CHAPTER 5 Formatting ASCII Data for Tecplot

This chapter tells you how to format data so your data files may be loaded directly into Tecplot. You can also load data generated by, or tabulated in, other software packages. Amtec has written some data loaders using Tecplot's Add-on Developer's Kit (ADK). These loaders convert data from a number of popular software packages into a format readable by Tecplot. They are described in Chapter 7, "Data Loaders: Tecplot's Import Feature." Tecplot users can also write loaders of their own using the ADK.

Data files read by Tecplot may be binary or ASCII. Reading an ASCII data file into Tecplot can be much slower than reading a binary data file, as binary data files take up less disk space. You can use Tecplot or Preplot to convert ASCII data files to binary. See Section 6.1, "Loading Tecplot-Format Data Files," for details on using Tecplot, or Section 5.5, "Converting ASCII Data Files to Binary," for details on using Preplot.

The following sections describe the format of ASCII data files. The documentation for the binary data file format is included as comments in the Preplot source code. If your data is generated by a computer program written in FORTRAN or C, you may be able to generate binary data files directly using the utilities described in Chapter 11, "Writing Binary Data for Loading into Tecplot," of the *Tecplot Reference Manual*.

5.1. ASCII Data File Records

An ASCII data file begins with an optional file header defining a title for the data file and or the names of the variables. The file header is followed by optional zone records which contain the plot data. Zone records may contain either ordered data or finite-element data. You may also include text records, geometry records, and custom-label records that create text, geometries, and/or custom labels on your plots. Each data file may have up to 32,700 zone records, up to ten custom label records, and any number of text records and geometry records. These records may be in any order.

The first line in a zone, text, geometry, or custom-label record begins with one of the keywords **ZONE**, **TEXT**, **GEOMETRY**, or **CUSTOMLABELS**. The maximum length of a line in a data file is 4,000 characters (unless you edit and recompile the Preplot source code). Any line may be continued onto one or more following lines (except for text enclosed in double quotes [**"**]). Double quotes must be used to enclose character strings with embedded blank spaces or other special characters. A backslash (****) may be used to remove the significance of (or escape) the next character (that is, **\"** produces a single double-quote). Any line that begins with an octothorp (**#**) is treated as a comment and ignored.

The following simple example of a Tecplot ASCII data file has one small zone and a single line of text:

```
TITLE="Simple Data File"
VARIABLES="X" "Y"
ZONE I=4 F=POINT
1 1
2 1
2 2
1 2
TEXT X=10 Y=90 T="Simple Text"
```

The format of the ASCII data file is summarized in Section 5.1.7, “Summary of Data File Records.”

5.1.1. File Header

In the file header of your data file, you may specify an optional title that is displayed in the headers of Tecplot frames. The title line begins with **TITLE=**, followed by the title text enclosed in double-quotes. You may also assign a name to each of the variables by including a line that begins with **VARIABLES=**, followed by each variable’s name enclosed in double quotes. The quoted variable names should be separated by spaces or commas. Tecplot calculates the number of variables (*N*) from the list of variable names. If you do not specify the variable names (and your first zone is in **POINT** or **FEPOINT** format), Tecplot sets the number of variables equal to the number of numeric values in the first line of zone data for the first zone, and names the variables **V1**, **V2**, **V3**, and so forth.

Initially, Tecplot uses the first two variables in data files as the X- and Y-coordinates, and the third variable for the Z-coordinate of 3-D plots. You may, however, order the variables in the data file any way you want, since you can interactively reassign the variables to the X-, Y-, and or Z-axes using Tecplot dialogs.

If the file header occurs in a place other than at the top of the data file, a warning is printed and the header is ignored. This allows you to concatenate two or more ASCII data files before using Tecplot (provided each data file has the same number of variables per data point).

5.1.2. Zone Records

A zone record consists of a control line that begins with the keyword "**`ZONE`**" followed by a set of numerical data called the zone data. The format of the zone control line is shown in Section 5.1.7, "Summary of Data File Records."

5.1.2.1. The Format Parameter. The zone data are in the format specified by the **`F`** (format) parameter in the control line. There are two basic types of zones: ordered and finite-element. Ordered zones have the formats **`POINT`** and **`BLOCK`**; finite-element zones have the formats **`FEPOINT`** and **`FEBLOCK`**. **`POINT`** format is assumed if the **`F`** parameter is omitted (thus, by default, zones are assumed to be ordered). See Section 5.2, "Ordered Data," for more information on ordered zones, and Section 5.3, "Finite-Element Data," for details on finite-element data.

In **`POINT`** and **`FEPOINT`** format, the values for all variables are given for the first point, then the second point, and so on. In **`BLOCK`** and **`FEBLOCK`** format, all of the values for the first variable are given in a block, then all of the values for the second variable, then all of the values for the third, and so forth. More detail on this is given below.

5.1.2.2. A Simple Example of POINT Format. If you have only one zone of data in **`POINT`** format, and it is one-dimensional (that is, *JMax=1*, *KMax =1*), you may omit the zone control line. If you want Tecplot to determine the number of variables, you may create a data file with only the zone data, such as the following:

```
12.5 23 45 1.
14.3 24 46 2.
12.2 24 50 3.
13.3 26 51 4.
13.5 27 55 5.
```

Tecplot calculates the number of data points (*IMax*) in the zone by assuming that each row represents a data point and each column represents a variable, and creates an I-ordered zone. This type of structure is good for XY-plots and scatter plots. If there are multiple zones, two- or three-dimensional zones, finite-element zones, or **`BLOCK`**-format zone data, you must include a zone control line at the beginning of each zone record.

