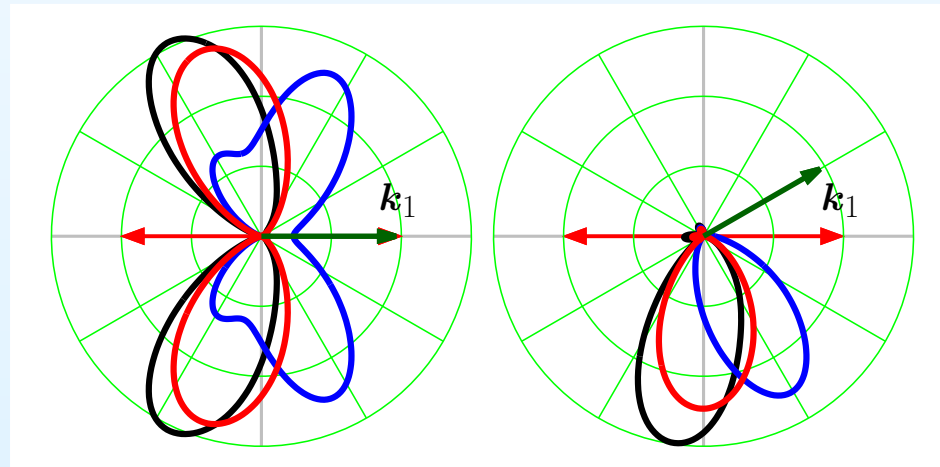


Plotting with gnuplot

Xiaoxu Guan

High Performance Computing, LSU

September 28, 2016



- What is **gnuplot** and what can **gnuplot** do for us?
- Fundamental concepts in gnuplot
- Plot functions, curves, points, . . .
- Built-in functions and functions user defined
- Customize axes, labels, and legends
- Other types of 2D plots
- Gnuplot with data files
- Borders and multiple plots
- Customize postscript plots
- Three-dimensional and contour plots
- Further reading

What is gnuplot?

- Gnuplot is one of the open-source cross-platform plotting systems;
- Note **gnuplot** is **NOT** a **GNU** project;
- It was written in C/C++, and other languages;
- Gnuplot is mostly a **command-line** driven plotting program;
- It supports many output formats of figure: (E)PS, PDF, SVG, L^AT_EX, PNG, ...
- Gnuplot is an implementation and a **scripting** system as well;
- It supports both batch and interactive modes;
- Gnuplot is a **backbone** of many other GUIs tools: Maxima, GNU Octave, Kayali, ...;
- It can be called by Perl, Python, Ruby, Fortran 95, ...;
- It has widely been used to create publication-quality figures;

Fundamental concepts in gnuplot

- Remember that gnuplot is a **command-line** driven program. It can also be thought as a scripting system;
- All the commands are case sensitive. They may be abbreviated as **short** as they are not ambiguous;
- Like shell scripting, gnuplot uses **#** as comments;
- Gnuplot reserves the **backslash** (\) as the character of line continuation;
- Plot on a **canvas**:

```
set terminal { <type of terminal> <options> }
```

```
set size x,y
```

```
set output "file.name"
```

- Type of **terminals**: png, jpeg, postscript, pdf, latex, pslatex, epslatex, svg, x11, and many others;

Fundamental concepts in gnuplot

- Different terminals may have different settings;

```
set terminal postscript portrait size 6, 4 \  
    enhanced monochrome "Helvetica" 16  
set output "figure1.ps"           # size in inch
```

```
set term png size 600, 400      # absolute size  
set size 0.5, 0.4              # scale factors  
set output "figure2.png"       # size in pixel
```

```
set term pslatex color dashed dl 4 15  
set size 1.0, 2.0  
set output "figure3.tex"       # size in inch
```

```
set term pdf color rounded linewidth 4  
set output "figure4.pdf"  
# size in 5 inches by 3 inches
```

- Use **help** command within gnuplot;

Plot functions, curves, points, ...

- **plot** is the primary command for 2D figures;

```
plot <ranges> <iteration> \
    <function> | "<datafile>" datafile-modifiers \
    axes <axes> <title-spec> with <style> ...
```

```
set xrange [a:b]
set yrange [c:d]
```

- Note that functions can be built-in or user defined
(`lt=linetype` and `lw=linewidth`);

```
a = 4.0 ; b = 5.2
my_fun(x) = exp(-a*x) * sin(b*x)
plot [-2:2] [-5:5] my_fun title "This my plot" \
    lt 1 lw 4
```

- Define two parameters `a`, `b`, and the function `my_fun(x)`;

