Data Collector for Ufiles

R. H. Bulmer

February 12, 2002

1 Introduction

This document describes a Ufile data collector, dc, and associated routines to expedite importation of data from Ufiles into a Caltrans session.

The data collector works in two distinct phases: (1) initially it collects the data from many Ufiles for a particular shot and stores the data in a single binary file, or database—a one-time operation for each shot; (2) thereafter, the binary data can be imported into Caltrans sessions along with the self-contained auxiliary routines which provide the capability to query the contents of the database, display the data graphically, interpolate at a particular time-point and write formatted data to disk.

The advantages of collecting Ufile data into a single binary file are: (1) much faster data access, and (2) only one file needs to be available for the Caltrans session—the Ufiles can be destroyed after the binary file has been created.

Section 2 contains a brief guide to using the data collector. Section 3 discusses the Ufile formats recognized by dc. Section 4 explains how to prepare for a new project. Section 5 describes how to create a database. Section 6 demonstrates how to use the database. Section 7 shows the various ways to display the database contents, and section 8 summarizes the dc commands.

2 Brief guide

There are three steps involved in making and using a database from a collection of Ufiles: (1) make a text file specifying file and variable naming conventions and descriptions, (2) create the binary database and (3) use the database.

2.1 File and variable name conventions

The first step is to create a file called `names.dc`, identifying Ufile prefix characters (for both file names and variable names) and data descriptions and units for each type of Ufile anticipated. Details of this first step are given in §4; a sample `names.dc` file is listed in Appendix A.

2.2 Making a database (PFB file)

The Ufiles must be in the current working directory and should all pertain to a common shot of interest. Start-up Caltrans, reading the script either on the command-line or with a Basis `read` statement, then execute "`dc`" at the prompt. The data collector will report the Caltrans variable names of the data arrays as they are loaded, where the names are constructed by concatenation of the prefix string specified by the user and the data names from the Ufile suffixes. All such constructed variable names will be in lowercase. The PFB[1] database is written to disk with a name of the form: *`shot`*`.pfb`.

Making a binary database from a collection of Ufiles:

```
    caltrans dc.bas
    dc
```

2.3 Using the database

The *`shot`*`.pfb` file can be used in future sessions by importing it with the Basis `restore` statement. Contents of the database can be listed with the `dcl` command; `dcp` and `dcp2` plot the data.

Using the database:

```
    caltrans
    restore shot.pfb
    dcl # list database contents
    dcp # plot 1D data
    dcp2 # plot 2D data
```

There are also other routines, enter "`dch`" for details.

[1]The PFB database files are written using the Portable Files from Basis facility. See `http://basis.llnl.gov`, specifically the PFB documentation in the Package Library.

3 Ufile formats

There are four flavors of Ufiles—for scalar (0D), 1D, 2D and 3D quantities, but dc recognizes only the 1D and 2D formats. In the 1D case, the data is interpreted as functions of time and for the 2D case as functions of a radial parameter and time. The radial parameter is assumed to be the normalized minor-radius-like function:

$$x = \sqrt{\frac{\Phi - \Phi_{axis}}{\Phi_{edge} - \Phi_{axis}}}, \quad 0 \le x \le 1$$

where Φ is the toroidal flux.

The one-dimensional data elements, $u = f(t)$, must be contained in Ufile names of the form:

f shot .u

where the prefix character *f* (called `prefix1`) is the same for all 1D files and u represents the variable name of the 1D quantity. The 1D Ufiles may have *different* numbers of elements (time points).

The two-dimensional data elements, $v = f(x, t)$, must be contained in Ufile names of the form:

p shot .v

where, again, the prefix character *p* (called `prefix2`) is the same for all 2D files and v represents the variable name of the 2D quantity. The 2D Ufiles, in contrast to the 1D Ufiles, must all have the *same* number of time and spatial points.

Ufile names may be in either upper or lowercase. The Caltrans variable names—taken from the u and v Ufile suffix strings—will all be in lowercase.

4 Preparation for a new project

Create a file named `names.dc` with the contents described below. A single `names.dc` file should be created for each project and placed somewhere in your Basis search path.