5.1.2.3. Data Types. 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.

5.1.2.4. Listing Your Data. 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
```

5.1.2.5. Zone Types and Their Control Lines. As stated above, there are two distinct types of zones: ordered zones and finite-element zones. Ordered zones are I-, IJ-, and IJK-ordered zones (formats **POINT** and **BLOCK**). Finite-element zones are FE-surface and FE-volume zones (formats **FEPOINT** and **FEBLOCK**). The control lines for these zone types differ in the parameters needed. Both zone types can use the **C** (*color*), **F** (*format*), **T** (*zonetitle*), **D** (*duplist*), and **DT** (*datatype*) parameters, although the format of the **F** and **D** parameters is slightly different for each zone type.

The **T** parameter specifies a title for the zone. This may be any text string up to 64 characters in length. If you supply a longer text string, it is automatically truncated to the first 64 characters. The titles of zones appear in the Plot Attributes and other dialogs, and, optionally, in the XY-plot legend. (You can use keywords in the zone titles to identify sets of zones to enable/disable or to change zone attributes.) The **C** parameter sets an initial color for the zone. This may be overridden interactively, or by use of a stylesheet. The **DT** (*type1*, *type2*, *type3*, ...) parameter specifies the data types for the variables in a zone.

The **D** (*duplist*) parameter specifies a list of variables to duplicate from the preceding zone, which must have the same dimensions (*IMax*, *JMax*, and *KMax*) as the new zone. If a variable is duplicated using the **D** parameter in the zone control line, no values are listed for this variable, and the values of the specified variables are obtained from the previous zone record. For example, if the zone control line has **D=(1,2,4)**, the first values listed would be for variable **V3**, the second, for variable **V5**.

For ordered zones, you may specify the **I** (*IMax*), **J** (*JMax*), and **K** (*KMax*) parameters, which store the number of data points in the I, J, and K directions. **J** and **K** both default to 1. **I** must be specified if **J** is used; **I** and **J** must be specified if **K** is used. If all are omitted, Tecplot assumes an I-ordered zone and calculates *IMax* for you.

Note: **I** and **J** are not equivalent to either the number of variables or the number of data points. The number of data points is equal to the product of **I**, **J**, and **K**.

For finite-element zones, described in Section 5.3, "Finite-Element Data," you must specify the **N** (*numnodes*) and may optionally include the **ET** (*elementtype*), the **E** (*numelements*), and or the **NV** (*nodevalue*) parameter. If the **E** parameter is not specified, Tecplot calculates it from the number of node sets in the connectivity list following the node data. The **NV** (*nodevalue*) parameter specifies the number of variables which represent the "Node" value in FE data.

The **D** (*duplist*) parameter specifies a list of variables to duplicate and/or the keyword **FECONNECT**, which duplicates the connectivity list of the preceding zone. The preceding zone must have the same *numnodes* and *numelements* as the new zone in order to use the **D** parameter.

The following sections give simple examples of zone data in various formats, as well as sample pieces of FORTRAN code that you can use as templates to print out your own data. Note that the sample code is intended only as a general example—the zone data that it produces contains only one value per line. You may want to modify the code to suit your own needs.

5.1.3. Text Record

Text records are used to import text directly from a data file. Text can also be imported into Tecplot using a macro file. Text may be titles, labels, or other information. You may create data files containing only text records and read them into Tecplot just as you would read any other data file. You may delete and edit text originating from data files just like the text that you create interactively.

The text record consists of a single control line. The control line starts with the keyword **TEXT** and has one or more options:

- The text string is defined in the required **T** (*text*) parameter.
- The color is controlled by the **C** (*color*) parameter.
- Use the **CS** (*coordinatesys*) parameter to specify the text coordinate system, either **FRAME** or **GRID**. If you specify the frame coordinate system (the default), the values of the **X** (*xorigin*) and **Y** (*yorigin*) parameters are in frame units; if you specify grid coordinates, **X** and **Y** are in grid units (i.e., units of the physical coordinate system). **X** and **Y** locate the anchor point of the text string.
- Use the **AN** (*textanchor*) parameter to specify the position of the anchor point relative to the text. There are nine possible anchor positions, as shown in Figure 5-1.

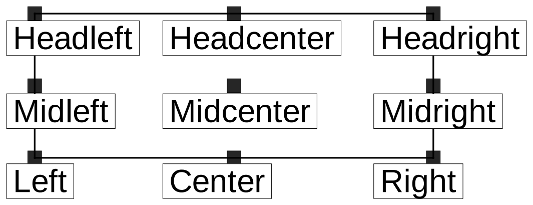

Figure 5-1. Text anchor positions—values for the **AN** parameter.

- Use the **HU** (*heightunits*) parameter to assign units for character heights. If the **CS** parameter is **FRAME**, you can set **HU** to either **FRAME** or **POINT**. If the **CS** parameter is **GRID**, you can set **HU** to either **GRID** or **FRAME**.
- Use the **H** parameter to specify the height; it is measured in the units defined by the **HU** parameter.
- To include multiple lines of text in a single text record, include **\\n** in the text string to indicate a new line.
- You can assign the line spacing for multi-line text using the **LS** (*linespacing*) parameter. The default value, 1, gives single-spacing. Use 1.5 for line-and-a-half spacing, 2 for double-spacing, and so on.

You may optionally draw a box around the text string using the **BX** (*boxtype*) parameter. The parameters **BXO** (*boxoutlinecolor*), **BXM** (*boxmargin*), and **LT** (*linethickness*) are used if the *boxtype* is **HOLLOW** or **FILLED**. The parameter **BXF** (*boxfillcolor*) is used only if the *boxtype* is **FILLED**. The default *boxtype*, **NOBOX**, ignores all other *box* parameters.

The **S** (*scope*) parameter specifies the text scope. **GLOBAL** scope is the same as selecting the check box Show in “Like” Frames in the Text Options dialog. See Section 18.1.6.3, “Specifying the Scope of the Text,” for details.

You may also use the **ZN** (*zone*) parameter to attach text to a specific zone or XY mapping. For further information, see Section 16.1.6.4, "Attaching Text to Zones or X-Y Mapping."

5.1.3.1. Examples of Text Records. You may attach a macro command to the text with the **MFC** parameter. See Section 18.5, “Linking Text and Geometries to Macros.”

Some simple examples of text records are shown below. The first text record specifies only the origin and the text. The next text record specifies the origin, color, font, and the text. The last text record specifies the origin, height, box attributes, and text. Note that the control line for the text can span multiple file lines if necessary (as in the last text record below).

```
TEXT X=50, Y=50, T="Example Text"

TEXT X=10, Y=10, F=TIMES-BOLD, C=BLUE, T="Blue Text"