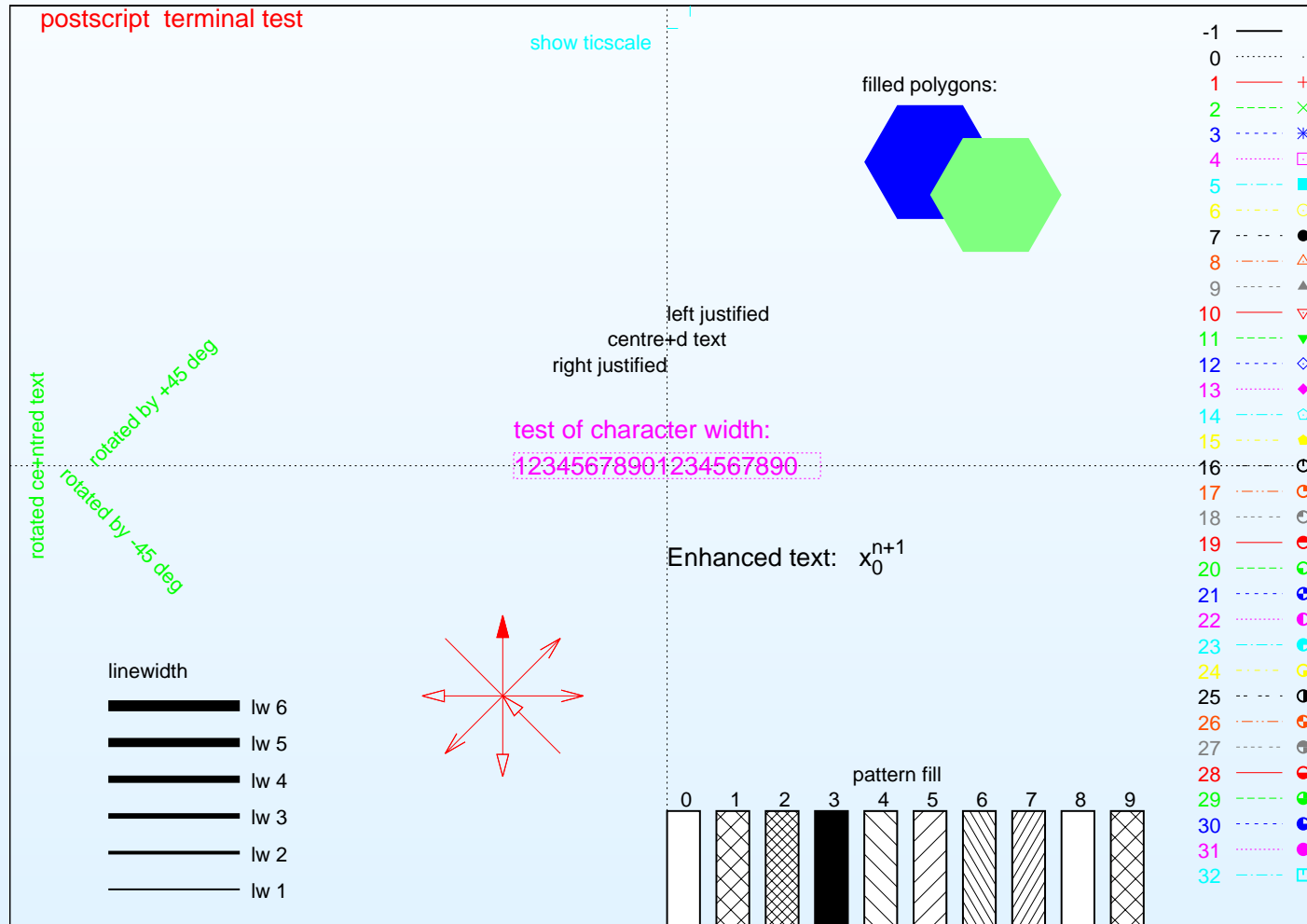
Built-in functions

- Gnuplot supports many built-in functions:

Function	Description
$\log(x)$	log function of x in natural base (e)
$\log_{10}(x)$	like the above, but in base 10
$\text{rand}(0)$	generator of pseudo random numbers (real)
$\text{sqrt}(x)$	\sqrt{x}
$a^{**}b$	a^b
$\text{exp}(x)$	e^x
$\text{sin}(x)$	$\text{sin}(x)$
$\text{tan}(x)$	$\text{tan}(x)$
$\text{abs}(x)$	$\text{abs}(x)$
$\text{sgn}(x)$	+1 if $x > 0$, or -1 if $x < 0$
$\text{gamma}(x)$	$\text{gamma}(x)$

Plot functions, curves, points, ...

