Introduction to Tecplot v10.0

Xingyu Gao
<gaoxy@lsec.cc.ac.cn>
Institute of Computational Mathematics,
Chinese Academy of Sciences

April 20, 2006

Outline

1. Start up by the help source

2. Data organization

3. Ordered data

4. Finite element data

5. Syntax in writing ASCII data files for Tecplot

6. ASCII data file conversion to binary

7. Data loaders: Tecplot's import feature

1 Start up by the help sources

On UNIX system, we could use "tar xvfz(j)" to unzip before installation. Run tecplot by typing "tecplot" at the shell prompt.

After installation, there appears a folder "TEC100" on your installation platform. This folder contains numerous sources facilitate quick learning. Let's start up with the arm of these helpful sources.

Three sub-folders worth noting:

Demo contains the binary file of plots for tecplot

Examples contains the examples of ASCII data files, loaders as well as macro files

Tutorials contains teaching flash movies

Direct attack by three examples:

- Meet the workspace and frames by a flash moive
- Create the contour plots
- Create the XY line with legend

Capabilities of post-processing

- Field plots, any 2D Cartesian or 3D Cartesian plot
- XY and polar line plots, two different types of line plots
- Mesh plots and boundary plots. A mesh plot is a field plot of the lines connecting neighboring data points within a zone. By changing the zone's Surfaces to Plot on the Surfaces page of the Zone Style dialog, you may limit the plotted mesh to the exterior surface (exposed cell faces) or to selected I-, J-, and K-grid planes. For finite element zones, the mesh is a plot of all edges of all elements which are defined by the connectivity list for the

node points. Mesh lines are straight lines between adjacent points. A boundary plot is a field plot of the boundaries of ordered data zones or 2D finite element zones. Boundary plots are frequently combined with other plot types.

- Contour plots, the variation of one variable across the data field. Plotting contours allows you to add an extra dimension to a plot, and thus contour plots may require one more variable than mesh plots of the same dimension. Where a 2D mesh plot shows two variables, a contour plot can show three, and a 3D contour plot can show four. This extra variable is called the contour variable.

- Vector plots, field plots of the direction and or magnitude of vector quantities. The vector quantities can be displacements, velocities, forces, or anything else that can be represented by vectors. One important use of vector components in Tecplot is to allow you to compute the trajectories of massless particles in a steady-state velocity field. These trajectories are called streamtraces.

- Streamtrace plots, the path traced by a massless particle placed at an arbitrary location in a steady state vector field.

- Scatter plots, plots of symbols at the data points in a field. The symbols may be sized according to the values of a specified variable, colored by the values of the contour variable, or may be uniformly sized or colored. Scatter plots do not require any mesh structure connecting the points, allowing you to make scatter plots of irregular data.

2. Data organization and ASCII data for Tecplot

Tecplot structures data in two levels. The highest level is a data set. Data sets consist of one or more zones. Zones, blocks of data making up a data set, are the second level which can be created in Tecplot, or loaded from a file.

If more than one data file is read into a frame, Tecplot groups all the zones from the files into one data set. Once in Tecplot, all zones in a data set must contain the same variables defined for each data point. (This does not necessarily mean each of your data files needs to have the same number of variables in the same order.) The number of zones in a concatenated data set is the sum of the number of zones in the loaded data files.

Data structures in zones

Tecplot accommodates two different types of data: ordered and finite-element. We will learn the ASCII representation for the two different data structure respectively. After that we can produce ASCII data files by C or FORTRAN which can be directly loaded by Tecplot to generate the wanted plots.

3. Ordered data

We will meet three types of ordered data referred by I-, IJ- IJK-ordered data. For ordered data, the numerical values in the zone data must be in either POINT or BLOCK format, specified by the DATAPACKING parameter. However, only BLOCK format is available if the cell-centered variables is involved.

I-ordered data

In I-ordered data, the I-index varies from one to IMax. The total number of data points is IMax. For zones with only nodal variables, the total number of values in the zone data is IMax*N (where N is the number of variables). For a mixture of nodal and cell-centered variables, the number of values in the zone data is IMax*Nn+(IMax-1)*Nc, where Nn is the number of nodal variables and Nc is the number of cell-centered variables. For data in POINT format, IMax is calculated by Tecplot from the zone data if it is not explicitly set by the zone control line (using the I-parameter).