TEXT X=25, Y=90, CS=FRAME, HU=POINT, H=14,
    BX=FILLED, BXF=YELLOW, BXO=BLACK, LS=1.5,
    T="Box Text \\n Multi-lined text"
```

5.1.4. Geometry Record

Geometry records are used to import geometries from a data file. Geometries are line drawings that may be boundaries, arrows, or even representations of physical structures. You may create data files containing only geometry and text records and read them into Tecplot. You may delete and edit geometries originating from data files just like the geometries that you create interactively.

The geometry record control line begins with the keyword **GEOMETRY**. Use the **CS** (*coordinatesys*) parameter to specify the geometry coordinate system, either **FRAME** or **GRID**. If you specify the frame coordinate system (the default), the values of the **X** (*xorigin*) and **Y** (*yorigin*) parameters are in frame units; if you specify grid coordinates, **X** and **Y** are in grid units (that is, units of the physical coordinate system). **X** and **Y** locate the anchor point, or origin, of the geometry, which is the center of a circle or ellipse, the lower left corner of a square or rectangle, and the anchor point of a polyline. The anchor point specifies the offset of all the points: if *X=1*, *Y=1*, and the first point is (1, 2), and the second point is (2, 4), then Tecplot draws at (2, 3) (1+1, 2+1) then (3, 5) (2+1, 4+1). In other words, the points for any geometry are always relative to the specified anchor point. The **Z** (*zorigin*) is specified only for **LINE3D** geometries, and, since **LINE3D** geometries are always in grid mode, **Z** is always in units of the Z-axis.

Geometry types are selected with the **T** (*geomtype*) parameter. The available geometry types are listed below:

- **SQUARE**: A square with lower left corner at **X**, **Y**.
- **RECTANGLE**: A rectangle with lower left corner at **X**, **Y**.
- **CIRCLE**: A circle centered at **X**, **Y**.
- **ELLIPSE**: An ellipse centered at **X**, **Y**.
- **LINE**: A set of 2-D polylines (referred to as multi-polylines) anchored at **X**, **Y**.
- **LINE3D**: A set of 3-D polylines (referred to as multi-polylines) anchored at **X**, **Y**, **Z**.

The color of the geometry is controlled by the **C** (*color*) parameter. Any geometry type except **LINE3D** may be filled with a color by using the **FC** (*fillcolor*) parameter. With both **C** (*color*) and **FC** (*fillcolor*) on the control line, the geometry is outlined in one color and filled with another. Each polyline of a **LINE** geometry is filled individually (by connecting the last point of the polyline with the first). Not specifying the **FC** (*fillcolor*) parameter results in a hollow, or outlined, geometry drawn in the color of the **C** (*color*) parameter.

You can control how geometries are drawn using the **L** (*linetype*), **LT** (*linethickness*), and **PL** (*patternlength*) parameters. You can set **L** to any of Tecplot's line patterns (**SOLID**, **DASHED**, **DOTTED**, **DASHDOT**, **LONGDASH**, **DASHDOTDOT**). You can set **LT** and **PL** to any value, using frame units.

The control line of the geometry is followed by geometry data. For **SQUARE**, the geometry data consists of just one number: the side length of the square.

For **RECTANGLE**, the geometry data consists of two numbers: the first is the width (horizontal axis dimension), and the second is the height (vertical axis dimension).

For **CIRCLE**, the geometry data is one number: the radius. For **ELLIPSE**, the geometry data consists of two numbers: the first is the horizontal axis length and the second is the vertical axis length. For both circles and ellipses, you can use the **EP** (*numellipsepts*) parameter to specify the number of points used to draw circles and ellipses. All computer-generated curves are simply collections of very short line segments; the **EP** parameter allows you to control how many line segments Tecplot uses to approximate circles and ellipses. The default is 72.

For **LINE** and **LINE3D** geometries, the geometry data is controlled by the **F** (*format*) parameter. These geometries may be specified in either **POINT** or **BLOCK** format. By default, **POINT** format is assumed. Each geometry is specified by the total number of polylines, up to a maximum of 50. Each polyline is defined by a number of points and a series of XY- or XYZ-coordinate points between which the line segments are drawn. In **POINT** format, the XY- or XYZ-coordinates are given together for each point. In **BLOCK** format, all the X-values are listed, then all the Y-values, and (for **LINE3D** geometries) all the Z-values. All coordinates are relative to the **X**, **Y**, and **Z** specified on the control line. You can specify points in either single or double precision by setting the **DT** (*datatype*) parameter to either **SINGLE** or **DOUBLE**.

For **LINE** geometries, you can specify arrowheads using the **AAT** (*arrowheadattach*), **AST** (*arrowheadstyle*), **ASZ** (*arrowheadsize*), and **AAN** (*arrowheadangle*) parameters. See Section 5.1.7, "Summary of Data File Records," for details. These parameters provide the same functionality available when you create a line geometry interactively.

The **S** (*scope*) parameter specifies the geometry's scope. **GLOBAL** scope is the same as selecting the check box Show in Like Frames in the Geometry dialog. See Section 18.2.2.6, "Specifying Geometry Scope," for details.

You may also use the **ZN** (*zone*) parameter to attach geometry to a specific zone or XY-mapping.

You may attach a macro command to the text with the **MFC** parameter. See Section 18.5, "Linking Text and Geometries to Macros."

LINE3D geometries must be created in a data file. They may not be created interactively. **LINE3D** geometries are always in grid mode. To view **LINE3D** geometries in Tecplot, you must be in 3D frame mode, which requires at least one zone. Thus, a data file with only **LINE3D** geometries is useful only as a supplement to other data files.

5.1.4.1. Examples of Geometry Records. The following geometry record defines a rectangle of **40** width and **30** height:

```
GEOMETRY  T=RECTANGLE
40 30
```

The following geometry record defines an origin and a red circle of **20** radius, with an origin of (**75**, **75**) that is filled with blue:

```
GEOMETRY X=75, Y=75, T=CIRCLE, C=RED, FC=BLUE,CS=FRAME
 20
```

The following geometry record defines an origin and two polylines, drawn using the Custom 3 color. The first polyline is composed of three points, the second of two points.

```
GEOMETRY X=50, Y=50, T=LINE, C=CUST3
 2
 3
 0 1
 0 0
 2 0
 2
 0 0
 1 2
```

In **BLOCK** format, the same geometry looks like the following:

```
GEOMETRY X=50, Y=50, T=LINE, C=CUST3, F=BLOCK, CS=FRAME
 2
 3
 0 0 2
 1 0 0
 2
 0 1
 0 2
```

The next geometry record defines a purple ellipse with a horizontal axis length of **20** and a vertical axis length of **10**, with an origin of (**10, 70**), that is filled with yellow.

```
GEOMETRY X=10, Y=70, T=ELLIPSE, C=PURPLE, FC=YELLOW
 20 10
```

The final geometry record is a 3-D polyline with four points that is composed of one polyline using the default origin of (**0, 0, 0**):

```
GEOMETRY T=LINE3D
 1
 4
 0 0 0
 1 2 2
 3 2 3
 4 1 2
```

In **BLOCK** format, this geometry record can be written as follows:

```
GEOMETRY T=LINE3D, F=BLOCK
 1
 4
 0 1 3 4
 0 2 2 1
 0 2 3 2
```

5.1.5. A More Extensive Example of a Geometry Record

In the **TextGeom** file shown below, there are four text records (showing a circle, ellipse, rectangle, and line). A plot of the file is shown in Figure 5-2.

```
TEXT X=20, Y=85, F=HELV-BOLD, C=BLUE, H=7.5,
      T="Example Text"
TEXT X=20, Y=75, F=TIMES-BOLD, H=5, T="Subtitle"
TEXT X=80, Y=25, F=TIMES-ITALIC-BOLD, H=4, C=RED,
      BX=FILLED, BXF=YELLOW, BXM=50, BXO=CYAN,
      T="Filled Box"
TEXT X=41, Y=8, H=4, F=COURIER-BOLD,
      C=CUST3, BX=HOLLOW, BXO=CUST4, T="Hollow Box"
GEOMETRY X=50, Y=50, T=RECTANGLE, FC=WHITE, C=BLUE
   40 30
GEOMETRY X=30, Y=30, T=CIRCLE, FC=BLUE, C=GREEN
   20
GEOMETRY X=70, Y=65, T=LINE, FC=PURPLE, C=BLACK
   1
   4
   -10 0
   0 10
   010 10
   10 0.6
GEOMETRY T=LINE, C=CUST1
   2
   3
   5 50
```

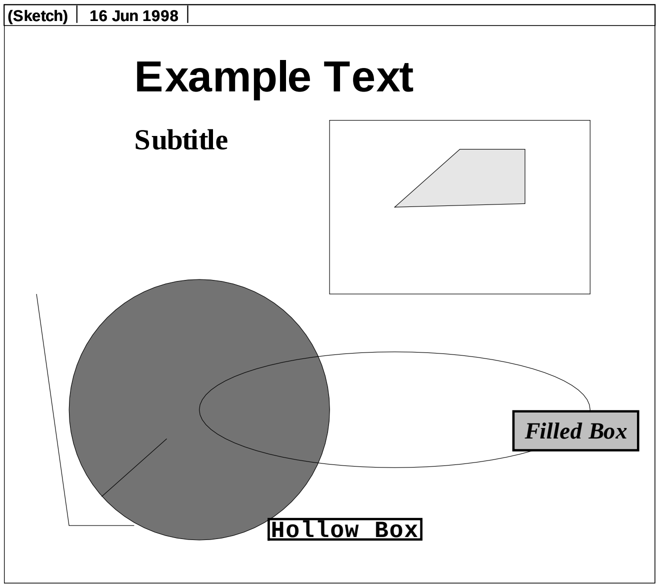

Figure 5-2. Text and geometries created from the sample in Section 5.1.5, “A More Extensive Example of a Geometry Record.”

