf
1 3 2 1 6 1 1
```

The variables, `ivfxj`, `ivflux`, `irst`, `ivisc`, `kturb`, `ibs`, and `ibf` are defined as:

- ***Jacobian:*** `ivfxj` **(Table 4)** is the flag for a Jacobian flux splitting scheme.
- ***Flux approximation :*** `ivflux` **(Table 5)** is the flag for selecting a flux splitting scheme.
- ***Integration:*** `irst` **(Table 6)** is the solution strategy flag.
- `ivisc` **(Table 7)** is the flag for selecting viscous parameters.

`ivisc` is defined as a three digit integer (`IvJvKnv`, where one value is required for each global set of iterations (see `ngit`).

Table 4. Definitions for `ivfxj`

Display ***Flux approximation***	Definition	Pab3D `ivfxj`
van Leer (1)	van Leer	1
van Leer (2)	Modified van Leer	2
S&W	Steggar and Warming	3

Table 5. Definitions for `ivflux`

Display ***Jacobian***	Definition	PAB3D `ivflux`
van Leer (1)	van Leer (for inviscid simulations)	1
van Leer (2)	Modified van Leer	2
Roe	Roe (recommended for viscous simulation)	3

Table 6. Definitions for `irst`

Display Integration	Definition	PAB3D `irst`
PNS	Parabolized Navire-Stokes	1
Time Dependent	Time Dependent	2
SMS	modified Roe's scheme solution (Space Marching) (recommended for supersonic and small pressure gradient subsonic flow problems), AIAA Paper 89-2196.	3

- `Iv` is the viscous option the I-direction.
- `Jv` is the viscous option in the J-direction.
- `Kv` is the viscous option in the K-direction.

A viscous option of 1 is for thin layer, 2 is for coupled viscosity, and 0 is for no viscous simulation. Example 222 is full Navier-Stokes simulation and 111 is thin-layer on all three computational directions (I, J and K)

Diffusion Terms: `ivisc`

Viscous Model: `kturb` **(Table 8)** is the turbulence model flag

Most of the two-equation k-ε models are listed in Reference 1

- `ibs` is the `i` start (cell #) of that solution.
- `ibf` is the `i` last (cell #) of that solution.

The variables, `ibs` and `ibf` represents the restricted range in the I-direction.

3.2.2 Orders and Limiters in I, J, K Directions, ICOND

The syntax of this statement is

```
i-order i-lmt j-order j-lmt k-order
k-lmt icond
3 2 3 2 3 2 0
```

Table 7. Definitions for `ivisc`

Display *Diffusion Terms*	Definition	PAB3D `ivisc`
000	Inviscid	000
001	K-direction Thin-Layer diffusion terms	001
010	J-direction Thin-Layer diffusion terms	010
100	I-direction Thin-Layer diffusion terms	100
011	J and K directions Thin-Layer diffusion terms	011
101	K and I directions Thin-Layer diffusion terms	101
110	I and J directions Thin-Layer diffusion terms	110
111	I, J and K directions Thin-Layer diffusion terms	111
022	J and K directions Coupled diffusion terms	022
202	K and I directions Coupled diffusion terms	202
220	I and J directions Coupled diffusion terms	220
122	I thin with J and K coupled diffusion terms	122
212	J thin with I and K coupled diffusion terms	212
221	K thin with J and I coupled diffusion terms	221
222	Full Navier-Stokes diffusion terms	222

Table 8. Definitions for `kturb`

Display *Viscous Model*	Definition	PAB3D `kturb`
Laminar	Laminar Flow	-1
1-Equation	Spalart's one-equation turbulence model	5
2-Equation Standard	k-ε turbulence model, standard, (C3 = 1.44)	6
2-Equation Modified JL	k-ε turbulence model. Modified Jones and Launder (C3 is a function of production and dissipation)	7
2-Equation Speziale	k-ε turbulence model based on Speziale et.	8
2-Equation Wilcox	Wilcox *k*-ω model.	9
2-Equation RNG	k-ε turbulence model based on Renomlization Group theory.	10
2-Equation YS	k-ε turbulence model based on Yang/Shih model using an new ε-equation ICOMP-94-21	11
2-Equation JL	Jones and Launder (C3 = 1.45)	16

The following input line describes the orders and limiters applied to each block*:* `i-order.`

The spatial discretization order flag is defined as in **Table 9.**

Table 9. Definitions for `iord`

Display ***Spacial accuracy***	Definition	PAB3D `iord`
First order	First order	1
Second order	Second order	2
Third order	Third order	3
Central difference	Central difference	4

Spacial accuracy: `iord`

The flag for the limiters is defined in **Table 10.**

Table 10. Definitions for `i-`, `j-`, and `k-lmt`

Display ***Limiter***	Definition	PAB3D `i,j,k -lmt`
Unlimited	Unlimited	0
van Albeda	van Albeda limiter	1
Minmod	minmod limiter	2
Spekereijse-Venkate (1)	Spekereijse-Venkate limiter	3
Spekereijse-Venkate (2)	Modified Spekereijse-Venkate limiter	4

`Limiter: i-lmt, j-order, j-lmt, k-order, k-lmt`

The ***icond*** flag is defined as:

- 0 = no preconditioning (recommended)
- 1 = Model 1, Low-Speed Preconditioning
- 2 = Model 2, Low-Speed Preconditioning

These models are not fully tested yet.

3.2.3 Specification of Cuts in Faces 5, 6

In the PAB3D code, faces 5 and 6 are the block boundaries at `i = 1` and `i = idim`, respectively. Faces 5 and 6 are zonal interfaces in the space marching scheme. Therefore, their specifications are separated from those of faces 1, 2, 3, and 4. The specification is done in two steps. First, the number of cuts on faces 5 and 6 are given in a sub-header line containing two integers. Second, a group of lines, listed first for face 5 and then for face 6, specifies the boundary/connectivity condition code and the `(j, k)` range of each cut.

3.2.4 Number of Cuts for Faces 5 and 6

The syntax of this statement is