I-ordered data in POINT format example

```
VARIABLES = "X","Y" ZONE I=5, DATAPACKING=POINT
2 4
3 9
5 25
6 36
7 49
```

FORTRAN code example to generate I-ordered data in POINT format

```
INTEGER VAR . . . WRITE (*,*) ' ZONE DATAPACKING=POINT, I= ', IMAX
DO 1 I=1,IMAX
    DO 1 VAR=1, NUMVAR
1       WRITE (*,*) ARRAY(VAR,I)
```

I-ordered data in BLOCK format example

```
VARIABLES = "X", "Y" ZONE I=5, DATAPACKING=BLOCK
2 3 5 6 7
4 9 25 36 49
```

FORTRAN code example to generate I-ordered data in BLOCK format

```
INTEGER VAR . . . WRITE (*,*) ’ ZONE DATAPACKING=POINT, I= ’, IMAX
DO 1 VAR=1, NUMVAR
    DO 1 I=1, IMAX
1       WRITE (*,*) ARRAY(VAR,I)
```

Multi-zone XY line plot example

A date file in POINT format is given below:

```
TITLE="Example: Multi-zone XY line plot"
VARIABLES ="Position","Temperature", "Pressure"

ZONE T="0.0 seconds",I=4
71.30 563.7 101362.5
86.70 556.7 101349.6
103.1 540.8 101345.4
124.4 449.2 101345.2
```

```
ZONE T="0.1 seconds", I=4
71.31 564.9 101362.1
84.42 553.1 101348.9
103.1 540.5 101344.0
124.8 458.5 101342.2

TEXT CS=FRAME, HU=POINT, X=16, Y=90, H=28, T="SAMPLE CASE"
```

A data file in BLOCK format is shown below. All of the values for the first variable (Position) at each data point are listed first, then all of the values for the second variable (Temperature) at each data point, and so forth.

```
TITLE = "Example: Multi-Zone XY Line Plot"
VARIABLES="Position","Temperature", "Pressure"
```

```
ZONE DATAPACKING=BLOCK, T="0.0 seconds",I=4
71.30 86.70 103.1 124.4
563.7 556.7 540.8 449.2
101362.5 101349.6 101345.4 101345.2

ZONE DATAPACKING=BLOCK, T="0.1 seconds", I=4
71.31 84.42 103.1 124.8
564.9 553.1 540.5 458.5
101362.1 101348.9 101344.0 101342.2

TEXT CS=FRAME, HU=POINT, X=16, Y=90, H=28, T="SAMPLE CASE"
```

A more compact data file for this example is in the point format shown below. Tecplot determines the number of variables from the number of values in the first line of data under the first zone. The variables and zones are assigned default names.

```
ZONE
71.30 563.7 101362.5
86.70 556.7 101349.6 103.1 540.8 101345.4 124.4 449.2 101345.2

ZONE
71.31 564.9 101362.1
84.42 553.1 101348.9 103.1 540.5 101344.0 124.8 458.5 101342.2

TEXT CS=FRAME, HU=POINT, X=16, Y=90, H=28, T="SAMPLE CASE"
```

Multi-zone XY line plot with variable sharing example

If the above data was taken at the same position for both times, variable sharing could reduce memory usage and file size. That file appears as:

```
TITLE ="Example: Multi-zone XY line plot with variable sharing"
VARIABLES = "Position", "Temperature", "Pressure"

ZONE T="0.0 seconds", I=4
71.30 563.7 101362.5
86.70 556.7 101349.6
103.1 540.8 101345.4
124.4 449.2 101345.2
```

```
ZONE T="0.1 seconds", I=4, VARSHARELIST=([1]=1)
564.9 101362.1
553.1 101348.9
540.5 101344.0
458.5 101342.2

TEXT CS=FRAME, HU=POINT, X=16, Y=90, H=28, T="SAMPLE VARIABLE
SHARING CASE"
```

IJ-ordered data

IJ-ordered data has two indices: I and J. IJ-ordered data is typically used for 2 and 3D surface mesh, contour, vector, and shade plots, but it can also be used to plot families of lines in XY plots. In IJ-ordered data, the I-index varies from 1 to IMax, and the J-index varies from one to JMax. The total number of data points (nodes) is IMax*JMax. For zones with only nodal variables, the total number of numerical values in the zone data is IMax*JMax*N (where N is the number of variables). For a mixture of nodal and cell-centered variables, the number of values in the zone data is IMax*JMax*Nn+(IMax-1)*(JMax-1)*Nc, where Nn is the number of nodal variables and Nc is the number of cell-centered variables. Both IMax and JMax must be specified in the zone control line (with the I and J parameters). The I- and J-indices should not be confused with the X- and Y-coordinates on occasions the two may coincide, but this is not the typical case. The

I-index varies the fastest, consistent with the storage scheme of FORTRAN. That is, when you write programs to print IJ-ordered data, the I-index is the inner loop and the J-index is the outer loop. Note the similarity between I-ordered data and IJ-ordered data with JMax=1.

