Scientific visualizations using Mathematica

Yang Li

Thursday, June 9, 2022

Graphics and Visualization

- Data visualization is an essential way to present your results
- Data visualization is also an important way to debug and to understand your data

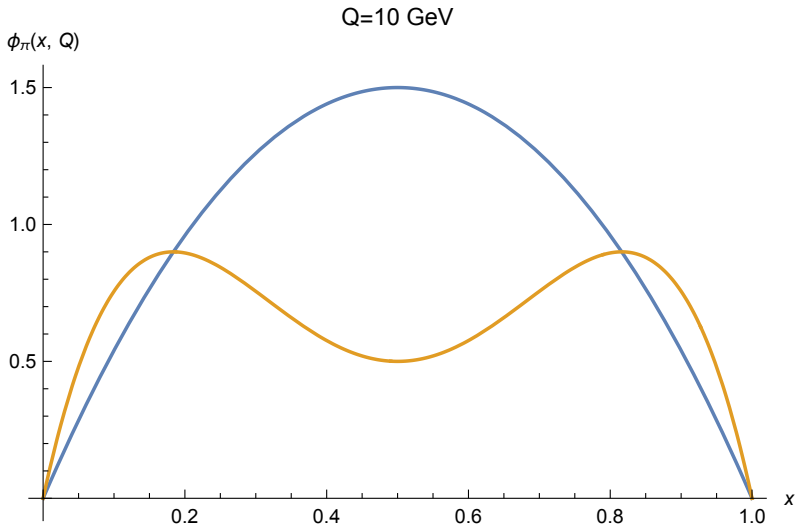
- Functionalities provided by *Mathematica*:
  - single-variable function visualization
    - Plot, ParametricPlot, DiscretePlot, PolarPlot, RegionPlot
    - LogPlot, LogLogPlot, LogLinearPlot, …
  - data visualization: ListPlot, ListLogPlot, ListPlot3D, …
  - double-variable function visualization
    - surface plot: Plot3D, ParametricPlot3D, ListParametricPlot3D
    - countour plot: ContourPlot, DensityPlot
    - vector field: VectorPlot, VectorDensityPlot, StreamPlot, StreamDensityPlot
    - three-variable: DensityPlot3D, ContourPlot3D, SliceDensityPlot3D, SliceContourPlot3D
  - miscellaneous:
    - Histograph, BarChart, MatrixPlot, …
  - Graphics, Graphics3D

Help yourself

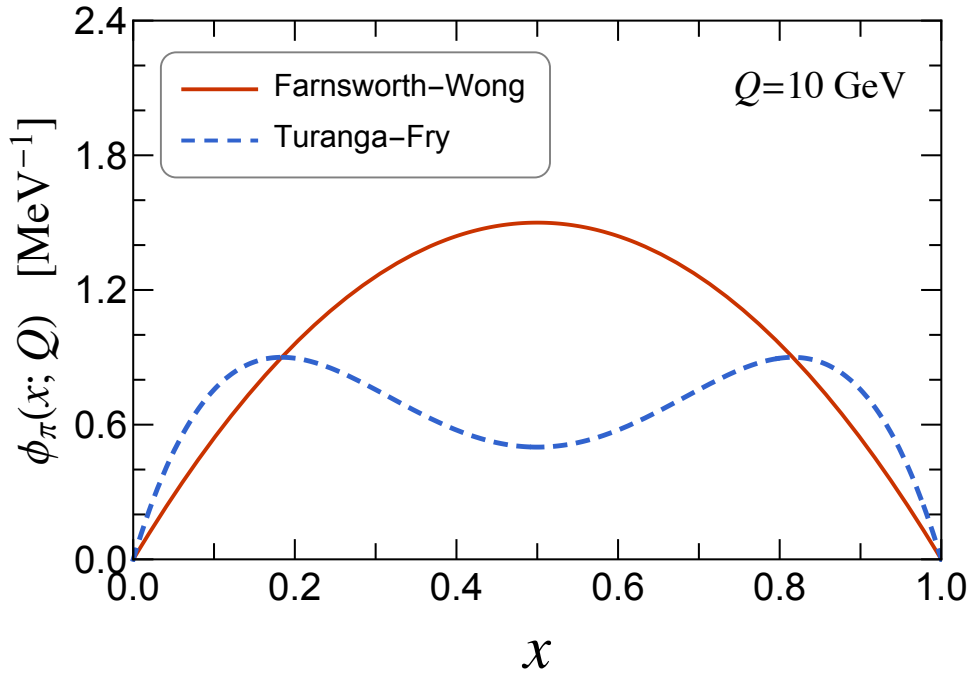
- *Mathematica* documentation center
  - Press F1 to inquires documentation of selected content
  - `?FunctionName` or `??FunctionName`
  - Examples are often very revealing
  - Online documentation centers
  - http://www.wolfram.com/language/fast-introduction-for-programmers/en/
- StackExchange
- Online *Mathematica* user groups