```
ncut-Imin ncut-Imax Imin & Imax Faces
1 1
```

The variables, `ncutpz` (5) and `ncutpz` (6), are defined as the number of cuts for faces 5 and 6. An example-input line is:

`1 1` (a single cut for face 5 and face 6)

3.2.5 Boundary Condition Specifications and Range

The syntax of this statement is

```
ibci j1 j2 k1 k2
-17 1 8 1 80
-17 1 8 1 80
```

The variables, `ibci`, `j1`, `j2`, `k1`, and `k2`, define the boundary conditions and range for faces 5 and 6. This is a group of repeated specifications, one for each cut in face 5 and then face 6 (total number of lines: `ncutpz` (5) + `ncutpz` (6)).

- `ibci` is the boundary condition flag

Adjacent Boundary: normally, the block number to which this face is connected. However, this can be a "10000" series code to prompt the "general patched interface". In this case, the "10000" code format is defined as:

<u>1</u>0000: the first digit is always "1". It is a flag for "general patched interface"

1<u>000</u>0: the second to fourth digits identify the target block; this number is right adjusted. E.g.,: 10020 is block 2 while 10200 is block 20.

1000<u>0</u>: the last digit is the face number for the target block that could have a value of 1, 2, 3, 4, 5, or 6. A value of 10000 is used to force extrapolation for the empty or undefined cells for any face.

Table 11. Definitions for **`ibcijk`**

Display ***Boundary Condition***	Definition	PAB3D **`ibcijk`**
Adjacent Face	This is the general patched interface, AIAA 95-2336.	10000+
Noslip Wall	Noslip adiabatic wall, NASA CR-182032.	0
Farfield (1)	Characteristic using Riemann Invarients.	-1
KI half-plane polar	Half-plane across a singular polar point with KI as the plane of symmetry	-2
IJ half-plane polar	Half-plane across a singular polar point with IJ as the plane of symmetry	-3
JK half-plane polar	Half-plane across a singular polar point with JK as the plane of symmetry	-4
Extrapolation	First order extrapolation for all the flow variables	-6
Constant total values (1)	This is a subsonic internal inflow (engine boundary condition) which fixes the total flow angle and total pressure and temperature. Automatically will switch to the Constant pressure boundary condition if the nozzle pressure is higher than the specified total pressure.	-11
Constant pressure (1)	This is usually used for subsonic exit boundary condition. However it will switch to Extrapolation condition for each cell if the flow is supersonic.	-14
Constant static values (1)	All the variables are fixed at their static conditions. This boundary is used to simulate supersonic inflow or low pressure gradient subsonic external flow.	-12
Constant Mach # (2)	The average Mach # is fixed at a constant value for which the pressure is adjusted to satisfy this boundary condition. It is usually used for subsonic inlet boundary condition.	-25
Periodic Boundary	It can be used for any periodic boundary plane regardless of the computational coordinates.	-16
General symmetry	This is based on the innovative formula provided by Paul and Abdol-Hamid (won NASA's Tech Brief of the year 1993). It can be used for any symmetry plane in regardless of the computational coordinates. It can also be used for slip wall boundary condition.	-17
Porous wall (2)	This bc uses either the bleeding boundary condition similar to the one in NPARC code.	-18
Porous wall temperature (2)	This is a porous flow and constant surface temperature in/out normal to the surface.	-19
Farfield (2)	Characteristic using Riemann Invarients.	-21
Constant total values (2)	This is a subsonic internal inflow (engine boundary condition) which fixes the total flow angle and total pressure and temperature. Automatically will switch to the constant pressure boundary condition if the nozzle pressure is higher than the specified total pressure.	-22
Constant static values (2)	All the variables are fixed at their static conditions. This boundary is used to simulate supersonic inflow or low pressure gradient subsonic external flow.	-23
Constant pressure (2)	This is usually used for subsonic exit boundary condition. However it will switch to Extrapolation condition for each cell if the flow is supersonic.	-24
Constant wall temperature (2)	This is to fix constant surface temperature and no-slip for wall boundary condition	-30
Spinning Wall	Sets velocity components on the spinning surface. Surface has to be surface of revolution. NB PAB3d does not allow for a moving grid	-32

Boundary Condition: **`ibcijk` (Table 11)**

1. It reads the **`initb.d`** file to set a non-uniform flow distribution at each grid point (see **`ginit`** section)
2. These boundary conditions are specified at the sub-face level (see **`surf.cont`** section)

3.2.6 Specification of Cuts for Faces 1, 2

In the PAB3D code, faces 1 and 2 are the block boundaries at `j = 1` and `j = jdim` respectively. Similar to faces 5 and 6, the specifications are given in two parts. First, the number of cuts for each of the faces, 1, 2 are given. Then, the detailed cut specifications are given consecutively in the order of faces 1 and 2.

3.2.7 Number of Cuts for Faces 1and 2

The variables, **`ncutb`** (1) and **`ncutb`** (2) are defined as the number of cuts for faces 1 and 2. The syntax of this statement is

```
ncut-Jmin ncut-Jmax Jmin & Jmax Faces
1 1
```

3.2.8 Boundary Condition Specifications and Range

The syntax of this statement is