IJ-ordered data in POINT format example

```
VARIABLES = "X", "Y", "Temperature", "Pressure"
ZONE I=2, J=3,DATAPACKING=POINT
3 0 0 50
7 2 0 43
2 4 1 42
6 6 0 37
1 8 1 30
5 9 1 21
```

FORTRAN code to generate IJ-ordered data in POINT format example

```
WRITE (*,*) ' VARIABLES = "X", "Y", "Temperature", "Pressure"'
WRITE (*,*) ' ZONE I=', IMAX, ', J=', JMAX,', 'DATAPACKING=POINT'

DO 1 J=1,JMAX
  DO 1 I=1, IMAX
1   WRITE (*,*) X(I,J), Y(I,J), T(I,J), P(I,J)
```

IJ-ordered data set in BLOCK format example

```
VARIABLES = "X", "Y", "Temperature","Pressure"
ZONE I=2, J=3, DATAPACKING=BLOCK
3 7 2 6 1 5
0 2 4 6 8 9
0 0 1 0 1 1
50 43 42 37 30 21
```

FORTRAN code to generate IJ-ordered data in BLOCK format example

```
 INTEGER VAR . . .
WRITE (*,*) 'ZONE DATAPACKING=BLOCK, I=', IMAX,', J=', JMAX DO 1
VAR=1,NUMVAR
  DO 1 J=1,JMAX
    DO 1 I=1,IMAX
1     WRITE (*,*) ARRAY(VAR,I,J)
```

IJ-ordered data with cell-centered data

```
VARIABLES = "X", "Y", "Temperature", "Pressure"

ZONE I=3, J=3,DATAPACKING=BLOCK, VARLOCATION=(3=CELLCENTERED,
4=CELLCENTERED)
3 7 11 2 6 10 1 5 9
0 2 3 4 6 8 8 9 10
0 2 1 3
45 60 35 70
```

Note that zones with cell-centered data must have DATAPACKING=BLOCK.

IJK-ordered data

IJK-ordered data has three indices: I, J, and K. This type of data is typically used for 3-D volume plots, although planes of the data can be used for 2- and 3-D surface plots. In IJK-ordered data, the I-index varies from 1 to IMax, the J-index varies from one to JMax, and the K-index varies from one to KMax. The total number of data points (nodes) is IMax*JMax*KMax. For zones with only nodal variables the total number of values in the zone data is IMax*JMax*KMax*N, where N is the number of variables. For a mixture of nodal and cell-centered variables, the number of values in the zone data is IMax*JMax*KMax*Nn+(IMax-1)*(JMax-1)*(KMax-1)*Nc, where Nn is the number of nodal variables and Nc is the number of cell-centered variables. The three indices, IMax, JMax, and KMax, must be specified in the zone control line using the I-, J-, and K-parameters. The I-index varies the fastest; the J-index the next fastest; the K-index the slowest. If you write a program to print IJK-ordered data, the I-index is the inner loop, the K-index is the outer loop, and the J-index is the loop in between. Note the similarity between IJ-ordered data and IJK-ordered data with KMax=1.

IJK-ordered data in POINT format example

```
VARIABLES = "X" "Y" "Z" "Temp"
ZONE I=3, J=2, K=2, DATAPACKING=POINT
0 0 0 0
3 0 1 5
6 0 3 10
0 6 3 10
3 6 4 41
6 6 6 72
0 0 8 0
3 0 9 29
6 0 11 66
```

```
0 6 11 66

3 6 12 130

6 6 14 169
```

FORTRAN code to generate an IJK-ordered zone in POINT format example

```
WRITE (*,*) 'VARIABLES = "X", "Y", "Z", "Temp"'
WRITE (*,*) 'ZONE I=',IMAX,' J=',JMAX,' K=',KMAX,'DATAPACKING=POINT'

DO 1 K=1,KMAX
  DO 1 J=1,JMAX
    DO 1 I=1,IMAX
1      WRITE (*,*) X(I,J,K), Y(I,J,K), Z(I,J,K), Temp(I,J,K)
```

