pV3 Batch User's Reference Manual

Rev. 2.05

for use with the following workstations:

Compaq ALPHAs
HP workstations (HPUX 10.20 ACE)
RedHat LINUX (Rev 6.1 or higher)
IBM RS/6000 workstations (AIX 4.3 or higher)
Silicon Graphics
SUNs (Solaris 2.6 or higher)
WindowsNT/2000

Bob Haimes
Massachusetts Institute of Technology

December 12, 2001

Sections marked with this change-bar are have had a major change from last release (Rev 2.00) and will require some programming modifications.

License

Contents

1 Introduction

It has been found that performing certain visualization tools, like streaklines (unsteady particle paths), in a post-processing manner can lead to incorrect results. The problem with post-processing is that the time-step selected for the visualization is based on the available disk space and not the physical problem. The **pV3** system does not exhibit this problem because co-processing is fully supported.

The problem with **pV3** in a production environment or for batch execution is that the user may not be around to fire-up the interactive server and view the results. This manual describes the use of a *Batch* server and post-processing viewer for the **pV3** suite. The client side remains basically unchanged. The CFD solver need not know if the results are currently being viewed or to be viewed at some later time.

Therefore, when a batch job starts, the *batch* **pV3** server should also be started. Data is read on where and what tools and probes are to be active and their locations. The results (tool extracts) are collected and written to disk for play-back later. This is different from the normal post-processing in that the entire volume of data is not written to disk every iteration. The file specification used to write the data can be found in this sub-directory (the name is 'FileSpec.txt').

The end result is something that is not interactive in the placement of tools, but can be thought of as analogous to a wind-tunnel experiment. You have to be smart in where you place probes to extract data of interest. If you miss an important area (or only find it after viewing these results) you will have to re-run the tunnel adding (or changing the location) of the probes. The post-processing viewer is highly interactive in dealing with time. This is due to the fact that the amount of data has been reduced by orders-of-magnitude.

The batch server and post-processing viewer can be thought of as the **pV3** interactive server split into two portions. These parts communicate through the data file.

2 pV3 Batch Server

It is advisable to start the **pV3** batch server either before or concurrently with the solver. The **PVM** daemon(s) must be executing. This scheme insures that you will get a proper initialiation and not possibly miss clients (as can happen with the interactive server – if started while the solver is calling **pV_Update**). If the solver has a long start-up time, adjust the Time-Out constant appropriatly.

The purpose of this software is to collect and write the visualization data to disk. Therefore some knowlegde of the underlying hardware must be used. The machine running the **pV3** batch server should be the computer that contains the file-system where the file is written. You do not want to further burden the network by using nfs or afs (the network file systems).

2.1 Batch Server Startup

The batch server, *pV3Batch*, takes three arguments at the command line (all optional). The first is a flag used to indicate how full the extracts should be collected. The second argument is the Time-Out constant. The third argument is the *extract start-up* file. An argument of '−' is a place holder, allowing the default to be used.

examples:
```
% pV3Batch ALL
% pV3Batch STATE 120
% pV3Batch
% pV3Batch − − case.startup
```

Output always goes to the *extract* file named 'pV3.viz' in the default directory (where execution began). Additional files may be generated when each reaches the 2 Gigabyte limit. These files are named 'pV3*ijk*.viz', where *ijk* starts at 001 and counts up.

- Extract Flag

  This argument declares how completely the extracts are collected and written to disk. The options are MINIMAL (the default) which is how the normal server collects the data. FULL indicates the complete extract ignoring the specified mask for the object, unless the mask is zero. STATE indicates that the state-vector should be sent instead of any scalar or vector field data. ALL is the equivalent of FULL and STATE giving the most flexibility at viewing (at the cost of more communication and larger files). Any argument not understood reverts to MINIMAL.

  The advantage of passing and storing state-vector information is that the scalar and vector data can be derived. This allows the use of more compact data and gives flexibility at viewing. The scalar and vector fields can be constructed by the viewer allowing the changing of these quantities during the post-processing. The disadvantage is that both the client-side and the **pV3** viewer must be modified to understand how to deal with the state-vector. See the section on Using State-Vectors.

- Time-Out

  This value is the same as the environment variable 'pV3_TO' (and overrides it's value). If the argument is set, it must be an integer string which is the number of seconds to use for the Time-Out constant (the batch server's default is 60). This may be required if the time between solution updates is long. See the section on *Time-Outs and Error Recovery* in the Server User's Reference Manual.