```
ibcjk k1 k2 i1 i2
-11 1 80 1 1
2 1 80 1 1
```

The variables, **`ibcji`**, **`k1`**, **`k2`**, **`i1`**, and **`i2`**, define the boundary conditions and range for faces 1 and 2. This is a group of repeated specification, one for each cut in face 1 then face 2 (total number of lines: **`ncutb`** (1) + **`ncutb`** (2)).

This is actually a group of lines for all the face cuts. The boundary condition code, **`ibck`**, represents the same set of code numbers given for **`ibci`** of faces 5 and 6. For faces 1 and 2 the grid range specifications are given in the order of **`k1`**, **`k2`**, **`i1`**, **`i2`**.

- **`ibck`** is the boundary/connectivity condition code
- **`k1`** is the starting cell index in **`k`**
- **`k2`** is the ending cell index in **`k`**
- **`i1`** is the starting cell index in **`i`**
- **`i2`** s the ending cell index in **`i`**

In this example, face 1 uses -11 (engine BC) and face 2 communicate with B2. Users can also use in more general way ‘10021’ to assign B2 and F1 as the destination for B1 and F2.

3.2.9 Specification of Cuts for Faces 3, 4

In the PAB3D code, faces 3 and 4 are the block boundaries at `k = 1` and `k = jdim` respectively. Similar to faces 5 and 6, the specifications are given in two parts. First, the number of cuts for each of the faces, 3 and 4 are given. Then, the detailed cut specifications are given consecutively in the order of faces 1 and 2.

3.2.10 Number of Cuts for Faces 3 and 4

The variables' **`ncutb`** (3), and **`ncutb`** (4), are defined as the number of cuts for faces 3 and 4. The syntax of this statement is

```
ncut-Kmin ncut-Kmax Kmin & Kmax Faces
1 1
```

3.2.11 Boundary Condition Specifications and Range

The syntax of this statement is

```
ibcjk j1 j2 i1 i2
-17 1 8 1 1
0 1 8 1 1
```

The variables, **`ibcji`**, **`j1`**, **`j2`**, **`i1`**, and **`i2`**, define the boundary conditions and range for faces 1 and 2. This is a group of repeated specification, one for each cut in face 1 then face 2 (total number of lines: **`ncutb`** (3) + **`ncutb`** (4)).

This is actually a group of lines for all the face cuts. The boundary condition code, **`ibcj`**, represents the same set of code numbers given for **`ibci`** of faces 5 and 6. For faces 3 and 4 the grid range specifications are given in the order of **`j1`**, **`j2`**, **`i1`**, **`i2`**.

- **`ibcj`** is the boundary/connectivity condition code
- **`j1`** is the starting cell index in *j*
- **`j2`** is the ending cell index in *j*

- `i1` is the starting cell index in *i*
- `i2` is the ending cell index in *i*

3.3 Code Execution Flags, Including Scale, Timing, Trip, Etc.

This segment of instructions follows all the zone/block job control and boundary condition specifications.

3.3.1 Title Line

An example of the input line is

```
Real Gas Simulation for 2D Nozzle
```

The user gives a title to the problem that is being resolved.

3.3.2 Input Line for Flags

The syntax of this statement is

```
rj dt iflagts fmax isym
0.0254 -30.00 -4 50.00 2
```

The flags for this input line are: `rj`, `dt`, `iflagts`, `fmax` and `isym`. They are defined as:

- `rj` is the multiplication factor for *x, y* and *z* that converts grid units to meters.
- `dt` is the time step.
  - < 0 local time step CFL- number = `abs (dt)`
  - 0 constant time-step = `dt*dt` (smallest time-step)
- `iflagts` is the number of iterations between CFL-number adjustment.
  - 0 the code uses local values for calculating time step
  - < 0 the code uses stored time step
- `fmax` is the maximum adjustment factor of CFL-number: CFL `fmax*abs(dt)`.
- `isym` controls turbulence stress representation.
  - = 1 cell center turbulence stress representation
  - = 2 cell face turbulence stress representation required for Algebraic Reynolds Stress models

3.3.3 Input Line for Flags

The syntax of this statement is

```
igrid iriso inorm kg1 kg2 iperf1
jkswp impvis
11 -1 1 1 5 0 0 1
```

The flags for this input line are**:** `igrid`, `iriso`, `inorm`, `kg1`, `kg2`, `iperf1`, `jkswap` and `impvis`. They are defined as:

- `igrid` **(This option is not used for all the PAB3D >2.0).** This is the format flag for the grid input file. The PRE and PAB3D codes accept only the PLOT3D format `"igrid=11"`.
  - = `11` is the PLOT3D Multi-block Grid Format. It reads grid header that contains number of blocks and dimensions and then reads a single record of floating numbers containing x, y, z for all grid points.
- `Irisio` **(This option is not used for all the PAB3D > 2.0).** This is the PLOT3D output file flag. The PLOT3D out files have fixed names: `xiris.d` is the grid file*;* `qiris.d` is the flow variables file; and `keiris.d` is the turbulence quantities file
  - 0: PLOT3D grid file
  - 1: PLOT3D grid, flow variables, and turbulence quantities files for full grid
  - 2: PLOT3D flow variables, turbulence quantities files for full grid
  - –1 same as 1 but for reduced grid
  - –2 same as 2 but for reduced grid

- `inorm` **(This option is not used for all the PAB3D > 2.0).** This is the normalization flag for plot3d files' variables.
    - 0 output in physical units
    - 1 non-dimensional output
- `kg1` is the temperature correction flag.
    - 1 no temperature correction for turbulence model
    - 2 temperature correction for turbulence model (research not for production)
- `kg2` is the number of iterations the L-U is frozen in using the space marching scheme.
- `iperf1` (This option is not used for PAB3D versions released after 2.0) is the performance calculation flag.
    - 0 = no flow performance calculations
    - 1 = flow performance calculations sent to `perf.out`, `perf.track`, and `body.track` files
- `jkswap` is the alternate `j-k` metrics inversion (0 recommended).
    - 0; always `j` and then `k` inversion
    - 1; always `j` and `k` alternate
    - 2; always `k` and then `j` inversion
- `impvis` controls the viscous contributions to the implicit terms.
    - = 1; contributions of the viscous terms to the implicit are evaluated (recommended during the initial calculations)
    - = 0; contributions of the viscous terms to the implicit terms are not evaluated (could save 15 - 20% of the computational time)

This line will be phased out in the future release of PAB3D. No PLOT3D files will be output from PAB3D code and other options will not be available or will have a fixed value.

3.3.4 Input Line for Flags

The syntax of this statement is