```
    10 10
    20 10
    2
    15 15
    25 25
GEOMETRY X=60, Y=30, T=ELLIPSE, C=CUST8
    30 10
```

5.1.6. Custom Label Record

The custom label record is an optional record to define sets of text strings for use in custom labeling the values of an axis, contour legend or value labels, or variable-value node labels. The custom-label record begins with the keyword **`CUSTOMLABELS`**, followed by one or more text strings. The text strings must be enclosed within double quotes (“**`"`**”) if they contain any commas, spaces or other special characters, or if they might be confused with valid data file keywords. Enclosing the strings in double quotes is always recommended.

The first custom-label string corresponds to a value of one on the axis, the next to a value of two, the next to a value of three, and so forth. Custom labels may appear one to a line, or there may be more than one on a line, separated by a comma or space. Multiple custom-label records can be present in a data file. If this is the case, you choose which set to assign to a given axis, contour legend, or variable-value node labels. Custom labels are discussed in more detail in Section 17.5.2, “Controlling Tick Mark Labels.”

A simple example of a custom-label record is shown below. **MON** corresponds to a value of **1**, **TUE** corresponds to **2**, **WED** to **3**, **THU** to **4**, and **FRI** to **5**. Since custom labels have a wrap-around effect, **MON** also corresponds to the values **6**, **11**, and so forth.

```
CUSTOMLABELS "MON", "TUE", "WED", "THU", "FRI"
```

5.1.7. Summary of Data File Records

The following table summarizes the records and parameters allowable in Tecplot data files.

Data File Section	Records	Parameter Descriptions
File Header	**TITLE** = "*filetitle*" **VARIABLES** = "*vname1*" "*vname2*" ...	Title for data file. Name of first variable. Name of second variable.
Ordered Zone Record	**ZONE** **T**="*zonetitle*" **I**=*IMax* **J**=*JMax* **K**=*KMax* **C**=*color* **F**=*orderedformat* **D**=(*duplist*) **DT**=(*datatypelist*)	 Title for zone. Number of points in I-direction. Number of points in J-direction. Number of points in K-direction. One of the following: **BLACK**, **RED**, **GREEN**, **BLUE**, **CYAN**, **YELLOW**, **PURPLE**, **WHITE**, **CUST1**, **. . .**, **CUST8**. Either **POINT** or **BLOCK**. List of variables to duplicate from previous zone. List specifying data type for each variable, from among the following: **SINGLE**, **DOUBLE**, **LONGINT**, **SHORTINT**, **BYTE**, **BIT**.

Data File Section	Records	Parameter Descriptions
Finite-element Zone Record	**ZONE**	
	T="*zonetitle***"**	Title for zone.
	N=*numnodes*	Number of nodes.
	E=*numelements*	Number of elements.
	ET=*elementtype*	One of the following: **TRIANGLE**, **QUADRILATERAL**, **TETRAHEDRON**, **BRICK**..
	C=*color*	See description above.
	F=*feformat*	Either **FEPOINT** or **FEBLOCK**.
	D=(*feduplist***)**	List of variables to duplicate from previous zone, and/or the keyword **FECONNECT**.
	NV=*nodevariable*	Which variable represents the Node value.
	DT=(*datatypelist***)**	See description above.

Data File Section	Records	Parameter Descriptions
Text Record	**TEXT**	
	X=*xorigin*	*X* origin of object in *coordinatesys* units.
	Y=*yorigin*	*Y* origin of object in *coordinatesys* units.
	F=*font*	One of the following: **HELV**, **HELV-BOLD**, **TIMES**, **TIMES-ITALIC**, **TIMES-BOLD**, **TIMES-ITALIC-BOLD**, **COURIER**, **COURIER-BOLD**, **GREEK**, **MATH**, **USER-DEF**.
	CS=*coordinatesys*	Either **FRAME** or **GRID**.
	HU=*heightunits*	In **FRAME** *coordinatesys*, either **FRAME** or **POINT**; in **GRID** *coordinatesys*, either **GRID** or **FRAME**.
	AN=*textanchor*	One of the following: **LEFT**, **CENTER**, **RIGHT**, **MIDLEFT**, **MIDCENTER**, **MIDRIGHT**, **HEADLEFT**, **HEADCENTER**, **HEADRIGHT**.
	C=*color*	See description above.
	A=*angle*	Angle in degrees, counter-clockwise from horizontal.
	H=*height*	Character height in *heightunits*.
	LS=*linespacing*	Line spacing for multiple-line text.
	S=*scope*	Either **LOCAL** or **GLOBAL**.
	T="*text*"	Alphanumeric text string.
	BX=*boxtype*	One of **NOBOX**, **HOLLOW**, or **FILLED**.
	BXM=*boxmargin*	Margin around text as fraction of text height.
	BXF=*boxfillcolor*	Fill color for box; use *color* options.
	BXO=*boxcolor*	Color of text box outline; use *color* options.
	LT=*boxlinethickness*	Line thickness of text box.
	ZN=*zone*	Zone (or XY-mapping) number to which this item is assigned.
	MFC="*macrofunctioncommand*"	Macro function command.

Data File Section	Records	Parameter Descriptions
Geometry Record	**GEOMETRY**	
	X=*xorigin*	*X*-origin of object in *coordinatesys* units.
	Y=*yorigin*	*Y*-origin of object in *coordinatesys* units.
	Z=*zorigin*	*Z*-origin of object in *coordinatesys* units.
	CS=*coordinatesys*	Either **FRAME** or **GRID**.
	C=*color*	See description above.
	L=*linetype*	One of the following: **SOLID**, **DASHED**, **DASHDOT**, **DOTTED**, **LONGDASH**, **DASHDOTDOT**.
	PL=*patternlength*	Pattern length for specified line type.
	LT=*linethickness*	Line thickness for geometry outline.
	T=*geomtype*	One of the following: **LINE**, **SQUARE**, **RECTANGLE**, **CIRCLE**, **ELLIPSE**, **LINE3D**.
	EP=*numellipsepts*	Number of points to use to approximate circles and ellipses.
	AST=*arrowheadstyle*	One of **PLAIN**, **HOLLOW**, or **FILLED**.
	AAT=*arrowheadattach*	One of the following: **NONE**, **BEGINNING**, **END**, **BOTH**.
	ASZ=*arrowheadsize*	Size of arrowhead in Frame units.
	AAN=*arrowheadangle*	Angle of arrowhead in degrees.
	DT=*datatype*	Either **SINGLE** or **DOUBLE** (applies to 2- and 3-D polylines only).
	S=*scope*	Either **LOCAL** or **GLOBAL**.
	F=*geomformat*	Either **POINT** or **BLOCK**.
	FC=*geomfillcolor*	Fill color for geometry; use *color* options
	ZN=*zone*	Zone (or XY-mapping) number to which this geometry is assigned.
	MFC="*macrofunctioncommand***"**	Macro function command.
Custom Labels Record	**CUSTOMLABELS**	
	"*label1*"	String for value of one when using custom labels.
	"*label2*" ...	String for value of two when using custom labels.

5.2. Ordered Data

For ordered data, the numerical values in the zone data must be in either **POINT** or **BLOCK** format, determined by the **F** (*format*) parameter.

5.2.1. I-Ordered Data

I-ordered data has only one index, the I-index. This type of data is typically used for XY-plots, scatter plots, and irregular (random) data for triangulation or for interpolation into an IJ-or IJK-ordered zone within Tecplot.

In I-ordered data, the I-index varies from one to *IMax*. The total number of data points is *IMax*. The total number of values in the zone data is *IMax*N* (where *N* is the number of 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).

5.2.1.1. Example of I-Ordered Data in POINT Format. A simple example of I-ordered data in **POINT** format is listed below. There are two variables (**X**, **Y**) and five data points. In this example, each row of data corresponds to a data point and each column to a variable. This data set is plotted in Figure 5-3; each data point is labeled with its I-index.

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

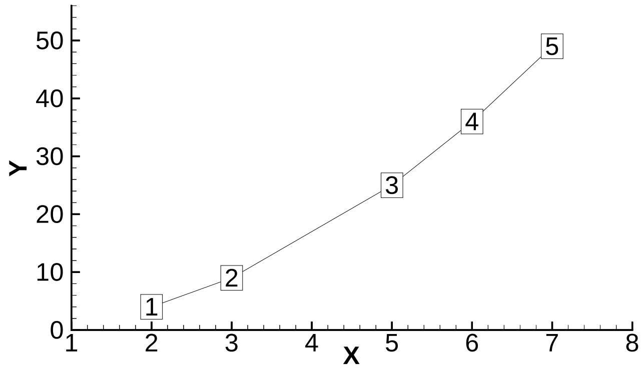

Figure 5-3. An I-ordered data set.

5.2.1.2. FORTRAN Code Example to Generate I-Ordered Data in POINT Format. The following sample FORTRAN code shows how to create I-ordered data in **POINT** format:

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

5.2.1.3. Example of I-Ordered Data in BLOCK Format. The same data as in Section 5.2.1.1. is shown below in **BLOCK** format. In this example, each column of zone data corresponds to a data point; each row to a variable.

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

In **BLOCK** format all *IMax* values of each variable are listed, one variable at a time.

5.2.1.4. Example of FORTRAN Code to Generate I-Ordered Data in BLOCK Format. The following sample FORTRAN code shows how to create I-ordered data in **BLOCK** format:

```
    INTEGER VAR
      .
      .
      .
    WRITE (*,*) ´ZONE F=BLOCK, I=´, IMAX
    DO 1 VAR=1,NUMVAR
        DO 1 I=1,IMAX
  1         WRITE (*,*) ARRAY(VAR,I)
```

5.2.1.5. Example: A Multi-zone XY-Plot. The two tables below show the values of pressure and temperature measured at four locations on some object at two different times. The four locations are different for each time measurement.

Time = 0.0 seconds:		
Position	**Temperature**	**Pressure**
71.30	563.7	101362.5
86.70	556.7	101349.6
103.1	540.8	101345.4
124.4	449.2	101345.2

Time = 0.1 seconds:		
Position	**Temperature**	**Pressure**
71.31	564.9	101362.1
84.42	553.1	101348.9
103.1	540.5	101344.0
124.8	458.5	101342.2

For this case, we want to set up two zones in the data file, one for each time value. Each zone has three variables (**`Position`**, **`Temperature`**, and **`Pressure`**) and four data points (one for each location). This means that *IMax=4* for each zone. We include a text record (discussed in Section 5.1.3., "Text Record") to add a title to the plot. A data file in **`POINT`** format is given below. The plot shown in Figure 5-4 can be produced from this file.

```
TITLE = "Example: Multi-zone XY-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=GRID, HU=GRID, X=0.36, Y=0.87, H=0.04, 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-Plot"
VARIABLES = "Position", "Temperature", "Pressure"
ZONE F=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
```

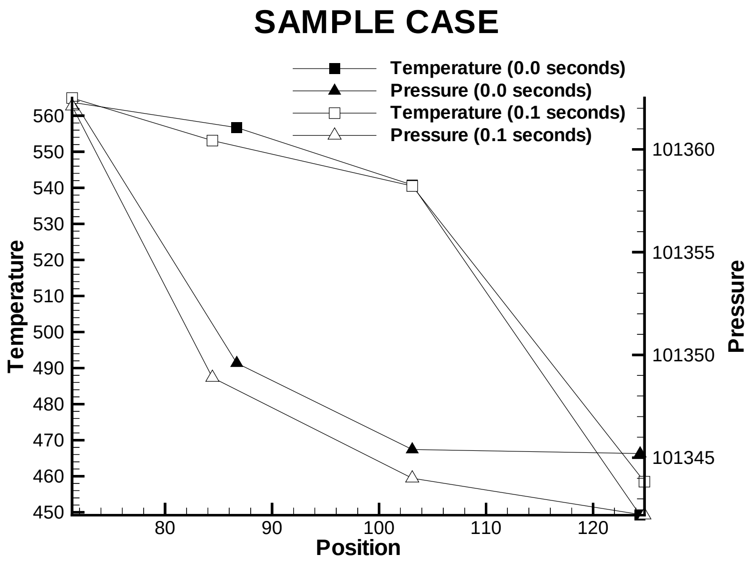

Figure 5-4. A multi-zone XY-plot.

```
ZONE F=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=GRID, HU=GRID, X=0.36, Y=0.87, H=0.04, 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=GRID, HU=GRID, X=0.36, Y=0.87, H=0.04, T="SAMPLE CASE"
```

5.2.2. IJ-Ordered Data

IJ-ordered data has two indices: I and J. IJ-ordered data is typically used for 2- and 3-D surface mesh, contour, vector, and shade plots, but it can also be used to plot families of lines in XY-plots. See Chapter 8, “XY-Plots,” for more information. 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 is *IMax*JMax*. The total number of numerical values in the zone data is *IMax*JMax*N* (where *N* is the number of 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. 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*.

5.2.2.1. Example of IJ-Ordered Data in POINT Format. A simple example of IJ-ordered data in **`POINT`** format is listed below. There are four variables (**`X`**, **`Y`**, **`Temperature`**, **`Pressure`**) and six data points. In this example, each row of data corresponds to a data point; each column to a variable. The first two lines are for *J=1*, the next two for *J=2*, the last two for *J=3*. The first, third, and fifth lines are for *I=1*; the second, fourth, and sixth lines are for *I=2*. This data is plotted in Figure 5-5; each data point is labeled with its IJ-index.

```
VARIABLES = "X", "Y", "Temperature", "Pressure"
ZONE I=2, J=3, F=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
```

5.2.2.2. Example of FORTRAN Code to Generate IJ-Ordered Data in POINT Format. The following sample FORTRAN code shows how to create IJ-ordered data in **`POINT`** format:

```
WRITE (*,*) ´VARIABLES = "X", "Y", "Temperature", "Pressure"’
WRITE (*,*) ´ZONE I=’, IMAX, ’, J=’, JMAX, ’, ’F=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)
```

5.2.2.3. Example of IJ-Ordered Data Set in BLOCK Format. The same data set as in Section 5.2.2.1. is shown in **`BLOCK`** format below. In this example, each column of data corresponds to a data point; each row to a variable.

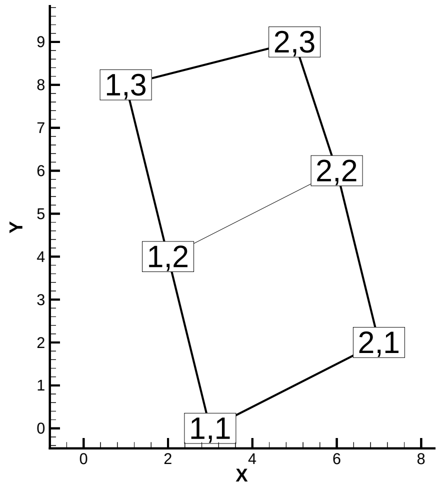

Figure 5-5. An IJ-ordered data set.