The format of the `names.dc` file is as follows. Comment lines begin with the "#" character; symbols and strings are delimited with the SPACE and/or TAB characters. This file consists of:

1. A line with five symbols, the first symbol is the project name (used to label output); the second symbol, `prefix1`, is the prefix character for the Ufiles containing 1D data, $u = f(t)$; the third symbol, `prefix2`, is the prefix character for Ufiles containing 2D data $v = f(x, t)$; the fourth symbol, `base`, is used as a *base* name for dc-generated variable names; the fifth symbol, `prefix0`, is used as a prefix string for interpolants. The symbols may be one or more characters in length.

2. Mandatory comment line(s) prefacing 1D data descriptions to follow.
3. A line for each 1D data element $u = f(t)$ containing: (a) the variable name (Ufile suffix, but lowercase), (b) units of the variable and (c) the variable description; if a variable is dimensionless, enter `"---"` in the units field.
4. A mandatory comment line(s) prefacing 2D data descriptions to follow.
5. A line for each 2D data element $v = f(x, t)$ containing: (a) the variable name (Ufile suffix, but lowercase), (b) units of the variable and (c) the variable description; if a variable is dimensionless, enter `"---"` in the units field.

Unit and description strings containing blanks or non-alpha-numeric characters must be quoted and they are limited to 40 characters in length.

The `names.dc` file need be created only once for each project. A sample input file for DIII-D is listed in Appendix A.

5 Creating a shot database

When a new set of shot data files are available, start-up Caltrans in the directory where the Ufiles exist and execute the data collector:

```
caltrans dc.bas
dc
```

As the data collector reads each Ufile a one-line summary will be displayed showing the one or two-dimensional Caltrans variable (array) name and number of elements. The variable names are constructed by concatenating the user-supplied base name, an underscore and the Ufile suffix characters (all lowercase).

It is not necessary to have a Ufile present for each entry in the `names.dc` file. Missing data will be flagged and omitted from the dc interpolation, graphics and file generation routines. In this way, only one `names.dc` file is needed for each project but the number of Ufiles available can be shot-dependent.

After all the Ufiles are read, dc writes the binary database file: *`shot`*`.pfb`. The database file contains the data from all of the Ufiles plus the `dc.bas` routines. The session can be terminated at this point.

6 Using the database

The binary database created as in §5 is portable across all Unix platforms, so need be created only once. It is not only self-describing as are all PFB files, but it is self-contained: it contains both the data and the dc routines needed to operate on it. There is no need to "`read dc.bas`" to use it, in fact, using the self-contained versions of the dc routines assures self-consistency between the data and the auxiliary functions. Therefore, just restore the database into a session with the Basis `restore` statement:

```
restore shot.pfb
```

Typical actions after a database has been restored are to display a summary of its contents with the `dcl` command, interpolate for values at a particular time point with `dct(`t`)` or at multiple time points with `dct(`$[t_1, t_2, \cdots, t_n]$`)` or to simply access the data directly, referencing it by name. The data interpolation and display routines are described in detail in §7.

Most applications of the database use an interface routine to map interpolants to application-specific arrays (e.g., the `pick` routine used in DIII-D applications), so the usage looks like this:

```
restore shot.pfb
dct(t)
pick
⋮
```

One may also reference the data directly by name. The 1D database variables are named: *base_u*(*base_*`t1`*_u*), with the variables `nt1`*_base_u* containing the number of points for each data element, and the 2D database variables are named *base_v*(*base_*`x`,*base_*`t2`) with the variables `nx`*_base* and `nt2`*_base* containing the number of points.

Remember that the 1D quantities may exist on different time spans, i.e., there is a unique time array *base_*`t1`*_u* variable associated with each 1D data element. In the case of 2D data elements, all quantities must have the *same* spatial and temporal independent variables.

7 Data display and selection routines

The `dc` routines are self-contained in each *shot*`.pfb` file to support data display, interpolation and exportation. To display a summary of the functions during a session, enter:

```
dch # display help message for all dc.bas routines
```

7.1 Listing database contents

To list the database element names and descriptions use the `dcl` routine, which accepts an optional non-zero argument to direct output to the graphics device(s) instead of the terminal window:

```
dcl # list database contents at the terminal window
dcl(1) # send list to graphics devices(s)
```

Sample `dcl` output is listed in Appendix B.

