SKILLPILLS

Skill Pill: Introduction to gnuplot

Lecture 1: It's all gnu to you

James Schloss

Okinawa Institute of Science and Technology
james.schloss@oist.jp

-66 (Trump Calendar)

OIST

Overview

SKILLPILLS

What is gnuplot good for?

- If you have data and want to plot it, Gnuplot is a standalone plotting tool that is great at making great plots

What is gnuplot good for?

- If you have data and want to plot it, Gnuplot is a standalone plotting tool that is great at making great plots
- If you are using matlab, python, or julia, you already have plotting tools at your disposal. Gnuplot is still worth using in these cases.

What is gnuplot good for?

- If you have data and want to plot it, Gnuplot is a standalone plotting tool that is great at making great plots
- If you are using matlab, python, or julia, you already have plotting tools at your disposal. Gnuplot is still worth using in these cases.
- If you are using C/C++ (CUDA), FORTRAN, or another language where plotting is not trivial, gnuplot is your best choice at creating plots

What is gnuplot good for?

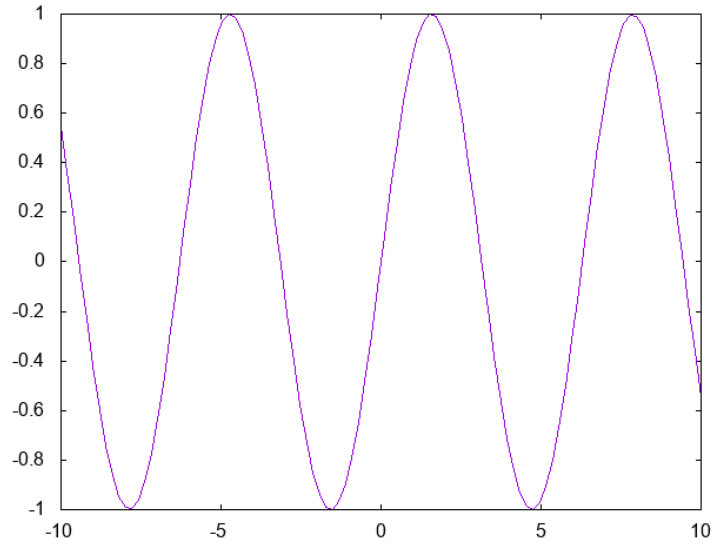
SKILLPILLS

- If you have data and want to plot it, Gnuplot is a standalone plotting tool that is great at making great plots
- If you are using matlab, python, or julia, you already have plotting tools at your disposal. Gnuplot is still worth using in these cases.
- If you are using C/C++ (CUDA), FORTRAN, or another language where plotting is not trivial, gnuplot is your best choice at creating plots
- Gnuplot makes plotting easy, so long as you have formatted the data appropriately.

How to plot

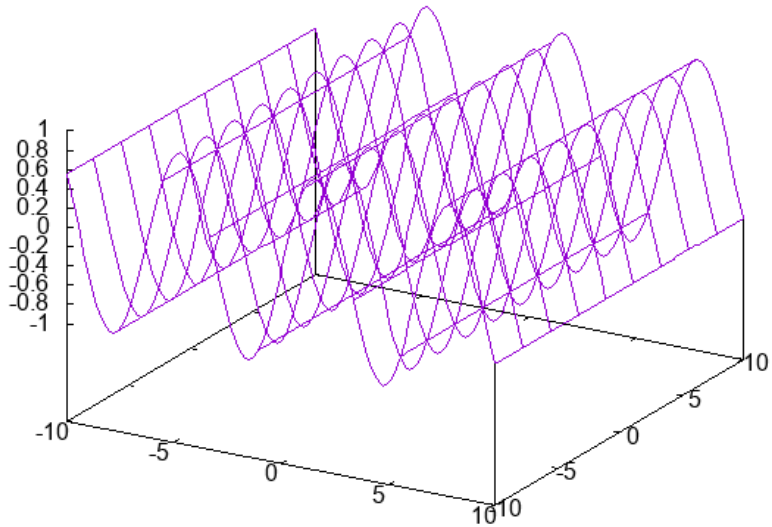
- For now, let's use gnuplot's inbuilt math libraries.
- Type

```
plot sin(x)
```