```
VARIABLES = "X", "Y", "Temperature", "Pressure"
ZONE I=2, J=3, F=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
```

In **BLOCK** format, all *IMax*JMax* values of each variable are listed, one variable at a time. Within each variable block, all the values of a variable at each data point are listed.

5.2.2.4. Example FORTRAN Code to Generate IJ-Ordered Data in BLOCK Format.

The following sample FORTRAN code shows how to create IJ-ordered data in **BLOCK** format:

```
 INTEGER VAR
   .
   .
   .
 WRITE (*,*) ´ZONE F=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)
```

5.2.3. 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. See Chapter 21, “Working with 3-D Volume Data,” for more information.

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 is *IMax*JMax*KMax*. The total number of values in the zone data is *IMax*JMax*KMax*N*, where *N* is the number of 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. That is, 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*.

5.2.3.1. An Example of IJK-Ordered Data in POINT Format.

A simple example of IJK-ordered data in **POINT** format is listed below. There are four variables (**X**, **Y**, **Z**, **Temperature**) and twelve data points. For this example, each row of data corresponds to a data point; each column to a variable. This data is plotted in Figure 5-6; each data point is labeled with its IJK-index.

```
VARIABLES = "X" "Y" "Z" "Temp"
ZONE I=3, J=2, K=2,F=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
```

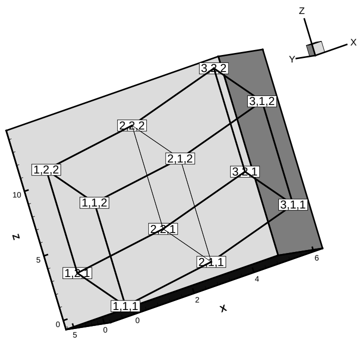

Figure 5-6. An IJK-ordered data set.

5.2.3.2. The Same Data in BLOCK Format. The same data set as Section 5.2.3.1., this time in `BLOCK` format, is shown below. For this example, each column of data corresponds to a data point; each row to a variable.

```
VARIABLES = "X" "Y" "Z" "Temp"
ZONE I=3, J=2, K=2, F=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
```

5.2.3.3. An Example of FORTRAN Code to Generate an IJK-Ordered Zone in POINT Format. The following sample FORTRAN code shows how to create an IJK-ordered zone in `POINT` format:

```
 WRITE (*,*) ’VARIABLES = "X", "Y", "Z", "Temp"’

 WRITE (*,*) ’ZONE I=’,IMAX,’ J=’,JMAX,’ K=’,KMAX,’ F=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)
```

In `BLOCK` format, all *IMax*JMax*KMax* values of each variable are listed, one variable at a time. Within each variable block, all the values of the variable at each data point are listed.

5.2.3.4. An Example of FORTRAN Code to Generate IJK-Ordered Data in BLOCK Format. The following sample FORTRAN code shows how to create an IJK-ordered zone in `BLOCK` format:

```
 INTEGER VAR
   .
   .
   .
   .
 WRITE (*,*) ´ZONE F=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)
```

5.2.4. One Variable Data Files

For ordered data, it is possible to read in a data file that has only one variable. Tecplot then creates the other required variables. That is, if your data is I-ordered, a variable containing the I-index values is created, numbered **`V1`**, and called "**`I`**". For IJ-ordered data, two variables, numbered **`V1`** and **`V2`** and called "**`I`**" and "**`J`**," are created to contain the I- and J-index values. For IJK-ordered data, three variables "**`I`**", "**`J`**", and "**`K`**" are created and numbered **`V1`**, **`V2`**, and **`V3`**. The variable in the data file is numbered with the next available variable number, that is, **`V2`** for I-ordered data, **`V3`** for IJ-ordered data, and **`V4`** for IJK-ordered data. The created variables are the default X-, Y-, and Z-variables. The data type for the created variables is determined according to the following table:

Maximum of IMax, JMax, and KMax	**Data Type**
< 256	**`BYTE`**
<32,766	**`SHORTINT`**
>=32,766	**`SINGLE`**

For example, if you have an ASCII file with 256 by 384 numbers representing intensities of a rasterized image, you could make a data file similar to the following:

```
VARIABLES = "TEMPERATURE"
 ZONE I=256, J=384
```
List all 98,304 values of temperature here.

Read the data file into Tecplot. Two new variables of type **`SHORTINT`** are created and used as the default X- and Y-coordinates. These variables are the I- and J-index values; they are named "**`I`**" and "**`J`**." You can now create any type of 2-D plot with the data.

If you have finite-element data, Tecplot will not create any new variables for you. If you need to add variables to finite-element data, you can do so using the Data menu.

5.3. Finite-Element Data

For finite-element data, the numerical values in the zone data must be in either **`FEPOINT`** or **`FEBLOCK`** format as specified by the **`F`** (*format*) parameter. 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). The element type (triangle, quadrilateral, tetrahedron or brick) is specified using the **`ET`** parameter. The zone data is divided into two logical sections (without any markers). The first section, the node data, lists the values of the variables at the data points (or nodes) 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 (**ET** parameter). (You may place blank lines between the node data and the connectivity list to help distinguish them.)

In the descriptions below, **N***E* is the *E*th node at a vertex of an element. The subscripts of **N***E* refer to the element number. For example, $\mathbf{N2}_3$ represents the second node of the third element.

For the triangle element type, each line of the connectivity list contains three node numbers that define a triangular element:

$\mathbf{N1}_M$, $\mathbf{N2}_M$, $\mathbf{N3}_M$

For the quadrilateral element type, each line of the connectivity list contains four node numbers that define a quadrilateral element:

$\mathbf{N1}_M$, $\mathbf{N2}_M$, $\mathbf{N3}_M$, $\mathbf{N4}_M$

If you need to mix quadrilateral and triangle elements, either create two zones or use the quadrilateral element type with node numbers ($\mathbf{N4}_M=\mathbf{N3}_M$) repeated to form triangles.

Zones created from the quadrilateral and triangle element types are called FE-surface zones.

For the tetrahedron element type, each line of the second section of the zone data contains four node numbers that define a tetrahedral element:

$\mathbf{N1}_M$ $\mathbf{N2}_M$ $\mathbf{N3}_M$ $\mathbf{N4}_M$

For the brick element type, each line of the second section contains eight node numbers that define a ‘‘brick-like” element:

$\mathbf{N1}_M$ $\mathbf{N2}_M$ $\mathbf{N3}_M$ $\mathbf{N4}_M$ $\mathbf{N5}_M$ $\mathbf{N6}_M$ $\mathbf{N7}_M$ $\mathbf{N8}_M$

Tecplot divides the eight nodes into two groups of four; nodes $\mathbf{N1}_M$, $\mathbf{N2}_M$, $\mathbf{N3}_M$, and $\mathbf{N4}_M$ make up the first group, and $\mathbf{N5}_M$, $\mathbf{N6}_M$, $\mathbf{N7}_M$, and $\mathbf{N8}_M$ make up the second group. Each node is connected to two nodes within its group and the node in the corresponding position in the other group. For example, $\mathbf{N1}_M$ is connected to $\mathbf{N2}_M$ and $\mathbf{N4}_M$ in its own group, and to $\mathbf{N5}_M$ in the second group. To create elements with fewer than eight nodes, repeat nodes as necessary, keeping in mind the basic brick connectivity just described. Figure 5-7 shows the basic brick connectivity. For example, to create a tetrahedron, you can set $\mathbf{N3}_M=\mathbf{N4}_M$ and $\mathbf{N5}_M=\mathbf{N6}_M=\mathbf{N7}_M=\mathbf{N8}_M$. To create a quadrilateral-based pyramid, you can set $\mathbf{N5}_M=\mathbf{N6}_M=\mathbf{N7}_M=\mathbf{N8}_M$. If you need a mixture of bricks and tetrahedra, either use two zones or use the brick element type with node numbers repeated so that tetrahedra result.

Zones created from the brick and tetrahedron element types are called FE-volume zones.

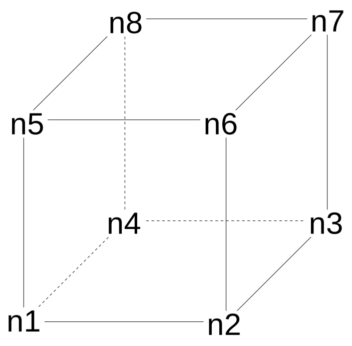

Figure 5-7. Basic brick connectivity.

If the keyword “**FECONNECT**” is specified in the **D** parameter in the zone control line, the connectivity list is duplicated from the previous zone. In this case, no connectivity list is given, just the node data. If you use **FECONNECT** for the first finite-element zone, Tecplot generates an error message.

5.3.1. Example of Triangle Data in FEPOINT Format

A simple example of triangle element type finite-element data in **FEPOINT** format is listed below. There are two variables (**X**, **Y**) and five data points. In this example, each row of the data section corresponds to a node and each column to a variable. Each row of the connectivity list corresponds to a triangular element and each column specifies a node number. This data set is plotted in Figure 5-8. Each data point is labeled with its node number.