BLOCK format of the same data

```
VARIABLES = "X" "Y" "Z" "Temp" ZONE I=3, J=2, K=2, DATAPACKING=BLOCK
0 3 6 0 3 6 0 3 6 0 3 6
0 0 0 6 6 6 0 0 0 6 6 6
0 1 3 3 4 6 8 9 11 11 12 14
0 5 10 10 41 72 0 29 66 66 130 169
```

FORTRAN code to generate IJK-ordered data in BLOCK format example

```
INTEGER VAR
. . .
WRITE (*,*) 'DATAPACKING=BLOCK, I=', IMAX,', J=', JMAX,', K=', KMAX

DO 1 VAR=1,NUMVAR
  DO 1 K=1,KMAX
    DO 1 J=1,JMAX
      DO 1 I=1,IMAX
1       WRITE (*,*) ARRAY(VAR,I,J,K)
```

4. Finite element data

For finite element data, the zone types, specified in the ZONETYPE parameter, may be FELINESEG, FETRIANGLE, FEQUADRILATERAL, FETETRAHEDRON, or FEBRICK. For any of these DATAPACKING may be POINT or BLOCK. The number of nodes (data points) is given by the N=numnodes parameter, and the number of elements is given by the E=numelements parameter (this is also the total length of the connectivity list). Zone data is divided into two logical sections. It has no markers, but you may place blank lines between the sections to distinguish them. The first section, the node (and sometimes element) data, lists the values of the variables at the data points (or nodes) or cell-centers (elements) as if they were I-ordered (one-dimensional) zone data. The second section, the connectivity list, defines how the nodes are connected to form elements. There must be numelements lines in the second section; each line defines one element. The number of nodes per line in the connectivity list depends on the element type specified in the zone control line (ZONETYPE parameter).

Triangle data in POINT format example

```
VARIABLES = "X", "Y"
ZONE N=5,E=3,DATAPACKING=POINT, ZONETYPE=FETRIANGLE
1.0 1.0
2.0 3.0
2.5 1.0
3.5 5.0
4.0 1.0
1 2 3
3 2 4
3 5 4
```

BLOCK format of the same data

```
VARIABLES = "X", "Y"
ZONE N=5, E=3, DATAPACKING=BLOCK, ZONETYPE=FETRIANGLE
1.0 2.0 2.5 3.5 4.0
1.0 3.0 1.0 5.0 1.0
1 2 3
3 2 4
3 5 4
```

Variable and connectivity list sharing

```
TITLE = "Example: Variable and Connectivity List Sharing"
VARIABLES = "X", "Y", "P"
ZONE T="P_1", DATAPACKING=POINT, N=6, E=4, ZONETYPE=FETRIANGLE
-1.0 0.0 100
0.0 0.0 125
1.0 0.0 150
-0.5 0.8 150
0.5 0.8 175
0.0 1.6 200
1 2 4
2 5 4
```

```
3 5 2

5 6 4


ZONE T="P_2", DATAPACKING=POINT, N=6, E=4, ZONETYPE=FETRIANGLE,

VARSHARELIST = ([1, 2]=1), CONNECTIVITY SHAREZONE = 1

110

135

160

165

185

200


ZONE T="P_3", DATAPACKING=POINT, N=6, E=4, ZONETYPE=FETRIANGLE,

VARSHARELIST = ([1, 2]=1), CONNECTIVITYSHAREZONE = 1

120
```

```
145
180
175
195
200
```

Example: 2D finite element data

As POINT format:

```
TITLE = "Example: 2D Finite-Element Data"
VARIABLES = "X", "Y", "P", "T"
ZONE N=8, E=4, DATAPACKING=POINT, ZONETYPE=FEQUADRILATERAL
0.0 1.0 100.0 1.6
1.0 1.0 150.0 1.5
3.0 1.0 300.0 2.0
0.0 0.0 50.0 1.0
1.0 0.0 100.0 1.4
3.0 0.0 200.0 2.2
```

```
4.0 0.0 400.0 3.0
2.0 2.0 280.0 1.9
1 2 5 4
2 3 6 5
6 7 3 3
3 2 8 8
```

As BLOCK format:

```
TITLE = "Example: 2D Finite-Element Data"
VARIABLES = "X", "Y", "P", "T"
ZONE N=8, E=4, DATAPACKING=BLOCK, ZONETYPE=FEQUADRILATERAL
0.0 1.0 3.0 0.0 1.0 3.0 4.0 2.0
1.0 1.0 1.0 0.0 0.0 0.0 0.0 2.0
100.0 150.0 300.0 50.0 100.0 200.0 400.0 280.0
```

```
1.6 1.5 2.0 1.0 1.4 2.2 3.0 1.9

1 2 5 4

2 3 6 5

6 7 3 3

3 2 8 8
```