- Now type

```
splot sin(x)
```



Plotting basics

- You can run commands

```
!ls
```

- **TITLES**

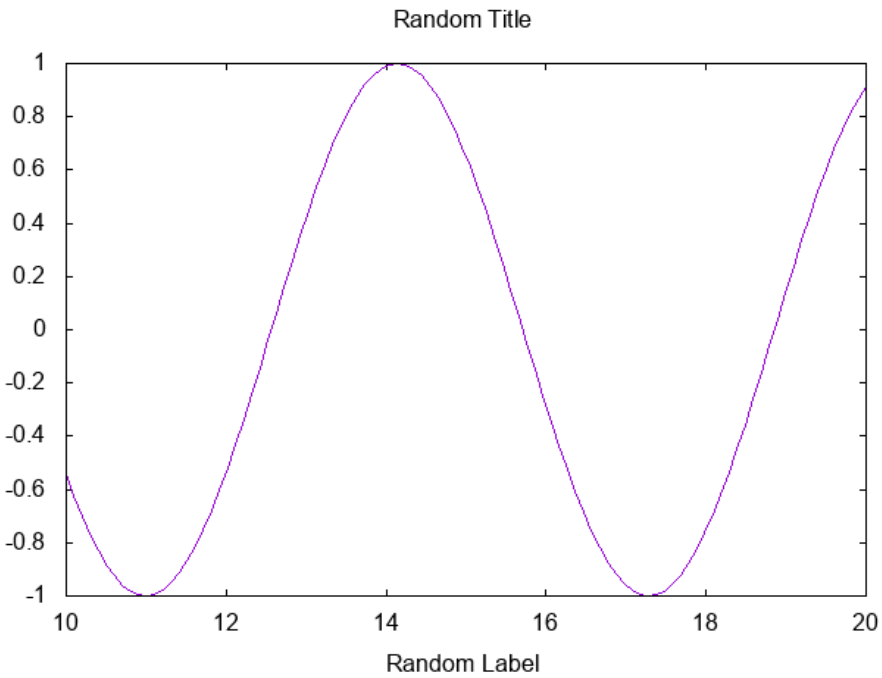
```
set title "Random title"
```

- **LABELS**

```
set xlabel "Random Label"
    font
set label "Totally useful
    label" at 0, 10
```

