

Goals for This Lecture:

- Learn how to use gnuplot to make simple plots of data from a file.
- Introduce some basic I/O concepts

Using Gnuplot to Plot data from a file

- Gnuplot (pronounced “Ga-new-plot”) is a simple plotting application found on most Linux systems that can be used to make simple plots
- Gnuplot can create a variety of plot types
- Gnuplot can display plots in many ways including
 - on the screen (an X11 window)
 - Default behavior of gnuplot
 - produce images of plots (PNG files)
 - Useful for plots that will appear on a web page
 - printable plots (PostScript files)
 - Useful for producing plots that will be printed or which will appear in printed form

A sample data set

- Stored in file named data1.dat

1.0	1.0	1.1
2.0	4.0	4.4
3.0	9.0	9.9
4.0	5.0	5.5
5.0	4.5	4.9
6.0	4.0	3.2
7.0	3.5	2.9

Plotting data from a file

- The Gnuplot command interpreter can be started by executing the command `gnuplot` at the shell prompt
- An x versus y plot of the data in columns 1 and 2 of the file can be produced by the following Gnuplot command:
`plot "file1.dat"`
- This produces a x-window containing a figure with the data plotted as points
- If you wish to produce a plot that uses lines instead of points issue the following command:
`plot "file1.dat" with lines`
- You can produce a plot with both lines and points using the command:
`plot "file1.dat" with linespoints`
- You can exit the Gnuplot interpreter by executing the command:
`exit`

Specifying the Columns Containing the Data

- Gnuplot allows you to specify which column to use for the x-data and which to use for the y-data
- This is accomplished by the “using” keyword in the plot command.
- For example:
`plot "file1.dat" using 2:3 with lines`
- Uses the data in column 2 for the x-data and the data in column 3 for the y-data

Producing Multi-dataset plots in Gnuplot

- You can produce multi-line plots in Gnuplot by adding more files (separated by commas) as arguments to the plot command

- For example:

```
plot "data1.dat" using 2:3 with lines,  
     "data1.dat" using 2:2 with points,  
     "data1.dat" using 3:3 with linespoints
```

- Gnuplot will automatically generate an entry in the legend, a.k.a. key for each X-Y data set
- If you do not wish to have the key appear it can be turned off with the command:

```
set nokey
```

Logarithmic Plots in Gnuplot

- You can make one or both axes of the plot be log-scaled by using the `logscale` keyword to the `set` command.
- For example:
 - `set logscale y`
 - `plot "data1.dat" using 2:3 with lines`produces a plot with a log-scaled y-axis
- To get a logarithmic x-axis:
 - `set logscale x`
 - `plot "data1.dat" using 2:3 with lines`
- To turn off log-scaling:
 - `set nologscale x`
 - or
 - `set nologscale y`

Controlling the x & y ranges of plots in Gnuplot

- You can control the x & y ranges of your plots by using the `xrange` and `yrange` keywords to `set` command.
- For example:
 - `set xrange [1:10]`
 - `set yrange [5:15]`
 - `plot "data1.dat" using 2:3 with lines`produces a plot with the x-axis running from 1 to 10 and a y-axis from 5 to 15
- These keywords will work even if the axes are logarithmic
- If `xrange` & `yrange` are not set, Gnuplot will scale the plot automatically

Producing plot image files

- You can produce an PNG image of a plot with two additional Gnuplot commands:

```
set terminal png
```

```
set output "plot.png"
```

```
plot "data1.dat" with linespoints
```

- The first command tells Gnuplot that it should produce an image instead of an X11-window
- The second command tells Gnuplot what the image file should be named
- The image file can be viewed using any image file viewer
- Eye-of-gnome (eog) works well:
>eog plot.png

Producing Postscript files

- You can produce a Postscript file containing a plot that can be printed on any Postscript compatible printer:

```
set terminal postscript
```

```
set output "plot.ps"
```

```
plot "data1.dat" with linespoints
```

- The first command tells Gnuplot that it should produce a Postscript file instead of an X11-window
- The second command tells Gnuplot what the postscript file should be named
- The postscript file can be viewed using any postscript file viewer
- ggv (Gnu Ghostview):
>ggv plot.ps

Labeling Gnuplot-produced Plots

- You can set the x & y labels on your plots by issuing the commands:

```
set xlabel "position (meters)"
```

```
set ylabel "velocity (meters/s)"
```

```
plot "data1.dat" with linespoints
```

- The xlabel command sets the label along the x-axis
- The ylabel command sets the label along the y-axis
- In an X11 window the ylabel appears horizontally in the upper-left corner of the plot
- In an PNG or Postscript plot the y-label may appear vertically along the y-axis

Gnuplot Documentation

- Gnuplot is well documented
- If you want to know how to do something:
RTFM!
- See the Gnuplot home page linked to from the course web page
- Also look at the Gnuplot tutorials linked to from the course web page
- Best way to learn how to do something is to look at the examples

Formatted WRITE & READ statements

- Thus far we have encountered list-directed write & read statements
- This is indicated by the second “*” argument to the WRITE or READ statement
- These statements are called free-format
- The formatting of the output or input is determined by the data type of the listed items in the WRITE or READ statement
- We will now examine how to control the formatting of the WRITE or READ operation using FORMATS

Formats in WRITE statements

- The second asterisk in the WRITE statement can be replaced by a FORMAT string that specifies how the data should be output

- Example:

```
write(*,'(1x,i4,2x,f10.2)') i,x_position
```

- This format specifies that the output should skip 1 space from the beginning of the line (1x), should output the integer i using an integer four digit field (i4), skip 2 spaces (2x), and output the real variable x_position as a decimal 10 digit field with 2 digits to the right of the decimal place(F10.2)

- This format string could be placed in a FORMAT statement that is labeled with a numerical value from 1 to 5 digits

- Example:

```
write(*,1000) i,x_position  
1000 format(1x,i4,2x,f10.2)
```

- The format string could be placed in character variable

- Example:

```
character(len=30) fmt1='(1x,i4,2x,f10.2)'  
write(*,fmt1) i,x_position
```

Format Descriptors in WRITE statements

- The items in the FORMAT string are called format descriptors
- Format descriptors control the position and manner in which data is displayed
- Format descriptors fall into four categories:
 - Those that control the vertical positioning of the data
 - Those that control the horizontal positioning of the data
 - Those that control the output format of the data
 - Those that control the repetition of portions of the format

Reading Assignment

- Read Sections 5.1-5.3