```
VARIABLES = "X", "Y"
ZONE N=5, E=3, F=FEPOINT, ET=TRIANGLE
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
```

5.3.1.1. The Same Data File in FEBLOCK Format. The same data in **FEBLOCK** format is shown below. In this example, each column of the data section corresponds to a node

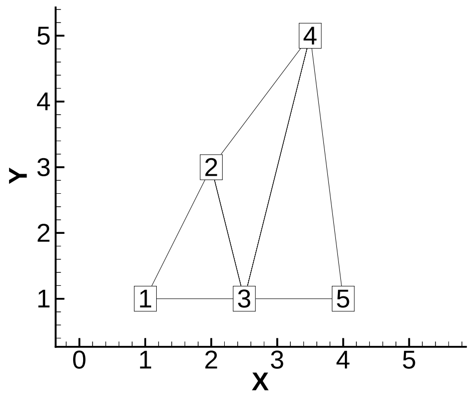

Figure 5-8. A finite-element triangle data set.

and each row to a variable. As above, each row of the connectivity list corresponds to a triangular element and each column specifies a node number.

```
VARIABLES = "X", "Y"
 ZONE N=5, E=3, F=FEBLOCK, ET=TRIANGLE
 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
```

5.3.2. An Example of FORTRAN Code to Generate Triangle Data in FEPOINT Format

The following sample FORTRAN code shows how to create triangle element type finite-element data in **FEPOINT** format:

```
 INTEGER VAR
   .
   .
   .
 WRITE (*,*) ´ZONE F=FEPOINT,ET=TRIANGLE,N=´, NNODES,´,E=´,NELEM
 DO 1 N=1,NNODES
     DO 1 VAR=1,NUMVAR
1        WRITE(*,*) VARRAY(VAR,N)

 DO 2 M=1,NELEM
```

```
     DO 2 L=1,3
2        WRITE (*,*) NDCNCT(M,L)
```

5.3.3. An Example of FORTRAN Code to Generate Triangle Data in FEBLOCK Format

The following sample FORTRAN code shows how to create triangle element type finite-element data in **FEBLOCK** format:

```
 INTEGER VAR
   .
   .
   .
 WRITE (*,*) ´ZONE F=FEBLOCK,ET=TRIANGLE,N=´,NNODES, ´,E=´,NELEM
 DO 1 VAR=1,NUMVAR
     DO 1 N=1,NNODES
1        WRITE(*,*) VARRAY(VAR,N)
 DO 2 M=1,NELEM
     DO 2 L=1,3
2        WRITE (*,*) NDCNCT(M,L)
```

5.3.4. An Example of a Finite-Element Zone Node Variable Parameters

The node variable parameter allows you to set the connectivity to match the value of the selected node variable. In the example below, the files appear to be identical in Tecplot, although the connectivity list has changed to reflect the values of the node variable Node Order. Notice that the index value of the nodes is not changed by the node variable value.

The original data set:

```
TITLE     = "Internally created dataset"
VARIABLES = "X"
"Y"
ZONE T="Triangulation"
 N=6, E=5,F=FEPOINT ET=Triangle
DT=(SINGLE SINGLE )
 2.00E+000 3.00E+000
 2.20E+000 3.10E+000
 3.10E+000 4.20E+000
 2.80E+000 3.50E+000
 2.40E+000 2.10E+000
 4.30E+000 3.20E+000
 1 2 5
 6 4 3
 5 4 6
 2 3 4
 5 2 4
```

The data set with the nodes re-ordered for connectivity:

```
TITLE     = "RE-ordered data"
VARIABLES = "X"
"Y" "Node-Order"
ZONE T="Triangulation"
 N=6, NV = 3, E=5,F=FEPOINT ET=Triangle
DT=(SINGLE SINGLE )
 2.00E+000 3.00E+000 5
 2.20E+000 3.10E+000 4
 3.10E+000 4.20E+000 1
 2.80E+000 3.50E+000 2
 2.40E+000 2.10E+000 6
 4.30E+000 3.20E+000 3
 1 2 3
 4 2 6
 5 4 6
 2 3 6
 1 2 4
```

5.4. Duplicating Variables and Connectivity Lists

The **`D`** parameter in the **`ZONE`** record allows you to duplicate variables or the connectivity list from the previous zone. The following is an example to illustrate this feature.

The table below shows Cartesian coordinates X and Y of six locations, and the pressure measured there at three different times (*P1*, *P2*, *P3*). The XY-locations have been arranged into finite-elements.

X	Y	P_1	P_2	P_3
-1.0	0.0	100	110	120
0.0	0.0	125	135	145
1.0	0.0	150	160	180
-0.5	0.8	150	165	175
0.5	0.8	175	185	195
0.0	1.6	200	200	200

For this case, we want to set up three zones in the data file, one for each time measurement. Each zone has three variables: X, Y, and P. The zones are of the triangle element type, meaning that three nodes must be used to define each element. One way to set up this data file would be to list the complete set of values for X, Y, and P for each zone. Since the X,Y-coordinates are exactly the same for all three zones, a more compact data file can be made by using the duplication list parameter (*D*). In the data file given below, the second and third zones have duplication lists that copy the values of the X- and Y-variables and the connectivity list from the first zone. As a result, the only values listed for the second and third zones are the pressure variable values. A plot of the data is shown in Figure 5-9. Note that the data could easily have been organized in a single zone with five variables. Since blank lines are ignored in the data file, you can embed them to improve readability.