- **RANGES**

```
set xrange [10:20]
```



- **KEYS**

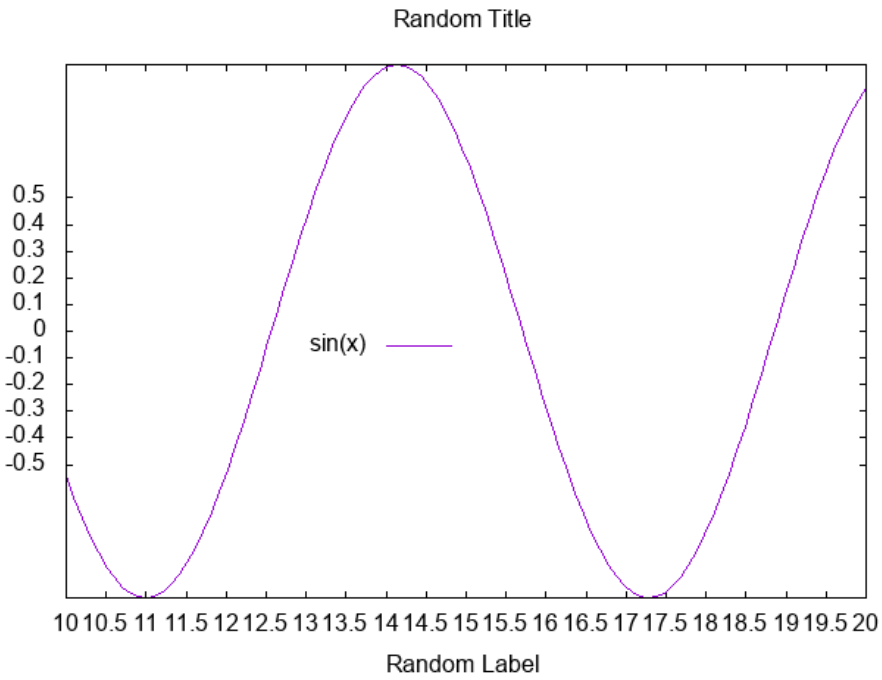
```
set key at 15,0
```

- **LOGSCALE** – Note, does not work well with sin(x)

```
set logscale y
```

- **TICS**

```
set xtics 0.5
set ytics -0.5,0.1,0.5
```



Plotting basics

- **SAVE**

```
save "setup.gp"
```

- **UNSET**

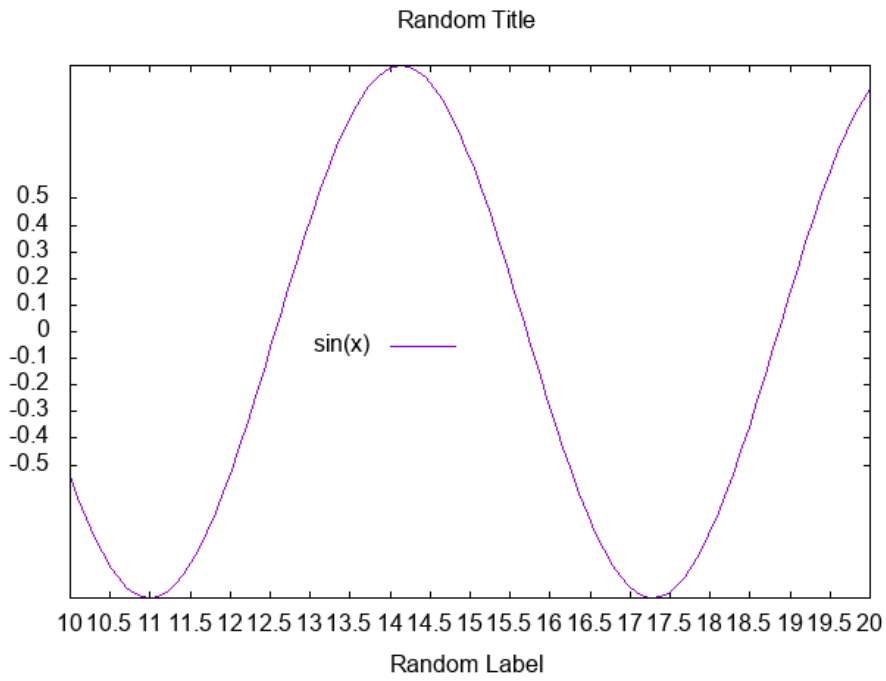
```
unset xtics
```

- **RESET**

```
rest
```

- **LOAD**

```
load "setup.gp"
```



- **REPLOT**

```
p sin(x), cos(x)
rep cos(2*x) w p
rep sin(2*x) w l lw 3 lt 0
    lc "magenta"
```