```
ibc i2d itrp
0 1 1
```

The flags, `ibc`, `i2d` and `itrp`, are defined as:

- `ibc` is the Riemann's Invarient flag for far field boundary.
    - 0 = constant pressure method
    - 1 = constant entropy method
- *i2d* is the topology flag.
    - 0; three-dimensional flow
    - 1; two-dimensional plane flow in the I-direction

In the example problem given in the Appendix-A, a 2D option is utilize to cut the computational overhead by 20%. Additionally, we have used the tripping option to transition to fully turbulent flow (see `tran.cont` segment of the `user.cont` description).

- `itrp` is the turbulent tripping flag.
    - 0 no tripping
    - 1 trip file exists (`tran.cont`)

3.3.5 Input Line for Flags

The syntax of this statement is

```
ivortt istat sigl sigu gam itreg
1 0 0.0 0.28 1.4 1
```

The flags, `ivort`, `istat`, `sigl`, `sigm`, `gam` and `itreg`, are defined as:

- **`ivort`** is the diffusion coefficient in the k and ε equations.
    - 1 = uses turbulent viscosity $f_\mu C_\mu k^2/\varepsilon$
    - 2 = uses $C_\mu k^2/\varepsilon$ and C_μ=.09
- **`istat`** controls the method used to solve the multi-species governing equations.
    - 0 = uses the fully upwind scheme
    - 1 = uses a similar scheme to the one used to solve the governing equations
- **`sigl`** is the lower limit for the algebraic stress variable C_μ.
- **`sigm`** is the upper limit for the algebraic stress variable C_μ.
- **`gam`** is the fixed ratio of specific heats.
- **`itreg`** is the number of iterations to adjust γ.

Section 4.0 User Control File `user.cont`

The segments of `user.cont` are given in **Table 12.** These segments can be in any order within the `user.cont` file. They start with Begin and end with End lines.

4.1 Ke Cont Input

The syntax of this statement is