Triangulated data sets

If you have 2D data without a mesh structure, it is probably simplest to enter your data points as an I-ordered data set, then use Tecplot's triangulation feature to create a finite-element data set. You can then edit the file, and particularly the connectivity list, to obtain the set of elements you want, rather than having to create the entire connectivity list by hand. See an example as following.

```
VARIABLES = "X", "Y", "T"
0.000000 0.000000 0.000000
0.200000 0.000000 0.000000
0.400000 0.000000 0.000000
0.600000 0.000000 0.000000
0.800000 0.000000 0.000000
1.000000 0.000000 0.000000
0.000000 0.200000 0.000000
0.200000 0.200000 0.000547
0.400000 0.200000 0.000742
```

```
0.600000 0.200000 0.000741
0.800000 0.200000 0.000522
1.000000 0.200000 0.000000
0.000000 0.400000 0.000000
0.200000 0.400000 0.000740
0.400000 0.400000 0.000760
0.600000 0.400000 0.000760
0.800000 0.400000 0.000687
1.000000 0.400000 0.000000
0.000000 0.600000 0.000000
0.200000 0.600000 0.000740
0.400000 0.600000 0.000760
0.600000 0.600000 0.000761
0.800000 0.600000 0.000692
1.000000 0.600000 0.000000
0.000000 0.800000 0.000000
0.200000 0.800000 0.000532
0.400000 0.800000 0.000742
0.600000 0.800000 0.000742
0.800000 0.800000 0.000544
1.000000 0.800000 0.000000
```

```
0.000000 1.000000 0.000000
0.200000 1.000000 0.000000
0.400000 1.000000 0.000000
0.600000 1.000000 0.000000
0.800000 1.000000 0.000000
1.000000 1.000000 0.000000
0.300000 0.300000 0.000749
0.500000 0.300000 0.000754
0.700000 0.300000 0.000749
0.300000 0.500000 0.000755
0.700000 0.500000 0.000757
0.300000 0.700000 0.000749
0.500000 0.700000 0.000754
0.700000 0.700000 0.000750
0.500000 0.500000 0.000762
0.761603 0.561603 0.000753
0.761603 0.438397 0.000753
0.732288 0.400000 0.000754
0.732288 0.600000 0.000754
```

The steps of triangulation:

1. Read the data file into Tecplot and switch the plot type to 2D Cartesian.
2. From the Data menu, choose Triangulate, then select the simple ordered zone as the source zone, and click Compute.
3. From the File menu, choose Write Data File. The Write Data File Options dialog appears.
4. Select the ASCII check box and the Point Format check box, then click OK.
5. Save to a file name of your choice.

Now we check the triangulated data file and load it to generate the figure as follows.

5. Syntax in writing Tecplot ASCII data files

Data size. Each variable in each zone in the data file may have its own data type. Tecplot supports the following six data types:

- SINGLE (four-byte floating point values).
- DOUBLE (eight-byte floating point values).
- LONGINT (four-byte integer values).
- SHORTINT (two-byte integer values).
- BYTE (one-byte integer values, from 0 to 255).
- BIT

The data type determines the amount of storage Tecplot assigns to each variable. Therefore, the lowest level data type should be used whenever possible. For example, imaging data, which usually consists of numerical values ranging from zero to 255, should be given a data type of

BYTE. By default, Tecplot treats numeric data as data type SINGLE. If any variable in the zone uses the BIT data type, the zone format must be BLOCK or FEBLOCK; you cannot use POINT or FEPOINT format. We could specify the data types by writing in the control line as **DT = (datatype datatype ... datatype)**.

Variable location. Each variable in each zone in a data file may be located at the nodes or the cell-centers. Each variable is specified as NODAL or CELLCENTERED in the VARLOCATION parameter array, located in the control line. The format is:
VARLOCATION=([set-of-vars]=var-location,[set-of-vars]=var-location, ...),
where set-of-vars is the set of the variables and var-location is either NODAL or CELLCENTERED. Variables omitted from the list are assumed to be NODAL. For example:
VARLOCATION=([3-7,10]=CELLCENTERED, [11-12]=CELLCENTERED)
specifies that variables 3 through 7, 10, 11 and 12 are cell-centered and all other variables are, by default, nodal for this zone. All cell-centered variables must list one value for each element. With nodal variables, one value must be listed for each node. Zones with cell-centered variables must

be in BLOCK data packing format.