- How to control **linetype**, **linewidth**, or **linecolor**, ...:
- Again, this is terminal dependent: run command **test**;

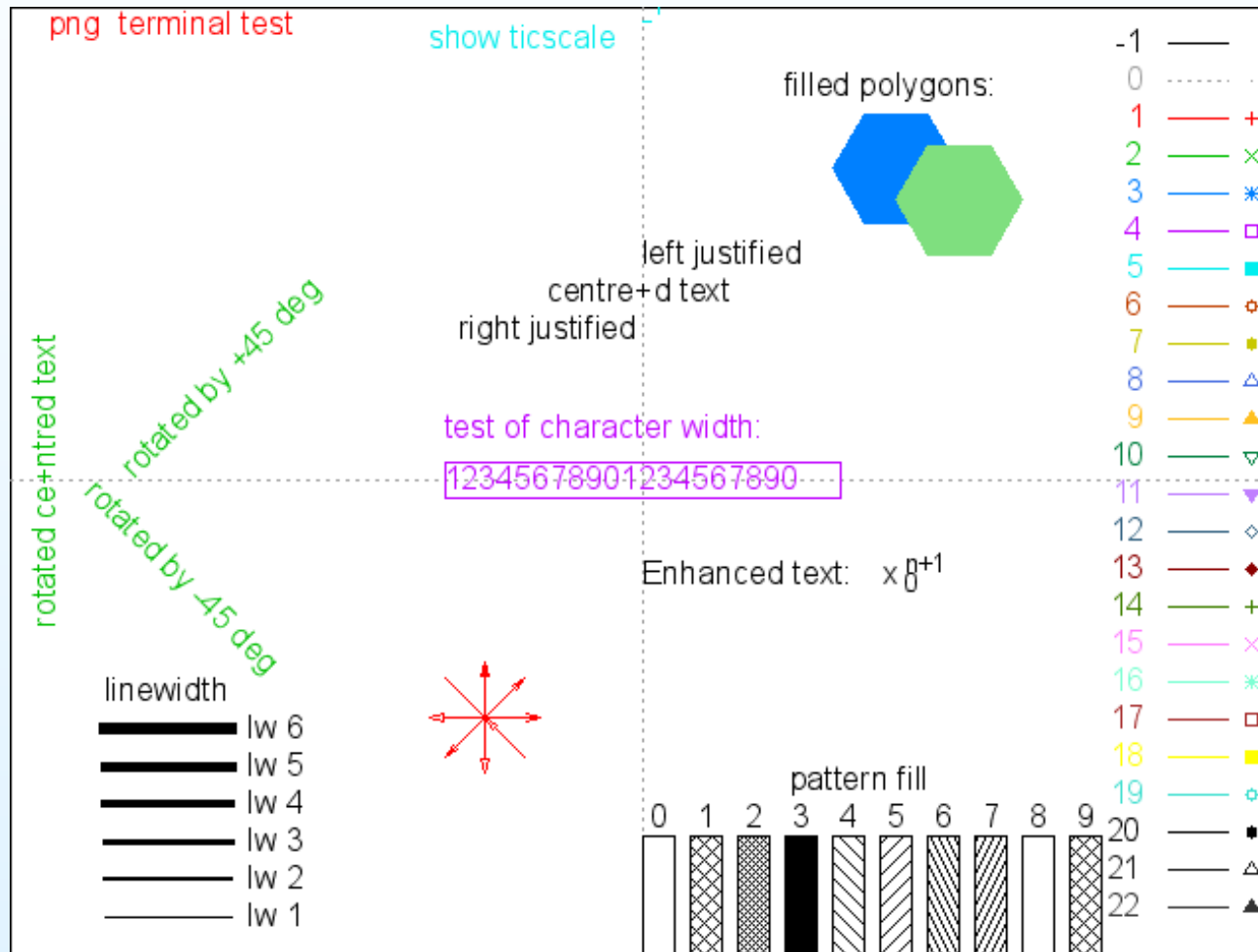


postscript
terminal

It ranges
from -1 to 32

Plot functions, curves, points, ...

- How to control **linetype**, **linewidth**, or **linecolor**, ...:
- Again, this is terminal dependent: run command **test**;



png
terminal

It ranges
from -1 to 22

Plot functions, curves, points, ...

- How to control **linetype**, **linewidth**, or **linecolor**, ...:
- Line colors can be represented either in name or in hex;

```
plot sin(x) lc rgb "blue" lt 1 lw 4, \  
      tan(x) lc rgb "#800000" lt 2 lw 2 # dark brown
```

- Gnuplot allows us to define user-specified line style (**ls**): it can be reused once defined;

```
set style line 1 lt -1 lc rgb "orange" lw 2  
set style line 2 lt 0 lc rgb "red" lw 3  
set style line 3 lt 1 lc rgb "cyan" lw 4  
plot sin(x) ls 1 title "orange", \  
      tan(x) ls 2 title "red",\  
      tan(x)*sin(x) ls 3 title "cyan" \  
      (x*x-x+2)*sin(x) ls 3 title "cyan"
```

- Note line color can also be a variable;

Plot functions, curves, points, ...