```
'Begin KE Cont'
ibk ilhg iord dtf itk icomp comp Int
ut/ul inl icu idmp
1 -14 1 4.0 2 0 1. 1.e-4 0.1 0 0 0
2 -14 1 4.0 2 0 1. 1.e-4 0.1 0 0 0
3 3 1 4.0 2 0 1. 1.e-4 0.1 0 0 0
4 3 1 4.0 2 0 1. 1.e-4 0.1 0 0 0
5 -14 1 4.0 2 0 1. 1.e-4 0.1 0 0 0
'End KE Cont'
```

This section describes the `ke.cont` segment of the `user.cont` file. This file controls the turbulence models flags and parameters. The `ke.cont` segment has a one-line input for each block where a turbulence model is specified. The following variables are contained in the data line: **`ib`**, **`ilhg`**, **`iord`**, **`dtf`**, **`itk`**, **`icomp`**, **`comp`**, **`Int`**, **`ut/ul`**, **`inl`**, **`icmu`**, **`ibn`**,

Table 12. Segments of `user.cont`

Default Name	**File Content**
`memo`	This is a user's segment for MEMO 'Begin Memo'
`ginit.cont`	input segment for `Gibc.f` program (unit=5, `Gibc<user.cont`) `'Begin Ginit Cont'`
`bc.cont`	input segment for `Gibc.f` program (unit=5, `Gibc<user.cont`) `'Begin Bc Cont'`
`tran.cont`	fixed name for turbulence trip location segment `'Begin Tran Cont'`
`surf.cont`	fixed name for special boundary conditions segment
`Track.cont`	Track Point/Probing segment
`perf.cont`	Performance Specification segment *(This option is not used for all the Pab3D versions released after V2.0)*
`ke.cont`	Turbulence model specification segment `'Begin Ke Cont'`
`mpi.cont`	Mpi setup for block load balance 'Begin MPI Cont'
`spec.cont`	Multispecies concentration and setup 'Begin Spec Cont'

Table 13. Definitions for `ilhg`

Display *Wall Integration*	Definition	Pab3D `ilhg`
LR (LS)	Low Reynolds # model of Launder and Sharma	-14
HR (Abid)	High Reynolds # model with Abid's damping function for dissipation equation	-11
LR (Yang)	Low Reynolds # model with Yang's damping function	-10
LR (YS)	Low Reynolds # model with Yang/Shih's damping function	-8
LR (Hamid)	Low Reynolds # model same as LR (f (η^+)) with all near wall model terms vanishing away from walls	-7
LR (NH)	Low Reynolds # model with Nagano-Hishida's modification	-6
LR (Specizal)	Low Reynolds # model with Specizal's damping function	-5
LR (JL)	Low Reynolds # model of Jones and Launder	-4
LR (Wall Function)	LR Reynolds # model with wall function	-3
LR (f (η^+)))	Low Reynolds # model with damping function (f (η^+))	-2
HR (Damping)	High Reynolds # model with damping function	1
HR	High Reynolds # model with no damping function	3
HR (Wilcox)	High Reynolds # model with Wilcox near wall model	4
HR (Wall Function)	High Reynolds # model with wall function	5

are defined as:

- `ib` is the block number
- `ilhg` is the two-equation turbulence model option flag.

Wall Integration : `ilhg` (see **Table 13**)

The most commonly used options for PAB3D are HR (LS) for internal and flow with solid wall and HR for external flow. Most of the Low Reynolds # are listed in AIAA Journal, Vol. 23, No. 9, pp. 1308-1319.

- `iorde` controls the accuracy of solving the turbulence model equations

Table 14. Definitions for `iorde`

Display *Spatial Accuracy*	Definition	Pab3D `iorde`
1	first order *k*-ε solution in I, J, and K directions	1
0	*k*-ε solution in I, J and K directions follow the governing equations	0

Spatial Accuracy: `iorde` (see **Table 14**)

- `dtf` is the *k*-ε CFL number reduction factor (1).
- `itk` is the flag for ε wall boundary condition for the two-equation models.

Wall Boundary : `itk` (see **Table 15**)

- μ is dynamic viscosity

Table 15. Definitions for `itk`

Display *Wall Boundary*	Definition	PAB3D `itk`
0	$\varepsilon = 0$	1
F(k,μ)	$2\varepsilon = \mu \frac{\partial \sqrt{k}}{\partial \eta}$	2

- `icomp` is the compressibility model.

Compressibility : `icomp` (see **Table 16**)

- `comp` is the compressibility correction factor used only with Sarkar's model.
- `Int`
    - For two-equation models, it is the lower limit for turbulence intensity $\sqrt{k}/C_0$, where C_0 is the speed of sound.
    - For one-equation mode, it is the initial value of the Spalart's variable ratio to laminar viscosity
    - .0001 for no-tripping
    - .01 for tripping (not implemented for the one-equation)
- `ut/ul` is the smallest turbulence to laminar viscosity ratio
    - 1.0 for tripping
    - 1.0-10. for no-tripping

Table 16. Definitions for `icomp`

Display *Compressibility*	Definition	Pab3D *icomp*
no compressibility	no compressibility correction factor for the turbulence transport equations	0
Sarker	Compressibility correction for the k-equation based on Sarker's model	1
Ziemman	Compressibility correction for the k-equation based on Ziemman's model	2
Ziemman (SF)	Compressibility correction for the k-equation based on Ziemman's model for shear flow (SF)	3
Ziemman (BL)	Compressibility correction for the k-equation based on Ziemman's model for boundary layer (BL)	4
Wilcox	Compressibility correction for the k-equation based on Wilcox's model	5

- `inl` controls the use of the Algebraic Reynolds Stress Models with the two-equation models.

Stress Model: `inl` (see **Table 17**)

- `icmu` controls the C_μ (Coefficient of Viscosity) functions
    - 0 = C_μ equals constant
    - 1 = C_μ as function of strain, k and ε based on the Shih and Lumley Model
    - 2 = C_μ as function of strain, k and ε based on the Abid et. al. Formula
- `ibn` is the blending function between high and low Reynolds Number Models
    - 0 = no blending function
    - 1 = F1 blending function
    - 2 = F2 blending function

Another example of `ke.cont` is given as follows:

```
ib ilhg iord dtf itk icomp comp Int
ut/ul ikn icmu ibn
3 -14 0 4. 2 5 1. 1.e-4 0.1 7 0 0
4 3 0 4. 2 5 1. 1.e-4 0.1 7 0 0
```

This example shows that k-ε turbulence model is used in blocks 3 through 9 in a nine-block grid. Default values are used for `iord`, `dtf`, `icomp`, `Int ut/ul`, `icmu`, and `ibn`. The `ilhg` values are chosen such that:

- `ilhg` sets to -14 if there is a wall
- `ilhg` sets to 3 for shear layer flows (jet or wake)

This example also shows that the Girimaji ARSM is used by choosing `ikn` = 7. The value of `dtf` = 4. is good for initial iterations at coarser grid levels in some cases. For iterations in medium or fine grid, the value of `dtf` should be lowered to `dtf` = 1, if possible.

4.2 Spec Cont Input

The syntax of this statement is