7.2 Hiding data elements

Often there will be more shot data than you wish to see. When the database is created flags are set to indicate which arrays have been filled from Ufiles. The array of flags for one-dimensional data is (`d1flag`) and for the two-dimensional data (`d2flag`). If no data exists for a particular element, its flag is set to -1, and if the data exists its flag is set to $+1$. You may optionally set any of the "$+1$" flags to zero to omit further processing (interpolation, plotting, etc.) of the corresponding data elements. For example, entering:

```
d1flag(5)=0; d2flag(3)=0; d2flag(7)=0
```

will turn-off graphics and interpolation for the 5th 1D data element and the 3rd and 7th 2D data elements. Use the `dcl` command to associate a data element name with its element number.

Alternatively, the `dcs` command can be used to cycle through all data names, prompting the user to reply "yes" or "no" to use each data element in subsequent processing:

```
dcs # [de]select data for interpolation and display
```

At the prompt (one per data element), enter "y" (or RETURN) to *use* the data element and "n" to *omit* the data element from further processing. Elements that are omitted can be reinstated by reissuing the `dcs` command.

7.3 Plotting data

To plot all of the 1D data elements versus time use the `dcp` command:

```
dcp # plot 1D data
```

If you are directing graphical output to an X-window, `dcp` will pause after each frame is displayed. The user may respond to the pause prompt with a RETURN key to continue or an integer number of seconds to change the pause action from a full pause to a timed delay.

In a similar fashion, to plot all of the two-dimensional data versus time and radial position , enter:

```
dcp2 # plot 2D data
```

As with the `dcp` command, a pause will be executed after each frame is displayed if a graphics window is open.

7.4 Interpolation

Frequently the user will need to interpolate for quantities at a particular time (or times). The `dct(`t`)` command, where t is the time in seconds, can be used:

```
dct(t) # interpolate at t seconds
```

The time argument t can also be a vector of times, so the following forms are allowed:

```
dct(t_vector) # with predefined times in t_vector
dct([t1,t2,···,tn]) # with a list of times
dct(Δt*iota(n)) # with a uniform sequence
```

The interpolants will be returned in variable names prefixed with the contents of `prefix0` as specified by the user in the `names.dc` file. Use the `dctl` command to list the explicit names of all interpolants:

```
dctl # list interpolant names
```

7.5 Plotting profile data

After executing the `dct` command you may view profile plots of the various quantities as a function of x at the selected times with the `dcpx` command:

```
dcpx # plot 2D interpolated quantities versus x
```

As with the `dcp` and `dcp2` commands, a pause action will occur after each frame is displayed.

7.6 Writing data to disk

The `dcw` command writes a formatted text file of the interpolants from `dct`.

```
dcw # write interpolated data
```

which will create a file named `*shot.time*.dat` file for each time point specified in the `dct` call.

8 Summary of commands

Most of the dc routines do not accept arguments (exceptions are `dcl` and `dct`). The list below summarizes the dc routines.

`dc`	collect Ufile shot data and create a new PFB database
`dch`	display a summary of `dc.bas` commands
`dcl(;plot)`	list database elements and descriptions
`dcp`	plot 1D data versus time
`dcp2`	plot 2D data versus radial position and time
`dcpx`	make profile plots for times specified by `dct`
`dcs`	select data elements to process
`dct(t)`	interpolate at the specified time point, or
`dct([t1,···,tn])`	interpolate at the specified times
`dctl`	list names of new variables created by `dct`
`dcw`	write files for time(s) specified by `dct`

The optional *`plot`* argument to `dcl`, if it is present, will direct the list to the graphics device(s) instead of the terminal window. The time argument to `dct` is in units of seconds, and may be a scalar or vector of time points.

Appendix

A Sample `names.dc` file

The `names.dc` file uses "#" as the comment character and blank lines are ignored. Symbols may be separated by SPACE and/or TAB characters; they must be quoted if they contain embedded spaces or non-alpha-numeric characters.