- What about **point type** or **size**?
- Again, it depends on terminals: run **test**;

```
plot sin(x) with points pt 2 ps 3  
plot sin(x) with p pt 4 ps 3
```

- How to add **points** on **lines**?

```
plot cos(x) with linespoints lt 1 lw 2 \  
  lc rgb "blue" pt 2 ps 3  
plot cos(x) with lp lt 1 lw 2 \  
  lc rgb "green" pt 4 ps 2
```

- Generalize the definition of **line style**:

```
set style line 1 lt -1 lc rgb "orange" lw 2 \  
  pt 10 ps 1 # define ls 1
```

Customize axes, labels, and legends

- By default, gnuplot automatically draws axes, tics, labels, **legends** (also known as **keys**), etc;
- Gnuplot also allows us to customize them;

```
set xrange [1:200] ; set yrange [-1:1]
```

- Control the axis tics:

```
set xtics 50 ; set ytics 0.2      # main tics
set mxtics 5 ; set mytics 2      # minor tics
set xtics add (1)
# manually add label 1 on the x axis
```

- Control the length of tics:

```
set tics scale 2.5      # main x and y tics scales
set xtics scale 2.5     # main xtics scale
set ytics scale 2.5     # main ytics scale
```

Customize axes, labels, and legends

- Sometimes we might want to draw tic marks outwards;

```
set xtics scale 1.5 out      # default is in
```

- Control the axis labels (note **offset**):

```
set xtics font "Times New Roman,12"  # font and size
set xlabel "Number of cores" offset 0.0, 0.3
# specify the x-axis label
set ylabel "Speedup"                # specify the y-axis label
```

- Control the formats of labels:

```
set format y "%3.1f"          # format of y labels
set log y                     # y tics in  $\log_{10}$ 
set format y "$10^{\%01T}$"   # y in  $10^k$ 
```

- Cancel it? (using **unset**)

```
unset log y                   # cancel the log setting of the y axis
```

Customize axes, labels, and legends

- How to control **keys (legends)**:

```
unset key          # don't draw the keys
set key at -5, 0.8 # draw the keys at the coordinates
set key at 0, 1.5 spacing 0.9 samplen 2
# draw the keys with specified line properties
```

- Add **labels**:

```
set label 1 at -1,0.2 "This is the first lobe"
set label 2 at 2,0.8 "T=0.4 a.u."
# define labels 1 and 2
```

- Draw **arrows**:

```
set arrow 1 from 1,0.4 to 3.5, 2.9 \
nohead filled lt 2 lw 1 lc rgb "blue" back
set arrow 2 from -1,0.0.2 to 0.5, 4.0 \
head filled lt 2 lw 1 lc rgb "dark-green" back
```

Other types of 2D plots

- Gnuplot supports many other types of 2D plots;
- Let's consider the **polar** plot:

```
set dummy t
set trange [0:pi*0.5]
set polar
plot 2-sin(t) lw 4
# define a dummy variable  $t$  and plot a function  $y(t)$ 
```

- Gnuplot also supports bar plots, pie plots, error bars, loading binary figures, multiple axes, filling space between lines and other features;

```
plot "lsutiger.png" binary array=(128,128) ...
plot "myfigure" binary ...
```

- Let's see how gnuplot can handle data files;

Gnuplot with data files

- For 2D plots, the **textual** data file can be in the format of **multiple** columns. Both floating numbers and scientific notation (1.4E-2 or 2.6e+10) are recognizable;
- Add **#** in your data files whenever necessary for comments;

```
# Intel MPI lib on SuperMIC.  
# 16 MPI tasks on 16 nodes. Each node has one MPI  
# task either in offload or no-offload mode.  
# offload mode      NON-offload (only on host)  
# column_1      column_2      column_3      column_4  
# 1              1710.53        1              133.06  
# 5              358.89         2              69.99  
# 10             192.72         3              48.79  
# ...  
# 240            22.17
```


Gnuplot with data files

- What we want is to plot the speedup ($S_n = T_1/T_n$) against the number of threads on **MIC** and **CPU**;
- Column index starts from 1;