```
'Begin Spec Cont'
1.0 3 'CO2''N2''Air' 1, 10,"temp.dat"
0.1 0.9 0.
0.1 0.9 0.
0.0 0.0 1.
'End Spec Cont'
```

This segment of the `user.cont` describes the multi-species used in the simulation. The first number is the Schmidt's Number. This is followed by the number of species (3), their names and the real gas model (1) in use. CO2, N2 and Air are the gases in use for the present simulation. The next three lines represent the species mass concentration. Each of the lines is equivalent the lines in the `ginit.cont` set.

Then imodel number, which describe the real gas models 0-4. See **Appendix A** for the description of

Table 17. Definitions for `inl`

Display ***Stress Model***	Definition	Pab3D `inl`
Linear	Basic stress model with linear relation to strain	0
SZL (1)	Nonlinear stress model based on modified Shih, Zhu and Lumley (SZL) AIAA 95-2246	1
SZL (2)	Nonlinear stress model based on SZL Stress Model NASA TM-106644, 1994	2
SSG (1)	Nonlinear stress model based on Speziale, Sarkar and Gatski (SSG) stress model with two-dimensional assumption	3
SSG (2)	Nonlinear stress model based on Speziale, Sarkar and Gatski (SSG) stress model with $\Pi_\delta = 0$	6
LRR	SSG with Launder, Reese and Rodi (LRR) coefficient for the stress terms	4
GL	SSG with Launder, Gibson and Launder (GL) coefficient for the stress terms	5
Girimaji	Nonlinear stress model based on Girimaji stress model with two-dimensional assumption	

the real gas models used in PAB3D code. The last number controls the chemical reaction model used by PAB3D (Ichem):

Ichem = 0 no chemical reaction

Ichem = 1 one-step chemical reaction model for H2O

Ichem = 2 explicit chemical reaction model

Ichem = 3 implicit chemical reaction model

If Ichem>0, PAB3D expect to read a file contains number of reactions, species and chemical properties.

4.3 Ginit Cont/Bc Cont input

The `Ginit` and `Bc` sections are identical in format. User can use Bc to specify boundary condition, which is different from the initial condition. The syntax of this statement is

```
'Begin Ginit Cont'
iunit nblock ireg
0 5 2
ncut jmin jmax kmin kmax iset
1
1 9 1 81 1
1
1 113 1 81 2
1
1 121 1 81 2
1
1 121 1 41 3
1
1 9 1 41 3
nset iinit
3 2
pres temp mach Int ut/ul alpha beta
gamma iin
85.000 1400. 0.5 0 0 0 0 1.4 0
pres temp mach Int ut/ul alpha beta
gamma iin
14.72 530 0.2 0 0 0 0 1.4 -1
pres temp mach alpha Int ut/ul gamma
beta iin
14.72 530 0.01 0 0 0 0 1.4 -1
'End Ginit Cont'
```

Fig. 5 shows the PAB3D Physical Domain. In this segment of `user.cont`, the user can specify initial/boundary condition for different sub-faces of a certain block. The simple form of this file is automatically generated with the AutoG3d.f utility.

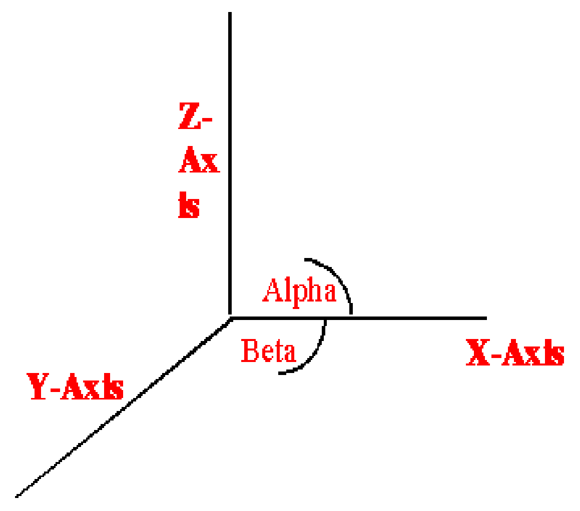


Figure 5. PAB3D Physical Domain

The line, `iunit, nblock`, and `ireg`, are defined as:

Units: `iunit` (see **Table 18**)

- `nblock` is the number of computational domain blocks .

Table 19. Definitions for `ireg`

Display ***Gas Model***	Definition	PAB3D `ireg`
Ideal	Ideal gas model with constant gamma for one gas	0
Real (Frozen)	Multispecies real gas simulation	2

- `ireg` is the Real Gas Model flag.

Gas Model: `ireg` (see **Table 19**)

The line, `ncut`, is the number of sub-faces describing the boundary condition and is repeated `nblock` times. `jmin`, `jmax`, `kmin`, `kmax`, and `iset` are re-

Table 18. Definitions for `iunit`

Display ***Units***	Definition	PAB3D `iunit`
English	English units, lb, inch, ...etc.	0
SI	International system units, kg, meter, …etc	1

peated **`ncut`** times where:

- **`jmin`** is the lower left corner *j* location.
- **`jmax`** is the upper right corner *j* location.
- **`kmin`** is the lower left corner *k* location.
- **`kmax`** is the upper right corner *k* location.
- **`iset`** is the set number of the boundary condition

The line, **`nset`**,and **`iinit`**, is defined as:

- **`nset`** is the number of sets.
- **`iinit`** is the set number used for normalization (reference condition).

The line, **`pr`**, **`temp`**, **`mach`**, **`muT/muL`**, **`alpha`**, **`beta`**, **`gam`**, **`Int`** and **`iin`**, are defined as:

- **`pr`** is the first variable where:
    - o = Reynolds Number/unit (meter or inch), if **`iin`** = -2
    - o = static pressure, if **`iin`** = -1
    - o = total pressure, if **`iin`** = 0
- **`temp`** is the second variable where:
    - o = total temperature , if **`iin`** = -2 or 0
    - o = static temperature, if **`iin`** = -1
- **`mach`** is the Mach number.
- **`Int`**
    - o For two-equation models is the lower limit for turbulence intensity $\sqrt{k}/C_0$, where C_0 is the speed of sound.
    - o For one-equation mode is the initial value of the Spalart's variable ratio to laminar viscosity
        - o .0001 for no-tripping
        - o .01 for tripping (not implemented for the one-equation)
- **`ut/ul`** is the smallest turbulence to laminar viscosity ratio
    - o 1.0 for tripping
    - o 1.0-10. for no-tripping
- **`alpha`** and **`beta`** are the flow angles .
- **`gam`** is the thermodynamic constant γ.
- **`iin`** controls pr and temp (see above).

4.4 Surf Cont input

This segment of the **`user.cont`** file complements the **`ginit.d`** file in prescribing initial/boundary condition. It has a one line input for each sub-face of a block where the boundary condition flag is less than -18. PAB3D uses a new format for Surf section as compared with all the previous PAB3D V<3.0. The following variables are contained in the data line: **`iblk`**, **`iface`**, **`icut`**, **iunit, n_x, n_y, n_z, Mach, P_s, T_s, amp, freq, phase and ifun**.

- **`iblk`** is the block number.
- **`iface`** is the face number (1-6).
- **`Iunit`** (see **Table 18**)
- **`n_x, n_y, n_z`** is the unit velocity normal
- **`Mach`** `is the Mach number.`
- **`P_s`** is static
- **`T_s`** is static temperature.
- **`amp`** `is the amplitude 0 to 1`
- **`freq`** `is the frequency` ω `[Hz]`
- **`phase`** `is the phase shift,` ϕ `[deg]`

- **`ifun`**

  0 Built in harmonic function

  1 User function u_pressure_1

  2 User function u_pressure_2

  2 User function u_pressure_3

  2 User function u_pressure_4

A set of generalized harmonic, jet boundary conditions for total pressure and velocity is provided. The boundary conditions are implemented as generalization for total and constant values boundary conditions, =-22 and 023 respectively. The form of the built-in oscillatory jet boundary condition for total pressure is

$pt= pt0\ [1 + A \cos(2\pi\omega t + \phi)$

Where pt0 is the steady state value.

When using the -18 boundary condition flag, these are the definitions:

- **`n_x`** is the area ratio (porosity).
- **`n_y`** is the platinum pressure ratio to free stream.
- **`n_z`** is the bleeding hole angle (20 or 90).
- **`p_s`**
  - = 1: use bleeding BC similar to the one in NPARK code
  - = 2: use Darcy's Law for porous wall
- **`T_s`** is the location of BL to set total pressure and temperature and Mach number.

When using **-32** boundary condition (Spinning Surface Boundary Condition) flag, these are the definitions:

- **`n_x, n_y, n_z`** is unit vector axis of rotation
- **`Mach`** is the Mach number. *"Currently Not Used. For Future use"*
- **X0,YO,Z0** are Cartesian coordinates for a point on the axis of rotation.
- **idr** *"Currently Not Used. For Future use"* is the computational direction of the axis of rotation where idr=1,2 & 3 defines the axis of rotation to be j, k, or I direction respectively. I.e.

  idr = 1 means axis of rotation is in the J direction.
  idr= 2 means axis of rotation is in the K direction.
  idr = 3 means axis of rotation is in the J direction.
- **`omega`** is the rotational speed of axis of rotation rad/sec]
- **`ifun`** *"Currently Not Used. For Future use"*

4.5 Tran Cont

The syntax of this statement is