The 1st non-comment line specifies the project name and four prefix strings:

`project prefix1 prefix2 base prefix0`

where:

`project` — string used only for labeling output

`prefix1` — prefix character(s) of the 1D Ufiles

`prefix2` — prefix character(s) of the 2D Ufiles

`base` — prefix string for Caltrans variable names, an underscore will also be appended by dc

`prefix0` — prefix string for interpolants created by `dct`

The Ufile prefix strings *`prefix1`* and *`prefix2`* are *case-insensitive*. The remaining lines contain strings associating units and descriptions to each of the Ufile variable names (the variable names must match each Ufile filename suffix).

```
# ================================================================
# Name definitions for DIII-D Ufile data
# ================================================================

# Label and 1D Ufile, 2D Ufile, variable, and interpolant prefixes
  "DIII-D"  f  p  d3  zz

# 1D data element names...
# NAME   UNITS     DESCRIPTION
  bdi    "---"     "Diamagnetic beta"
  dfx    "Wb"      "Diamagnetic flux"
  fnr    "1/sec"   "Neutron rate (plastic)"
  ipl    "A"       "Plasma current"
  l2b    "---"     "li/2 + betap"
  rbt    "m-T"     "RB_tor (vacuum)"
  rcy    "n/sec"   "Neutral flux input"
  vsf    "V"       "Surface voltage"

# 2D data element names...
# NAME   UNITS        DESCRIPTION
  nel    "1/m^3"     "Electron density"
  nim    "1/m^3"     "Impurity density"
  nio    "1/m^3"     "Ion density"
  ome    "rad/sec"   "Omega"
  qp2    "---"       "Safety factor (q)"
  qra    "W/cm^3"    "Radiated power"
  tel    "keV"       "Electron temperature"
  tio    "keV"       "Ion temperature"
  zf2    "---"       "Zeff (CER)"
```

B Sample `dcl` listing

A line for each entry in the `names.dc` file is listed. Where a Ufile is not available for the data element it is marked with "`*`". A "`+`" sign means the data is active (inactive elements are marked with "`-`").

The length(s) of each 1D or 2D array are included, along with the time span. The time spans for 1D arrays may not be the same so there is a time array for each 1D data array. The 1D time arrays have names of the form *base*_t1_*name*.

Both the number of radial elements and time elements for 2D arrays must be identical. The vector of radial coordinates is *base*_x and the vector of time points is *base*_t2.

```
DIII-D   shot #103740

  8 1D u=f(t) arrays with variable length...
  1 + d3_bdi  ---    Diamagnetic beta        168 from:  0.622 to 3.889 s
  2 + d3_dfx  Wb     Diamagnetic flux        168 from:  0.622 to 3.889 s
  3 + d3_fnr  1/sec  Neutron rate (plastic) 4096 from: -1.116 to 6.425 s
  4 + d3_ipl  A      Plasma current         4096 from: -1.116 to 6.425 s
  5 + d3_l2b  ---    li/2 + betap            168 from:  0.622 to 3.889 s
  6 + d3_rbt  m-T    RB_tor (vacuum)         168 from:  0.622 to 3.889 s
  7 * d3_rcy  n/sec  Neutral flux input        0
  8 + d3_vsf  V      Surface voltage         168 from:  0.622 to 3.889 s

      independent variable (time) in arrays: d3_t1_<name>

  9 2D v=f(x,t) arrays with dimension (101,168) from:  0.622 to  3.889 s
  1 + d3_nel  1/m^3    Electron density
  2 + d3_nim  1/m^3    Impurity density
  3 + d3_nio  1/m^3    Ion density
  4 + d3_ome  rad/sec  Omega
  5 * d3_qp2  ---      Safety factor (q)
  6 + d3_qra  W/cm^3   Radiated power
  7 + d3_tel  keV      Electron temperature
  8 + d3_tio  keV      Ion temperature
  9 * d3_zf2  ---      Zeff (CER)

      independent variables (x,time) in arrays: d3_x and d3_t2

Symbols "+", "-", "*" ==> "active", "inactive", "empty"

Valid time range (t_min,t_max) for active data: 0.6220 to 3.8890 s
```