```
TITLE = "Example: Duplicated Variables and Connectivity Lists"
VARIABLES = "X", "Y", "P"
ZONE T="P_1", F=FEPOINT, N=6, E=4, ET=TRIANGLE
-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", F=FEPOINT, N=6, E=4, ET=TRIANGLE, D=(1,2,FECONNECT)
110 135 160 165 185 200

ZONE T="P_3", F=FEPOINT, N=6, E=4, ET=TRIANGLE, D=(1,2,FECONNECT)
120 145 180 175 195 200
```

5.5. Converting ASCII Data Files to Binary

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

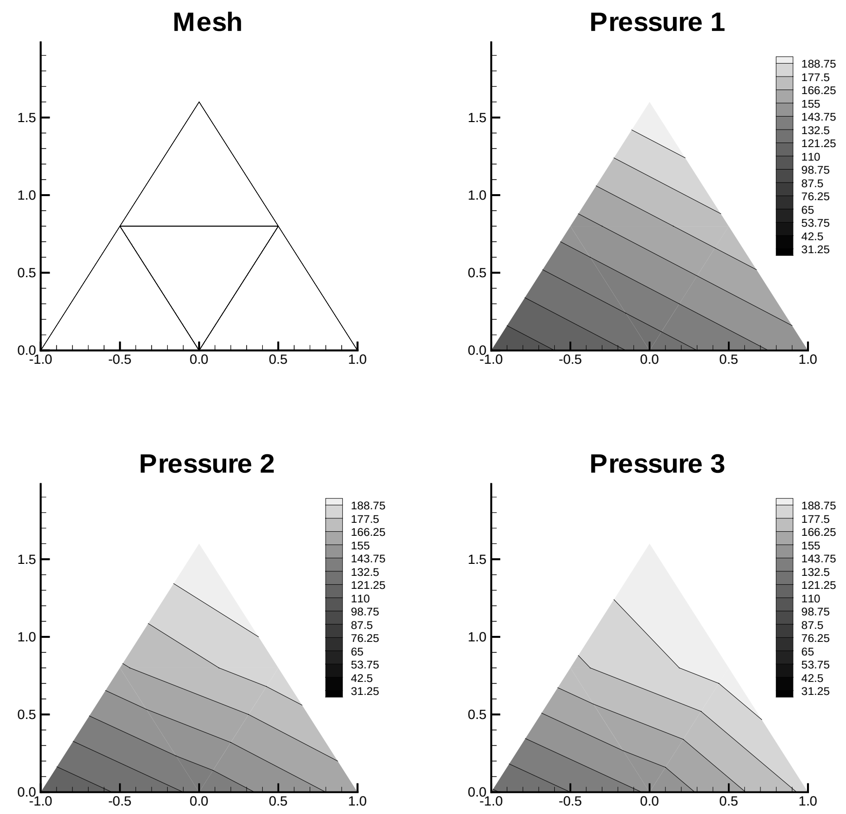

Figure 5-9. Plot of finite-element zones.

5.5.1. Standard 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.

For more Preplot command lines, see Appendix B.3, "Preplot."

5.5.2. Examples of Using Preplot

If you have an ASCII file named **dset.dat**, you can create a binary data file called **dset.plt** with the following Preplot command:

```
preplot dset.dat dset.plt
```

By default, Preplot looks for files with the **.dat** extension, and creates binary files with the **.plt** extension. Thus, either of the following commands is equivalent to the above command:

```
preplot dset
```

```
preplot dset.dat
```

Preplot checks the input ASCII data file for errors such as illegal format, numbers too small or too large, the wrong number of values in a data block, and illegal finite-element node numbers. If Preplot finds an error, it issues a message displaying the line and column where the error was first noticed. This is only an indication of where the error was *detected*; the actual error may be in the preceding columns or lines.

If Preplot encounters an error, you may want to set the debug option to get more information about the events leading up to the error:

```
preplot dset.dat -d
```

You can set the flag to **-d2**, or **-d3**, or **-d4**, and so forth, to obtain even more detailed information.

In the following Preplot command line, the number of points that are written to the binary data file **dset.plt** is less than the number of points in the input file **dset.dat**:

```
preplot dset.dat -iset 3,6,34,2 -jset 3,1,21,1 -iset 4,4,44,5
```

For zone 3, Preplot outputs data points with I-index starting at 6 and ending at 34, skipping every other one, and J-index starting at one and ending at 21. For zone 4, Preplot outputs data points with the I-index starting at four, ending at 44, and skipping by five.

In the following Preplot command line, every other point in the I-, J-, and K-directions is written to the binary data file:

```
preplot dset.dat -iset ,,,2 -jset ,,,2 -kset ,,,2
```

The *zone*, *start*, and *end* parameters are not specified, so all zones are used, starting with index 1, and ending with the maximum index. The overall effect is to reduce the number of a data points by a factor of about eight.

5.5.3. Using Preplot to Convert Files in PLOT3D Format

PLOT3D is a graphics plotting package developed at NASA. Some numerical simulation packages and other programs can create graphics in PLOT3D format. There are two paths by which you can get files in PLOT3D format into Tecplot. This section describes the Preplot path; you can also use the PLOT3D loader described in Section 7.9, “The PLOT3D Data Loader.”

Preplot can read files in the PLOT3D format and convert them to Tecplot binary data files through the use of special switches. You do not need to know about these switches unless you have data in PLOT3D format.

PLOT3D files typically come in pairs consisting of a grid file (with extension **`.g`**) and a solution file (with extension **`.q`**). Sometimes only the grid file is available. The grid itself may be either a single grid, or a multigrid, and the data may be 1D, 2D, 3D-planar, or 3D-whole (equivalent to Tecplot’s 3-D volume data). The PLOT3D files may be binary or ASCII. The PLOT3D-specific switches to Preplot allow you to read PLOT3D files with virtually any combination of these options.

The *ilist*, *jlist*, and *klist* are comma-separated lists of items of the form:

start[:*end*][:*skip*]]

where *start* is the number of the starting I-, J-, or K-plane, *end* is the number of ending I-, J-, or K-plane, and *skip* is the skip factor between planes. If *end* is omitted, it defaults to the starting plane (so if just *start* is specified, only that one plane is included). The *skip* defaults to one (every plane) if omitted; a value of two includes every other plane, a value of three include every third plane, and so on.

You must specify one of the flags **`-1d`**, **`-2d`**, **`-3dp`**, or **`-3dw`**. You may also specify only one of **`-ip`**, **`-jp`**, or **`-kp`** and only one of **`-b`** or **`-f`**.

If the input PLOT3D file is 3-D whole (**`-3dw`**) and none of the plane-extraction switches **`-ip`**, **`-jp`**, or **`-kp`** is specified, the PLOT3D file is converted directly to an IJK-ordered zone (or multiple zones if the file is multi-grid).

For example, in the following command line, Preplot reads from the PLOT3D files **`aero.g`** and **`aero.q`**. The input is binary and 3-D whole. The J-planes 2, 3, 4, 45, 46, and 47 are processed and made into six IJ-ordered zones, in a binary data file named **`aero.plt`**:

```
preplot aero -plot3d -b -3dw -jp 2,3,4,45,46,47
```

In the following command line, the plane-extraction switches are omitted, so Preplot creates a single IJK-ordered zone:

```
preplot aero -plot3d -b -3dw
```

The following command line reads an ASCII file **`airplane.g`** for which there is no corresponding **`.q`** file; the data is 3-D whole:

```
preplot airplane -plot3d -gridonly -3dw
```

The following command line reads a multi-grid, 3-D planar, binary-FORTRAN pair of PLOT3D files, **`multgrid.g`** and **`multgrid.q`**:

```
preplot multgrid -plot3d -m -f -3dp
```