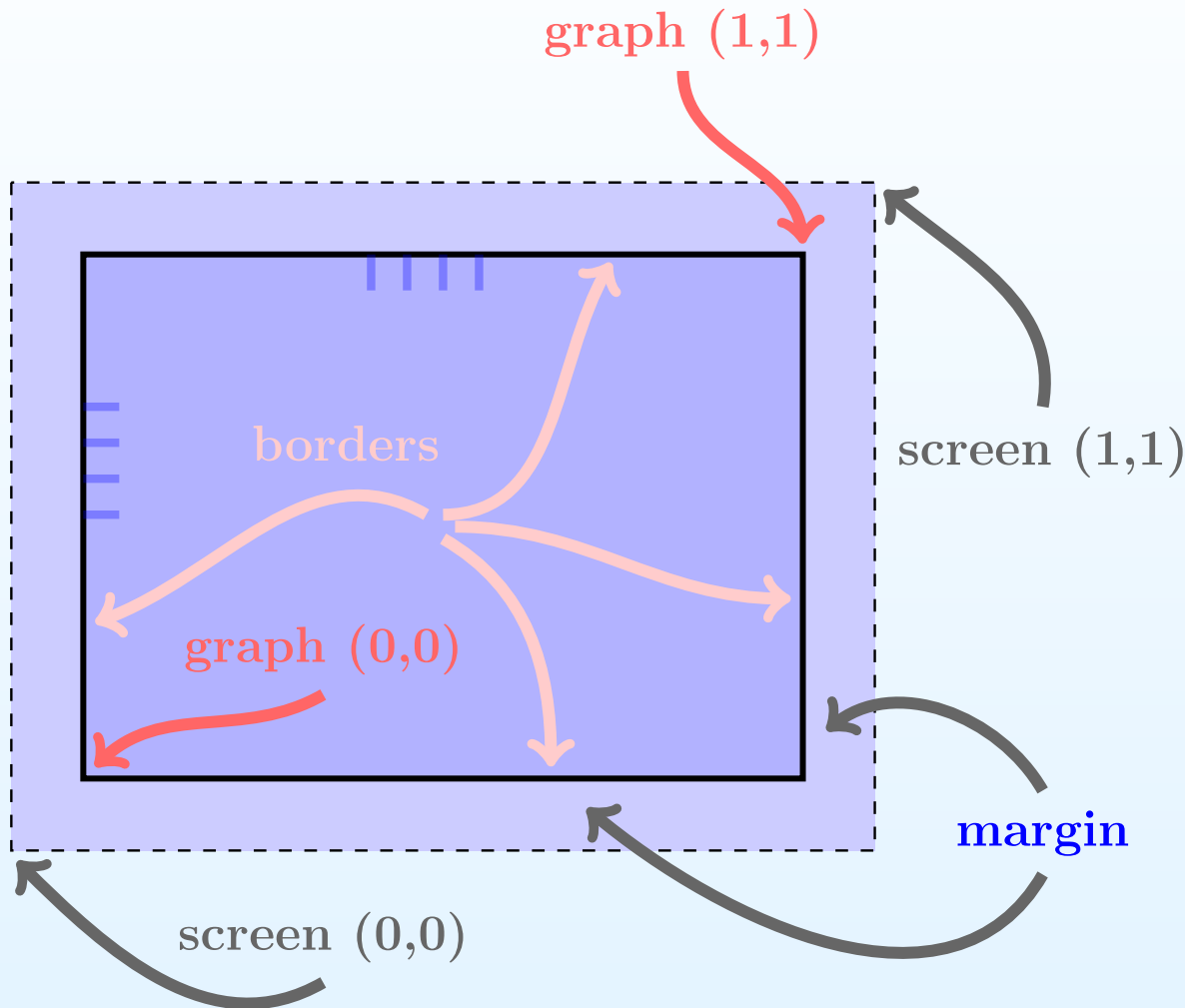
```
tmic = 1710.53 ; tcpu = 133.06  
plot "timing.dat" using 1:2 w l lt 2 lw 3  
plot "timing.dat" u 1:(tmic/$2) w l lt 2 lw 3
```

- **Multiple** lines on the same plot. If necessary, we can use the same or other data files;
- Gnuplot supports the **path** in filename as well;

```
tmic = 1710.53 ; tcpu = 133.06  
plot "timing.dat" u 1:(tmic/$2) w l lt 2 lw 3, \  
"" u 3:(tcpu/$4) w l lt 3 lw 2 \  
pt 2 ps 3 # add points
```

Borders and multiple plots

- More terminologies: **screen**, **graph**, **origin**, **margin**, etc;