```
'Begin Tran Cont'
Number of Blocks with Trip Points &
Trip K- Int
2 5.0000001E-02
Block Number & Number of I-Planes
1 1
Number of Points for Plane #1
1
Location J K I (for the finest grid)
8 0 1
Block Number & Number of I-Planes
5 1
Number of Points for Plane #1
1
Location J K I (for the finest grid)
8 0 1
'End Tran Cont'
```

This segment of the `user.cont` file controls the trip intensity and the three-dimensional location for a certain block. The input line, **`nbltr`** and **`trpv`**, are defined as:

- **`nbltr`** is the number of blocks with trip turbulence intensity.

- `trpv` is the turbulence intensity at the trip locations (.01-.05).

The input line, `blockn` and `plann` (this line is repeated `nbltr` times), are defined as:

- `blockn` is the block number.
- `plann` is the number of *i* planes with trip.

The input line, `npplan`, is the number of trip points in this plane. This line is repeated `plann` times.

The input line, `j`, `k`, and `i`, is the three-dimensional trip location for the finest grid in use. This line is repeated `nnplan` times. For surface location, either use 0 or the maximum value in this direction. If `j` and `k` are 0, the code will trip the full `j-k` plane at the `i`-location specified in the previous line.

4.6 TimeStep Cont

(See Time-Accurate article: unsetady01.pdf) .The interface to the time accurate features are given as follows:

'Begin TimeStep Cont'

ir0 ratr ita cfltau ncycle

1 0.1 -2 5. 5

ir0 l2 norm residual calculations

ir0 = 0 use Q over cells

ir0 = 1 use r over cells

ratr is limit the rate of change of all the variables

ita positive for physical time sub-iteration and negative for dual sub-iteration

cfltau is CFL number for dual time sub-iteration

ncycle is the number of time retardations per physical time level

Track_Point Input

Probing or Point tracking is the ability to select points in the computation domain and track selected variables in time. Output is written to file trackpoint.dat located in restart.d directory. The interface to the point time tracking/probing features are given as follows:

'Begin Track_Point.Cont'

```
npts,incrmnt
 2      1
iblk  i    j   k    i_var
 2    20   60  45     1
 4    25   60  45     5
```

The first line defines the number of points to be tracked and the incremnet to write to the data file.

npts , incrmnt

```
 2      1
```

npts is **npts** is the number of points to be tracked.. 2 in this case means we are tracking 2 points in time.

incrmnt is the increment to write to the data file. 1 in this case means we are storing data at every time step

The line **iblk, i , j, k, i_var** is repeated npts times. One line for each point to be tracked. Each point correponds to a variable at a particular point in the computaion domain. Each point is identified by its block number **iblk** and the **i,j,k** location of the point to be tracked. i_var is the variable to be tracked. Refer to table 20 for i_var definitions. Each property to be tracked will have a separate line.

To track rho and p for the same points