- Start-up file

  The startup file is an ASCII list of static *extracts* that are desired at batch server initialization. This file can be generated by the **pV3** server at any time during the visualization session by hitting 'W' in the Key window. Objects will take 2 iterations to show up in the *extracts* output file. More complete post-processing viewing is possible by specifying FULL or ALL for the Extract flag. The scalar, vector and threshold fields may also be specified via this file.

  NOTES:
  1) For Programmed extracts, the Extract flag or start-up file have no effect.
  2) The format for this type of file has changed for Rev 1.20 in order to support multi-disciplinary visualization. Older files are still valid at startup for single discipline cases.

2.2 Environment Variables

The batch server also looks at these three Unix environment variables:

'pV3_TO' can also be used to change the internal Time-Out constant. If the variable is set, it must be an integer string which is the number of seconds to use for the Time-Out constant.

'pV3_Group' is usefull for differentiating multiple **PVM** jobs running under the same user ID. If this variable is set for the solver (client-side) before execution, it overrides the default client side group name *pV3Client*. The name used is the string assigned to this variable with *Client* appended. By setting this variable before Batch Server execution, it will set the server group to the variable's string with *Server* appended instead of using *pV3Server*. Only clients with the appropraite matching group name will be connected to this session.

'pV3_Threading' is used to specify how handshaking is handled between the active threads. There are two methods; (1) 'Hard' where the thread sits in a hard loop (with a thread yeild) looking for a change of state, or (2) 'Flag' where the threads use Semaphores for waiting until the state changes. The advantage of 'Hard' is interactivity, the advantage for 'Flag' is less processor time consumed. By default, this variable is set to 'Hard' for most situations with single processor workstations and 'Flag' for multi-processors.
Exceptions: ALPHA's default is 'Hard', SUN's default is 'Flag', reguardless of number of processors.

'pV3_FileSize' is used to specify a viz file limit smaller than 2 gigabytes. The value must the the number of bytes. This option is useful if you wish to store these files for archival purposes. One can specify 650 Megabytes to go to CDs. WARNING: a value too small (smaller than the header information) will cause problems.

2.3 Batch Server Shutdown

The batch server will shutdown when when one of the following conditions occur:

- A client exits
- The time-out condition is reached
- The batch server is killed – this is not recommended!
- The file 'pV3Batch.stop' is found in the directory that the batch server is running from (i.e., where it writes the file 'pV3.viz').

2.4 Advanced Programming

Doing **pV3** advanced programming with the batch server requires slightly different control. This is because there is no interactive user controlling the session and no **X** event loop.

Note the following:

- Module in the Server Suite

  You can figure-out which server application is running via a call to **pV_GetState** with $OPT = 0$. This returns whether *pV3Server*, *pV3Batch* or *pV3Viewer* is running. One advanced programming set of source can be generated to be used with all modules.

- **pVEvents**

  Because there is no **X** event loop, there are no calls to **pVEvents**.

- **pVSafe**

  Use the call to **pVSafe** to control the session. A single call is made before the data buffers are flipped and the next set of requests are made to the client(s). That is, once per iteration.

3 pV3 Post-Processing Viewer

The **pV3** post-processing viewer can be run at anytime from a supported graphics workstation and does not require **PVM**. The extracts file is parsed and read to collect the data.

3.1 Viewer Startup

The batch post-processing viewer, *pV3Viewer*, takes three (or more) arguments at the command line (all optional). The first is the *setup* file with the default of 'pV3.setup', see Special Files. The second argument is the *color* file to be used at startup (the default is 'spec.col'). The third and subsequent arguments are the *extract* file with the default of 'pV3.viz'. An argument of '−' is a place holder, allowing the default to be used.

examples:
```
% pV3Viewer case.setup
% pV3Viewer pV3.setup bw.col
% pV3Viewer
% pV3Viewer case.setup − case.viz
% pV3Viewer case.setup − pV3.viz pV3001.viz pV3002.viz
```

When using multiple *extract* files the information within each is checked for consistancy. The number of disciplines and the field variables for each discipline must match. This option is useful for viewing the results of multiple runs on the same data or when *pV3Batch* has generated multiple files (due to the size of the data streams).