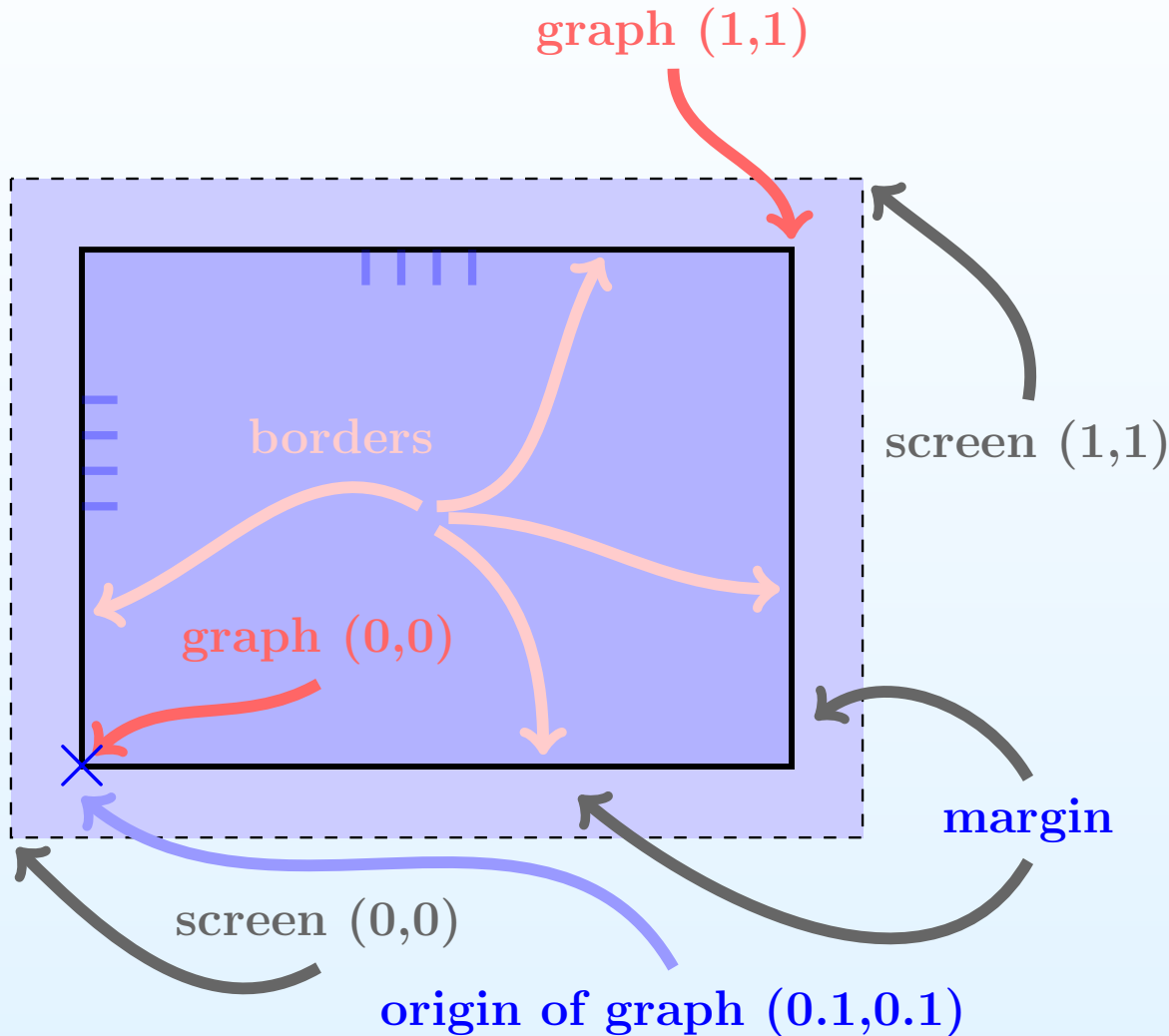


- **Screen** (0,0), (1,1) is the entire area (**canvas**).
- Borders mean four axes with tics.
- Borders define the **graph** area: (0,0),(1,1).
- **Margin** area is for axis labels and tics.

- What about the origin?

Borders and multiple plots

- More terminologies: **screen**, **graph**, **origin**, **margin**, etc;



- **set origin (0.1,0.1)**
- In the **screen** coordinates, set the **origin of graph**.
- Origin is useful to **multiple** plots.
- Each panel can have its own **origin**.

Borders and multiple plots

- **Borders** in gnuplot mean four **axes**;
- Gnuplot allows us to control four borders using **mask** values;

```
set border <options>          # define borders
```

- **Mask** values: bottom is 1, left 2, top 4, and right 8.

```
unset border    # important!  
set border 3    # draw bottom and left axes (3=1+2)  
set border 7    # draw bottom, left, and top axes (7=1+2+4)  
set border 15   # same as the default  
show border
```

- Note the above setting has nothing to do with tic marks;

```
set xtics nomirror    # turn off x tics mirror  
set ytics nomirror    # turn off y tics mirror
```

- Borders are drawn on the **top** of all plot elements (default);