```
iblk i   j   k    i_var
 2   1   60  45     1
 2   1   60  45     2
```

4.7 Flow Axis Cont

The syntax of this statement is:

```
'Begin FlowAxis Cont'
 idr  amx  amy  amz  direction of
J=1, K=2, I=3
 1    1    0    0
'End FlowAxis Cont'
```

This segment of the user.cont files complements ginit.d file in prescribing axis of rotation for periodic boundary condition bc = -16. The following variables are contained in the data line: **idr amx** **`amy`** **`amz`**

idr is the computational direction of axis of rotation where idr=1,2 & 3 defines the axis of rotation to be j, k, or I direction respectively.

amx, **amy**, and **amz** are the direction cosines of axis of rotation for periodic boundary condition.

Section 5.0 Solution Procedure

To start the solution procedure, the user needs to prepare two control files as described in the above sections. We created a utility program AutoG3d.f, which is capable of generating the two essential control files for using the PAB3D code (`user.cont` and the solver control file `'tpab3d.cont'`). This utility program reads the grid in the PLOT3D format and generates at least 90% of the contents of this control file. The user needs to edit these files for boundary condition flags and initial and boundary condition values in the `ginit.cont` segment of the `user.cont` file.

If you have the source code, you need to edit the `"Make.defs"` make file for the corresponding computer architecture in use. The Makefile will create a `/bin` directory in the user home and set all the executable in this directory. User needs to add to the `"set PATH"` command in the `.cshrc` file the path to the `/bin` directory. Then, the user will generate four executable programs, which are completely independent of the problem size:

1. make AutoG3d
2. make Gibc
3. make Pre2
4. make Post
5. make Pab2

The following are the procedures for producing a solution with the PAB3D code:

1. AutoG3d is an interactive program, which will generate the following files:
   - `autog3d.cont`, the solver control file and `userx.cont`, you can use it to setup the `user.cont` file.
2. Edit and fix these control files with your problem specifications.
3. Run the Preprocessing program, `Pre2`.
   - `Pre2 autog3d.cont`, will generate `tpab3d.cont` file and modify the input grid file (now it is the geometry file). The geometry file, which contains the grid information, data base information and memory parameters.
4. Run `Pab2` and get a converge solution. Input file is `tpab3d.cont` and the screen output file is `PAB.out.`
   - Save you restart file for latter recovery.
   - `Pab2`, and so on until the solution converges through all the possible mesh sequencing steps

5.1 MPI Procedure

Simulation of complex aerodynamic problems requires very large grid (order of millions). Basically, we will need large memory and fast computers to simulate such problems. Another alternative is to distribute the problem into smaller and slower computers. The speed of workstation is approaching super computer speed with a lot of memory. This is a continuation of the work started by Dr. Fabio during the Summer of 1997 at the ICASE. This report addresses the implementation of MPI in the PAB3D code. This allows the use of distributed computing capability for workstation clusters (multiple platforms) using the MPI system. Code modification is done using directive command lines. This allows us to keep only a single version of the PAB3D code for different platforms for single and distributed capabilities. Appendix D describes briefly the MPI implementation and benefits to CFD.

User needs to setup four files to use the LAM/MPI package as follows:

1. Change in the `.cshrc` file adding. For example

```
Source $HOME/Lam.rc
```

2. Add the `Lam.rc` in users home in each of the processors will be used. For example

```
set lam = ~hamid/lam
set path = (path $HOME/$lam/bin)
setenv LAMHOME $lam
setenv LAMHF77 /usr/bin/f77
```

3. Edit a **Lam.host** file that assignees the computers as nodes. For example

```
#This is a 4-node file
cib2.larc.nasa.gov hamid
aero100.larc.nasa.gov hamid
aero4.larc.nasa.gov hamid
cable.larc.nasa.gov hamid
```

   Each computer need to be added to the user's **.rhosts** file

User needs to know a minimum of 5 MPI commands:

- lamboot: Boot and configure the required nodes as

```
lamboot -v ~/Lam.host
```

- wipe: Wipe the configurations as

```
wipe -v ~/Lam.host
```

- mpirun: Run the code in MPI mode as

```
ln -s /home/hamid/bin/Pab2 Pab2
mpirun -c 4 -v Pab4
```

  The number of the processors that will be used in this MPI run is 4. The `-s h` is only used if all the nodes can use the same executable otherwise remove the `-s h`. If you remove the `-s h` the executable should be located in the any of the directory of the user's path (see your `.cshrc` file)

- mpitask: To monitor the tasks in any of the processors

- lamclean: To kill the running MPI job.

User needs to setup two files to use the MPICH/MPI package as follows

1. Change in the `.cshrc` file adding Source `$HOME/Ch.rc`. Add the `Ch.rc` in users home in each of the processors will be used `Set path = (path $HOME/mpich/bin)`

2. Edit a **Pab.pg** file that assignees the computers as nodes

```
#This is a 4-node file
cib2    0
aero100 1 /home/hamid/bin/Pab2
aero4 1 /home1/hamid/bin/Pab2
cable 1 /home/hamid/bin/Pab2
ln -s /home/hamid/bin/Pab2 Pab2
mpirun –p4pg Pab.pg Pab2
```

The details of MPI implementation are shown in **Appendix B**. This appendix provides details of cluster performance and test cases.

Appendix C describes the procedure for solving an example problem.