3.2 Environment Variables

The **pV3** viewer uses three Unix environment variables. Some are the same as the ones used for the server and **Visual3**. The variable 'Visual3_CP' defines the file path to be searched for color files, if they are not in the user's current directory. This allows all of the color files to be kept in one system directory.

The second variable, 'Visual_KB' is optional. This variable, if defined, must point to a file that contains alternate keyboard bindings for the special keys used by **pV3**. The file is ACSII. The first column is the key name (10 characters) and the second is the X-keysym value in decimal (use 'xev' to determine the appropriate values for the key strokes).

The third is 'pV3_Threading' and functions as described for *pV3Batch*.

3.3 Special Files

- Window Manager Resource File

  By default, all modern Xwindows Managers allow the closing or deleting of windows by either double-clicking on the menu pull-down or selecting ‘Close’ or ‘Quit’ from the pull-down. This will abort the execution of the Server or Viewer. To avoid this, an additional window manager menu description can be added to the default information for the Window Manager. This is accomplished by specifying a user-level resource. In the distribution the following files can be found in the “servers” subdirectory; ‘user.4Dwmrc’, ‘user.mwmrc’, and ‘user.dtwmrc’. These files are for SGI’s default window manager, the Motif window manager and CDE’s window manager, respectively. The entire resoure file (for the appropriate window manager) must be copied from the system (usually something like ‘/usr/lib/X11/system.XXwmrc’ or ‘/etc/dt/config/sys.dtwmrc’) – if not already done. This file gets named ‘.XXwmrc’ (where XX is 4D or m) and then the appropriate user resource file appended to the end. Note: for CDE, this file gets put in the ‘.dt’ directory at the users top level and must be given the name ‘dtwmrc’.

- .Xdefaults

  If the SGI window manager is used, ‘4DWm’ must be told what to do with **pV3**s windows. These commands must be placed in the file ‘.Xdefaults’. See the file ‘user.Xdefaults’. For DEC, IBM and SUN systems this information is found in the window manager’s Setup File (Mwm or the appropriate file for the WM used).

  The **pV3** viewer requires three X fonts. The file ‘.Xdefaults’ in the users home directory is examined for the font names and are designated “Visual*large”, “Visual*medium” and “Visual*small”. The sample file ‘user.Xdefaults’ comes with the distribution and may be concatinated to the user’s ‘.Xdefaults’ file.

  The X fonts loaded on any system may be examined by the command ‘xlsfonts’.

- ‘Mwm’ Setup File

  When a user begins a session using the Motif window manager, it reads the initialization file ‘Mwm’ in the user’s home directory. This file contains information on how to treat various classes of windows and the window focus (as well as other things). A sample file ‘user.Mwm’ is in the distribution and may be used as ‘Mwm’ or its contents concatenated to the user’s ‘Mwm’ file.

- ‘twm’ Setup File

  If the ‘twm’ window manager is used, when a user begins a session, twm reads an initialization file ‘.twmrc’ in the user’s home directory. This file defines certain key bindings and window attributes. It is necessary for correct **pV3** operation for the user to modify the standard ‘.twmrc’ file as shown in the ‘user.twmrc’ file on the **pV3** distibution, or just use this file as ‘.twmrc’.

- Setup File

  When the **pV3** viewer starts, it looks in the user's current directory for the setup file specified as the first argument on the command line. This is an ASCII file which contains a number of useful defaults that the user may want to set to different values than **pV3**'s initial defaults. It also contains a set of viewing positions and cutting plane positions. Normally this file is generated by **pV3** when the user wants to store certain favorite parameters and viewing positions so that they can be used again on another data set, which is particularly useful when the user wishes to directly compare two different data sets with the same computational grid or geometry. However, an experienced user can also generate this file from scratch. NOTE: This file is NOT compatible with **Visual3**'s setup file.

- Color Files

  A number of different color files are supplied on the distribution. For those who wish to define their own color files, the format of these ASCII files is as follows:

$$\begin{array}{l}
\begin{array}{ccc} nc & nb & \end{array} \\
\left.\begin{array}{ccc} r & g & b \\ & . & \\ & . & \\ & . & \\ & . & \end{array}\right\} nc \\
\left.\begin{array}{ccc} r & g & b \\ & . & \\ & . & \\ & . & \end{array}\right\} nb
\end{array}$$

  where nc is the number of colors, nb is the number of background colors (0–4), and r, g, b are red,green,blue intensity values (0.0–1.0). The four background colors are for window background; grid color; tuft/streamline/ribbon color; contour line color. The default values which are used if $nb = 0$ are black; white; white; white. If $nb \neq 0$ then the specified colors over-ride the defaults for the first nb colors.