Borders and multiple plots

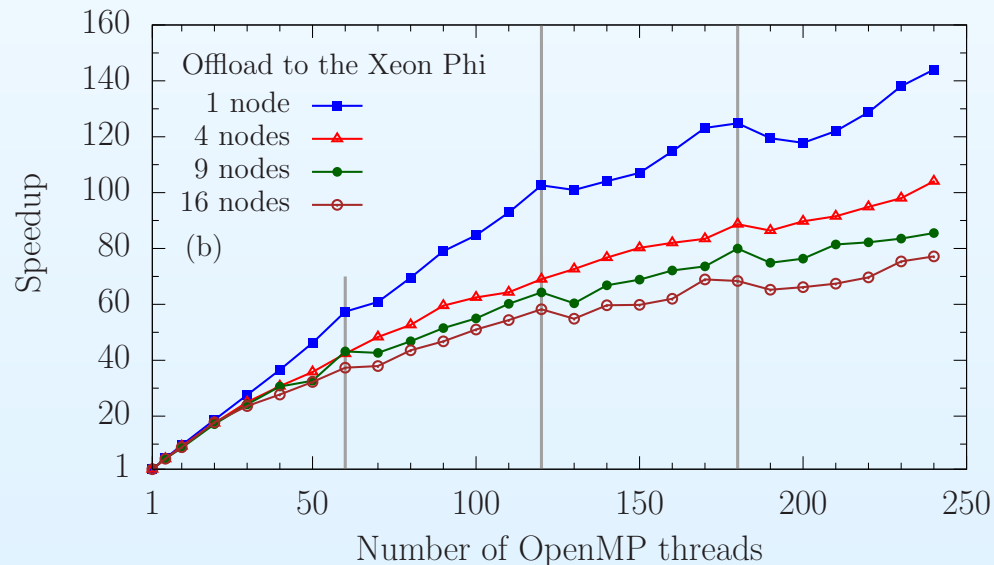
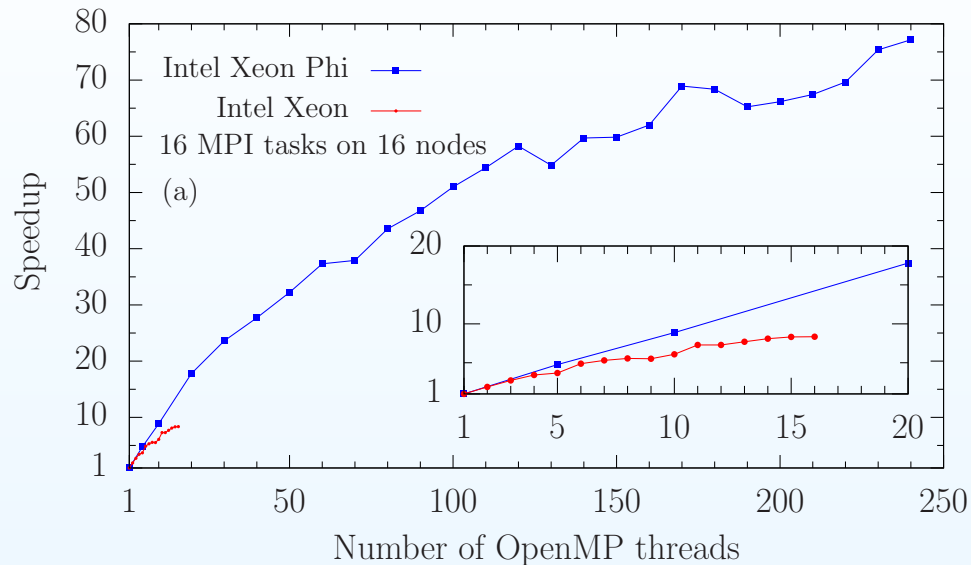
- On the same **canvas**, draw **multiple** plots on several panels;
- We may organize plots in terms of $m \times n$ (# rows \times # cols) panels. Each panel may be customized separately;

```
set multiplot          # only once
set origin 0,0         # for the canvas or a panel
# the first panel
set origin 0.5/5.0, -1.5/3.5
set lmargin 0          # left margin
set rmargin 0          # right margin
set tmargin 0          # top margin
set bmargin 0          # bottom margin

# the second panel
set origin 0.5/5.0, 0.0/3.5
...
```

Borders and multiple plots

- The output of the script **multiplot.gpl**:



- The terminal is **pslatex**.
- Define a **canvas** first on which all panels will be drawn.
- Generally, we can use **set multiplot** to organize multiple panels in any way we want.
- Each panel can have its **own origin, size, and margins**, etc.