- **UNSET**

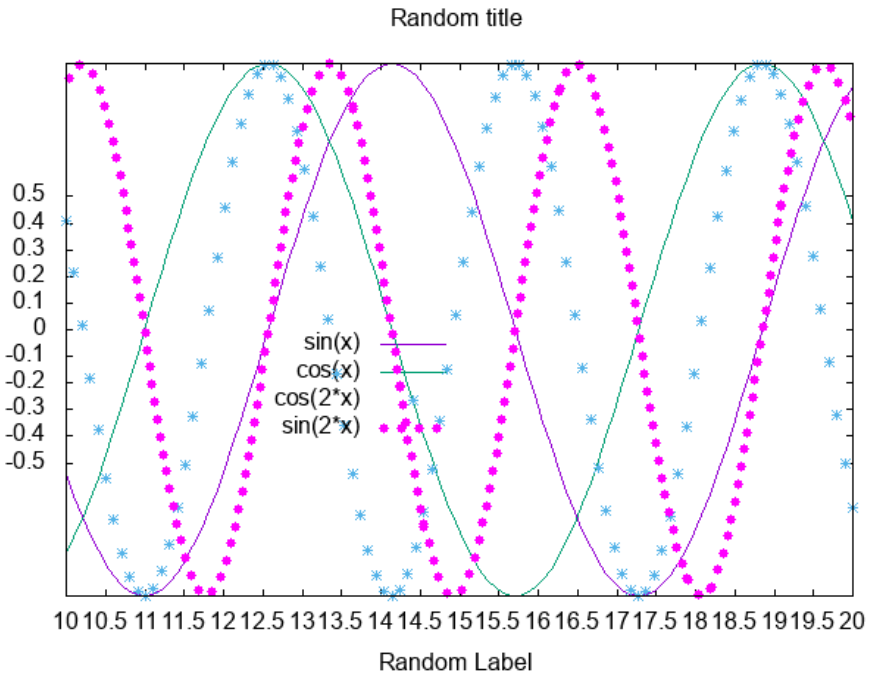
```
unset xtics
```

- **RESET**

```
rest
```

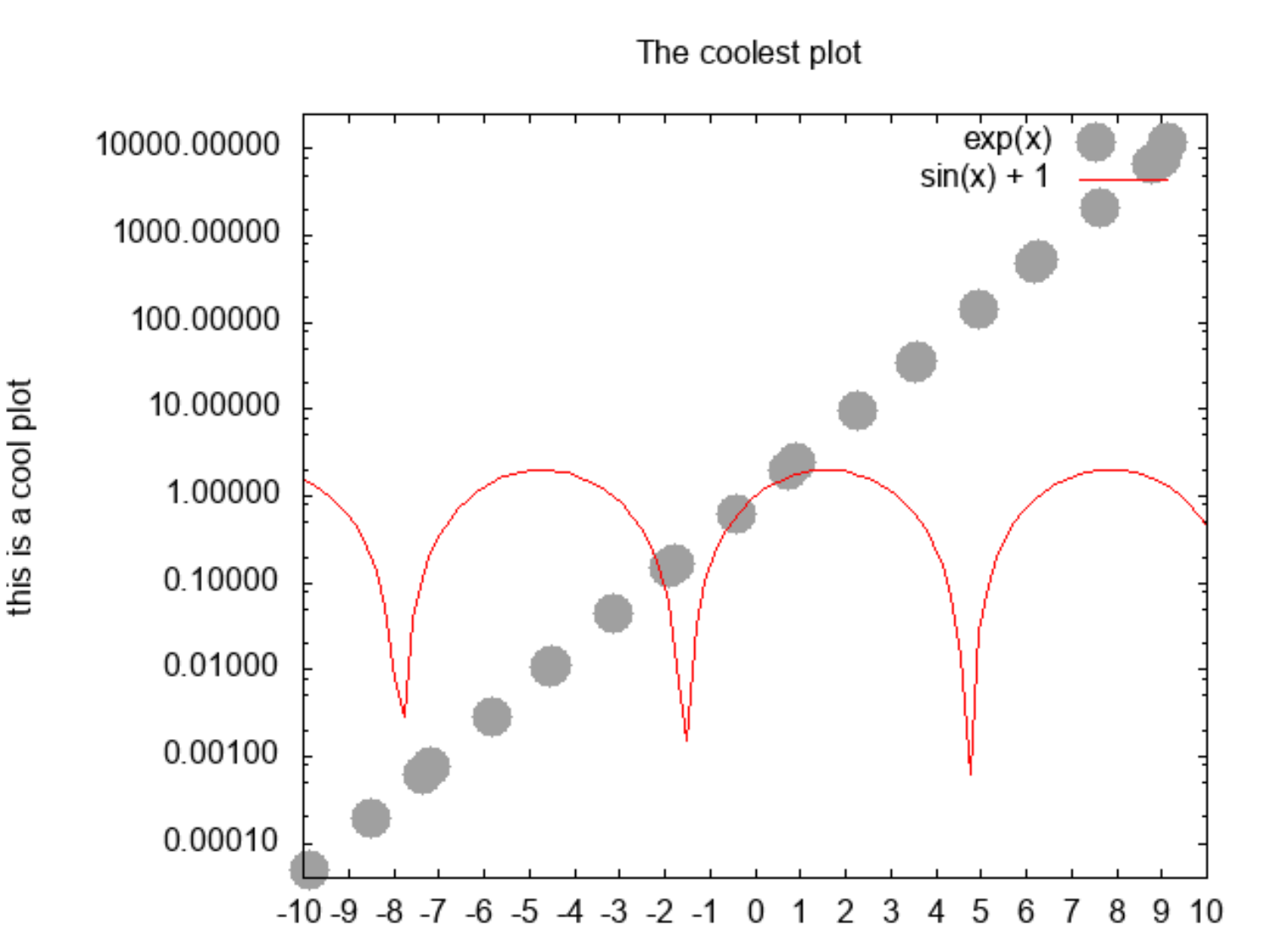
- **LOAD**

```
load "setup.gp"
```



SKILLPILLS

MATCH THIS PLOT



Outputting images (and other things)

- Gnuplot has many different output environments known as "terminals"
- Common terminals: epslatex, png, x11
- To output to file, simply type

```
set terminal png
set output "myfavoritepngfile.png"
rep
```

- The easiest way to dynamically create plots is to work in the x11 terminal and switch to png when needed