  When the **pV3** viewer searches for named color files, it first looks in the current directory, and then follows the color file path specified by the environment variable 'Visual3_CP'.

- Lock File

  If the server is running on a multi-processor SGI workstation a file is used for the coordination of the 2 threads generated during execution. This file has the name '.pV3.locks' and is open in the current directory. It should be noted that running two invokations of the **pV3** server or viewer from the same directory will NOT work. Both will use the same file for the lock and semaphore arena!

3.3.1 Notes on the Common Desktop Environment – CDE

For the **pV3** suite to work properly with CDE, the Style Manager must be used to change the default methods CDE uses for cursor/window functioning. Under the Window section "Point in Window to Make Active" must be selected. Also, "Raise Window when Active", and "Allow Primary Windows on Top" must NOT be selected.

3.4 User Interface

The user interface for **pV3**'s Viewer is roughly the same as the interface for the server as documented in the Server User's Reference Manual. The differences are listed below:

- Traffic Light

  There is only one *traffic light* in the Key window. This light indicates only local activity – there are no requests that are being waited upon.

- Simulation Time

  The entire time range in known. The time key (Key Window) reflects this range by having it displayed as the minimum and maximum of the scale. A specific time may be selected by clicking on the time key with the right mouse button (like specifying a scalar value for an iso-surface via the key).

  When the last simulation time frame has been displayed, the process recycles. The limits in the Key Window (when viewing Time) specifies the time range for this cycling. This can be used to clip off the first 2 frames that do not contain the start-up *extracts*.

  Note: the Viewer currently assumes that time monotonically increases for new disk frames.

- User Requests that Cannot be Satisfied

  Any request for the change of an objects attribute, specifying a dynamic surface, or any state change that cannot be satisfied is ignored. If the data is available the request will be granted.

- Remapping Surfaces

  Planar cuts can always be remapped to the 2D window by selecting the surface in the Surface Data Base (Dial Window) and hitting F2 in the 3D Window.

  If 'pV3.viz' was written using either the FULL or ALL attributes, then Domain and Programmed Cut Surfaces can also be remapped. In these cases, again select the desired surface and hit F4 (in the 3D window) if it is a Programmed Cut Surface or F5 (in the 3D Window) if the surface is a Mapped Domain Surface.

- Deleteing Streamlines and Surfaces

  You cannot delete either Streamlines or Surface Data Base entries. If viewing is not desired, just turn rendering off.

- Scalar and Thresholding Flipping

  If FULL was specified for data collection and both the scalar and threshold fields are actually scalars (the threshold does not have an FKEYS = 5) then the scalar and threshold definitions can be swapped. This is accomplished by hitting F9 in the 3D window.

3.5 Advanced Programming

For the most part, advanced programming within the post-processing viewer is 100 percent compatible with the interactive server.

Note on Programmed Extracts: Calls to **pV_Register** must be done in the same order (for multiple instances of extracts with the same index) as was done for the batch server. For simplicity, the same code for **pVSafe** can be used for both *pV3Batch* and *pV3Viewer* and all extracts should be registered within the first call.

3.6 File-Spec Compliance

It must be noted that *pV3Viewer* does not fully deal with all files written that follow the file specification documented in 'FileSpec.txt'. It deals with the subset that *pV3Batch* writes.

3.7 Warning and Error Messages

The following warning and error messages are unique to the viewer:

- Warning: Premature EOF!

  The end of the file was reached without a proper close. *pV3Batch* was probably terminated without the client(s) exiting.

- ERROR - FileSpec: S*FILE** rev: *x.xx*

  The rev for the FileSpec is not supported.

- ERROR - too many disciplines: *num*

  The number of disiplines *num* is greater than the viewer can support.

- ERROR - Discipline Index Out of Range: *num*

  A discipline index *num* is not within the range declared in the file.

- ERROR - Partition Index Out of Range: *num*

  The partition index *num* is not valid.