3D surfaces and contours

- The data file needs to be **blocked** and each block separates by a blank line:

```
x1 y1 z11
x1 y2 z12
x1 y3 z13
...
x1 ymax z1max
[blank line]
x2 y1 z21
x2 y2 z22
x2 y3 z23
...
x2 ymax z2max
[blank line]
...
```

- A **blank line** between blocks.
- **Whitespace** between columns.
- The sequence of columns is **irrelevant**.
- It is better to use the terminal of **postscript** (or other vector terms).
- In most cases, **contour** plots are more useful.
- The gnuplot command is **splot** <options>

3D surfaces and contours

- Gnuplot supports “**heat map**” plot;
- Color mapping is used to represent one of the columns;

```
set term postscript enhanced color rounded 21
set pm3d map
set size square
```

- The **splot** syntax:

```
splot "myfile_1.dat" u 1:2:3
splot "myfile_2.dat" u 2:1:3
splot "myfile_3.dat" u 2:1:$3/1.e-4
splot "myfile_4.dat" every 2 u 2:1:($3)/1.e-4
```

- Control **colorbox**:

```
set colorbox horizontal user \
origin 0.2,0.11 size 0.62,0.015 # also vertical
```


3D surfaces and contours

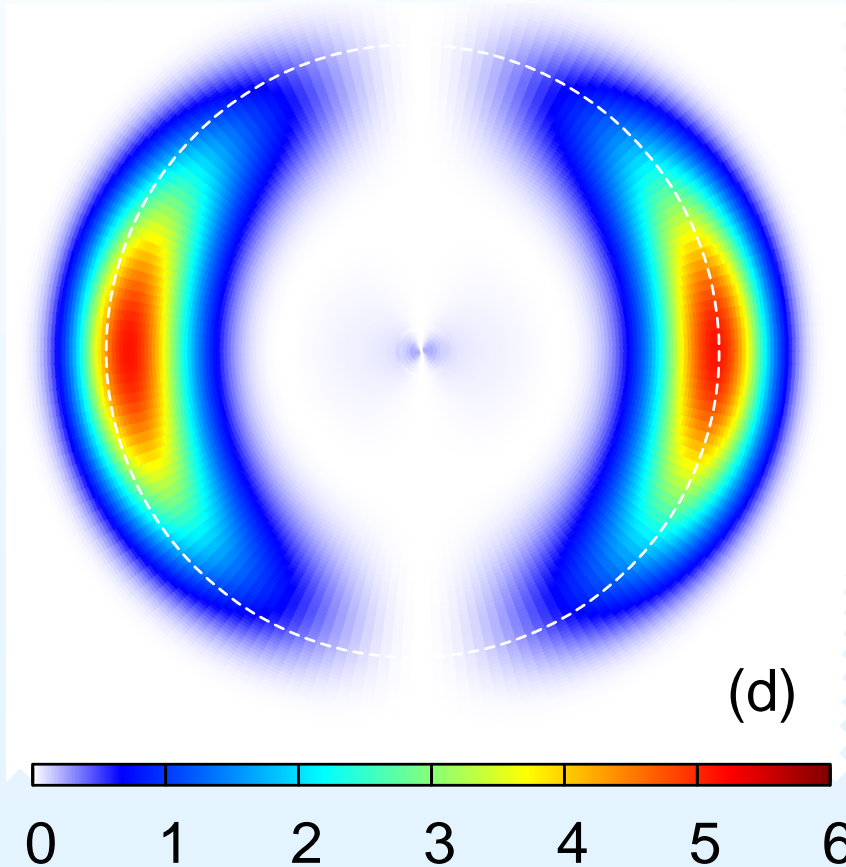
- We now know how to control the position and orientation of **colorbox**, etc;
- Let's consider the color scheme: **palette**;
- Some useful examples of palette:

```
set palette rgb 21,22,23 # hot (black-red-yellow-white)
set palette negative rgb 21,22,23
set palette rgb 33,13,10
# rainbow (blue-green-yellow-red)
set palette rgb 34,35,36
# AFM hot (black-red-yellow-white)
```

```
set palette defined ( 0 "#000090", 1 "#000fff", \  
2 "#0090ff", 3 "#0fffee", \  
4 "#90ff70", 5 "#ffee00", \  
6 "#ff7000", 7 "#ee0000", 8 "#7f0000" ) # default matlab
```

3D surfaces and contours

- The output of **map3d.gpl**:



- set multiplot
- set size square
- set pm3d map
- unset border
- set colorbox ...
- set palette defined ...
- splot 'map3d.dat'
every 2 u
(\$1)*sin(\$2) :
(\$1)*cos(\$2) :
(\$3)/(\$1)/(\$1)/1.e-4
- ...

- **Gnuplot homepage:** <http://www.gnuplot.info>
- **Not So FAQs:** <http://lowrank.net/gnuplot/index-e.html>
- **Also a good one:** <http://www.gnuplotting.org>

Questions?
`sys-help@loni.org`