EXERCISE

Output that last plot to png

How to data

Here's where we are going to spend a majority of our time... Gnuplot has a few different formats to read in

- For this file format, we treat the data as a matrix

```
splot "out.dat" matrix w l
```

```
a11     a12     a13     a14
a21     a22     a23     a24
a31     a32     a33     a34
a41     a42     a43     a44
```

How to data

Here's where we are going to spend a majority of our time... Gnuplot has a few different formats to read in

- For this file format, we output data into columns, and we can plot these columns individually

```
p "out.dat" u 2:3
splot "out.dat" u 1:2:3
```

- We leave 2 lines between every index

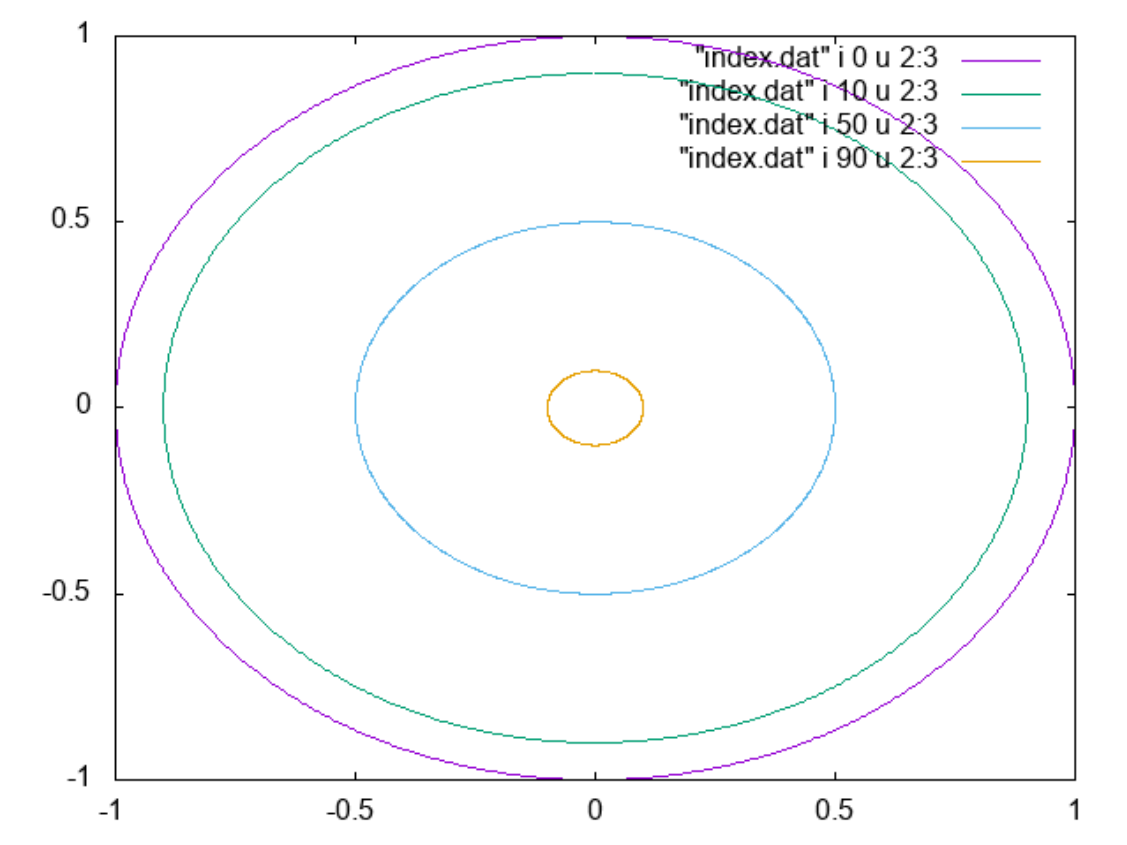
```
p "out.dat" i 1 u 2:3
```

```
X       Y       Z       W
1.0     2.0     3.0     4.0
1.1     2.1     3.1     4.1
1.3     2.3     3.3     4.1


1.4     2.4     3.4     4.4
1.5     2.5     3.5     4.5
1.5     2.5     3.5     4.5
```

EXERCISE

- Plot index_0.dat using 1:2, 2:3, and 3:4
- Plot index.dat using multiple indices and recreate this plot:



Using every

The crazier the data set, the more we can do with it!