- ERROR - Index Out of Range: *num*

  The index for a field variable is less than 0 or too large.

- ERROR - Extract header mis-match!

  The definition of an extract does not match the **pV3** internals. The file was not written by *pV3Batch*.

- ERROR in Programmed Extract - *index*!

  The definition of a programmed extract does not match the **pV3** internals.

- ERROR - Unknown Extract!

  An unkown extract type was encountered in the file.

- ERROR - SubEntity index out of range: *num*!

  A sub-extract index is incorrect.

- ERROR - Illegal tag: *string*!

  An invalid file marker was found.

- ERROR - Entity Marker Mismatch!

  The trailer of an extract does not match the header.

- ERROR - sub-index out of range: *num*!

  A sub-extract index is incorrect.

- ERROR - Extract *index* Not Registered!

  The programmed extract indexed by *index* has not been registered. Register it via the first invokation of **pVSafe**.

- Extract ERROR ...

  Filling the listed sub-extract required more memory than allocated. This is an internal error that you should not see.

4 Using State-Vectors

4.1 Client-side

The routines PVSTATE and/or PVPSTATE need to be supplied to give the **pV3** system the data for the state-vector associated with the extract. Also, a minor modification to the definition of PV_INIT allows the specification of a state vector (only one is valid):

4.1.1 pV_Init

PV_INIT(TITL, CID, CNAME, DNAME, IOPT, NPGCUT, TPGCUT, NKEYS, IKEYS, TKEYS, FKEYS, FLIMS, MIRROR, REPMAT, MAXBLK, ISTAT)

This subroutine initializes **pV3**. The complete description can be found in the **pV3** Server User's Reference Manual.

IKEYS:i: I(NKEYS)	X-keypress return code for each key. For state-vectors, this indicates the rank (the size of the vector). For example, Euler equations would be 5.
FKEYS:i: I(NKEYS)	Type of function controlled by each key: **FKEYS()=0** State-Vector **FKEYS()=1** Scalar **FKEYS()=2** Vector **FKEYS()=3** Surface scalar **FKEYS()=4** Surface vector **FKEYS()=5** Threshold

4.1.2 pVState

PVSTATE(RANK,KN,SV)

This subroutine supplies **pV3** with state-vector function values when requested.

RANK:i: I	the size of the state-vector as specified via IKEYS at initialization
KN:i: I	index for the requested node
SV:o: R(RANK)	returned state-vector values for the node

4.1.3 pVPState

PVPSTATE(RANK,INDEX,LN,SV)

This subroutine supplies **pV3** with state-vector function values for the local nodes in the specified complex polyhedron.

RANK:i: I	the size of the state-vector as specified via IKEYS at initialization
INDEX:i: I	The index to the polyhedral element (in the range one to KPHEDRA).
LN:i: I	index for the local requested node
SV:o: R(RANK)	returned state-vector values for the node

4.2 Viewer-side

The **pV3** Viewer needs to be modified so that the state-vector data can be translated back to scalar and vector information on the extracted objects. The viewer can be re-built by including the object module of the following routine in the link specification of *pV3Viewer*. This can be found in the file 'Makefile' of the 'servers' and 'modules' sub-directories in the distribution.

4.2.1 pVConvert

PVCONVERT(DISCIPLINE,RANK,LEN,XYZ,SV,ISCL,S,IVCT,V,ITHR,T)

This subroutine converts state-vectors to **pV3** functional values. The indices used for the field variables (scalars and vectors) are the values specified at batch server run time (when the data was collected). Therefore, all the field variables of interest must be specified at the client-side – even if state-vectors are always written.

DISCIPLINE:i: C*20	the name for the discipline
RANK:i: I	the size of the state-vector
LEN:i: I	the number of nodes in the object
XYZ:i: R(3,LEN)	the (x, y, z)-coordinates for the nodes
SV:i: R(RANK,LEN)	the state-vector values for all the nodes
JSCL:i: I	scalar index (JKEY) - if zero do not fill S
S:o: R(LEN)	returned scalar function values
JVCT:i: I	vector index (JKEY) - if zero do not fill V
V:o: R(3,LEN)	returned vector function values (Vx, Vy, Vz). If right-handed coordinates are desired reverse sign of the Vz values
JTHR:i: I	threshold index (JKEY) - if zero do not fill T
T:o: R(LEN)	returned threshold function values