Data lists. Numerical values in zone data must be separated by one or more spaces, commas, tabs, new lines, or carriage returns. Blank lines are ignored. Integer (101325), floating point (101325.0), and exponential (1.01325E+05) numbers are accepted. To repeat a particular number in the data, precede it with a repetition number as follows: **"Rep*Num,"** where Rep is the repetition factor and Num is some numeric value to be repeated. For example, you may represent 37 values of 120.5 followed by 100 values of 0.0 as follows:
37*120.5, 100*0.0

Variable sharing between zones. Frequently, some variables are exactly the same for a set of zones. In this case, Tecplot¡¯s memory usage may be dramatically reduced by sharing the coordinate variables between the zones. The zones that variables are shared from are specified in the VARSHARELIST in the control line of the current zone. The format is:
VARSHARELIST=([set-of-vars]=zzz, [set-of-vars]=zzz)

where set-of-vars is the set of variables that are shared and zzz is the zone they are shared from. If zzz is omitted, the variables are shared from the previous zone. For example:

VARSHARELIST=([4-6,11]=3, [20-23]=1, [13,15])

specifies that variables 4, 5, 6 and 11 are shared from zone 3, variables 20, 21, 22, and 23 are shared from zone 1, and variables 13 and 15 are shared from the previous zone. For variable sharing, ordered zones may only share with ordered zones having the same dimensions. Finite element zones may share with any zone having the same number of nodes, for nodal variables, or the same number of cells, for cell-centered data.

The connectivity list (finite element only) and face-neighbors may be shared between zones using the CONNECTIVITYSHAREZONE parameter in the control line of the current zone. The format is:

CONNECTIVITYSHAREZONE=nnn

where nnn is the number of the zone that the connectivity is shared from. To use connectivity sharing, the zone must have the same number of points and elements, and be the same zone type.

6. ASCII data file conversion to binary

Although Tecplot can read and write ASCII or binary data files, binary data files are more compact and are read into Tecplot much more quickly. Your Tecplot distribution includes Preplot, which converts ASCII to binary data files. You can also use Preplot to debug ASCII data files Tecplot cannot read.

Preplot Options

To use Preplot, type the following command from the UNIX shell prompt, from a DOS prompt, or using the Run command in Windows: **preplot infile [outfile] [options],** where infile is the name of the ASCII data file, outfile is an optional name for the binary data file created by Preplot, and options is a set of options from either the standard set of Preplot options or from a special set of options for reading PLOT3D format files. If outfile is not specified, the binary data file has the same base name as the infile with a .plt extension. You may use a minus sign ("-") in place of either the infile or outfile to specify standard input or standard output, respectively. Any or all of -**iset,** -**jset,** and -**kset** can be set for each zone, but only one of each per zone.

Options with standard Tecplot data files:

- **-d** Turn on debug echo. Use -d2, -d3, -d4 for more detailed debug information.
- **-iset [zone], [start], [end], [skip]** Create the binary data file using only the specified range and skipping for the I-index. The arguments are optional, but the commas are not. The zone parameter specifies which zone this option affects; if not specified, all zones are affected. The start parameter is the starting I-index; the default is one. The end parameter is the ending I-index; the default is the last index value. The skip parameter specifies the I-interval, that is, the distance between indices; one means every index is used, two means every other index, and so on. For example, -iset 1, 3, 7, 2 indicates that for zone 1 only I-index values of 3, 5, and 7 are used. Only one -iset option is allowed per zone.
- **-jset [zone], [start], [end], [skip]** Same as -iset above, except with respect to the J-index.

- **-kset [zone], [start], [end], [skip]** Same as -iset above, except with respect to the K-index.
- **-zonelist start[:end[:skip]],...**Specify the zones to process. You may supply more than one specification. By default Preplot processes all zones.

6. Data loaders: Tecplot's import feature

- Computational Fluid Dynamics General Notation System (CGNS)
- Digital Elevation Map (DEM)
- Digital exchange Format (DXF)
- Excel (Windows only)
- Fluent Version 5 and 6 (.cas and .dat)
- Gridgen

- Hierarchical Data Format (HDF)
- Image
- PLOT3D
- Polygon (PLY)
- Text spreadsheet

Acknowledgements

Cui Tao offered me the introduction to this new software. **Dai Xiaoying** and **Liu Xin** helped me making the slider. And **Zhai Fangman** offered me part of original data files.

Many thanks to them all!

Many thanks for your attendance!