- **EVERY** allows you to sift through your data.

```
# Plots every 10th line
p "out.dat" u 2:3 every 10

# plots every 10th line of
    every 10th block, starting
    at line 10 or the 1th block
    and going to line 50 of the
    5oth block
p "out.dat" u 2:3 every
    10:10:10:10:50:50
```

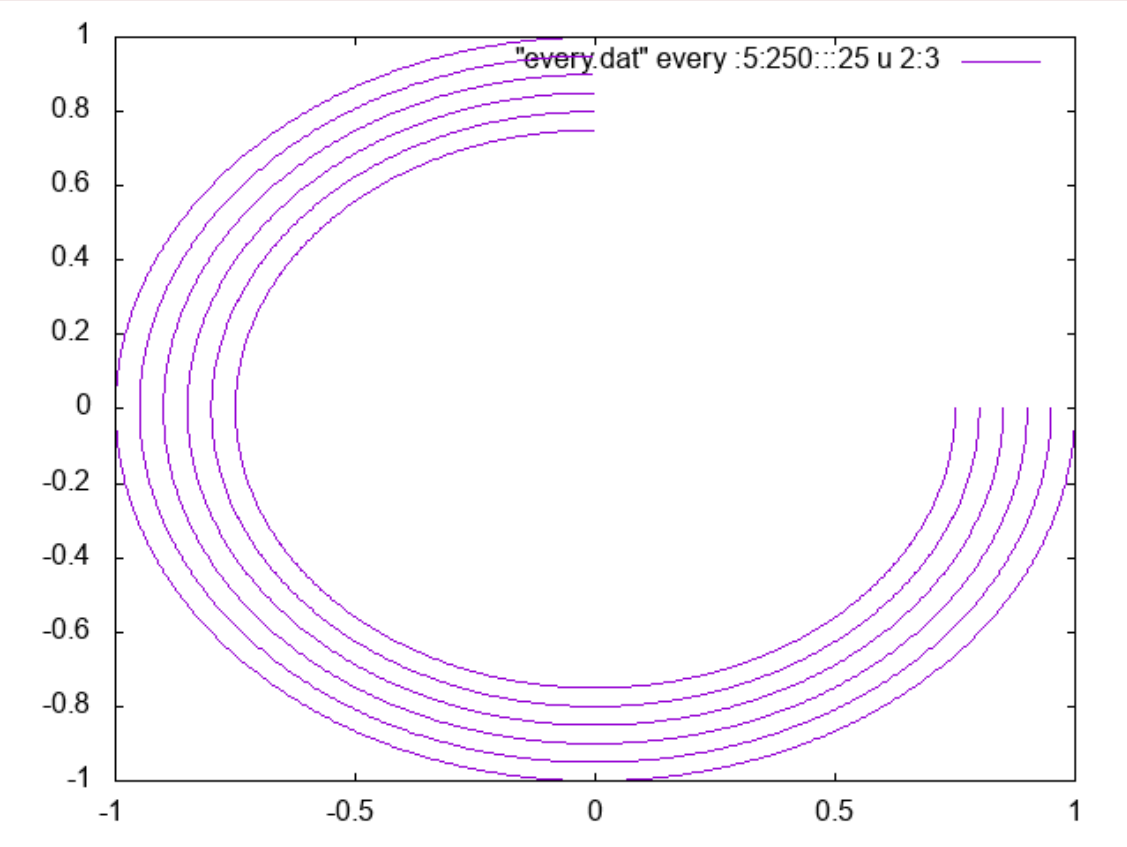
```
plot "my.dat" every A:B:C:D:E:F

A: line increment
B: data block increment
C: The first line
D: The first data block
E: The last line
F: The last data block
```

NOTE: for this to work, we need one line between every index (or "block") instead of 2!

EXERCISE

- Using every, recreate this plot:



Control structures

- We can use loops

```
do for [ii=0:100:1] {plot "out.dat" i ii; pause 0.01}
plot for [ii=0:100:1] "out.dat" i ii u 2:3
x = 0
while (x < 100){x = x + 1; print(x)}
```

- We can use conditions

```
x = 5
if (x==5){plot "out.dat" u 2:3 w l}
```

EXERCISE

- Make an animation using your favorite columns from your favorite dataset using your favorite control structure!
- To clarify: your favorite control structure is a for loop.

Conclusions

- gnuplot is a powerful plotting tool that can be complicated but is worth using in many situations.
- Next week we will learn how to do some crazy plots, so if you have any plot you want to see done in particular, let me or Albert know!