https://reference.wolfram.com/language/

Plot and ListPlot



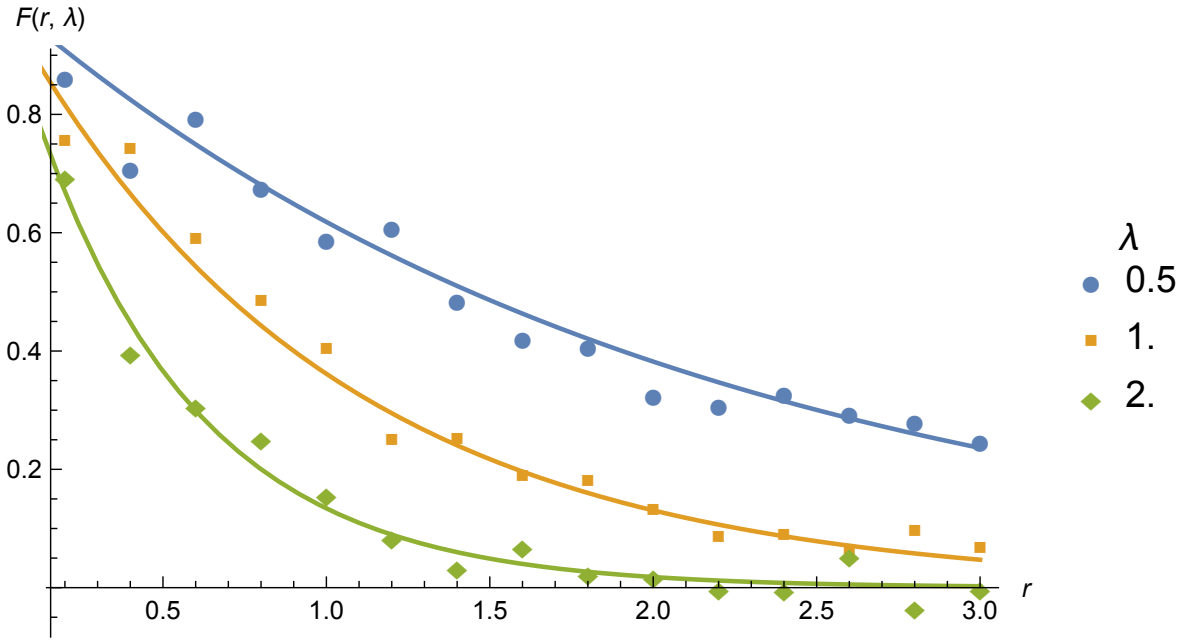

```
Plot[
{6x(1-x),4 x (3-13 x+20 x^2-10 x^3) }, {x, 0, 1},
AxesLabel→{x,ϕ_π[x,Q]},
PlotLegends→{"Farnsworth-Wong","Turanga-Fry"},
PlotLabel→"Q=10 GeV"
]
```



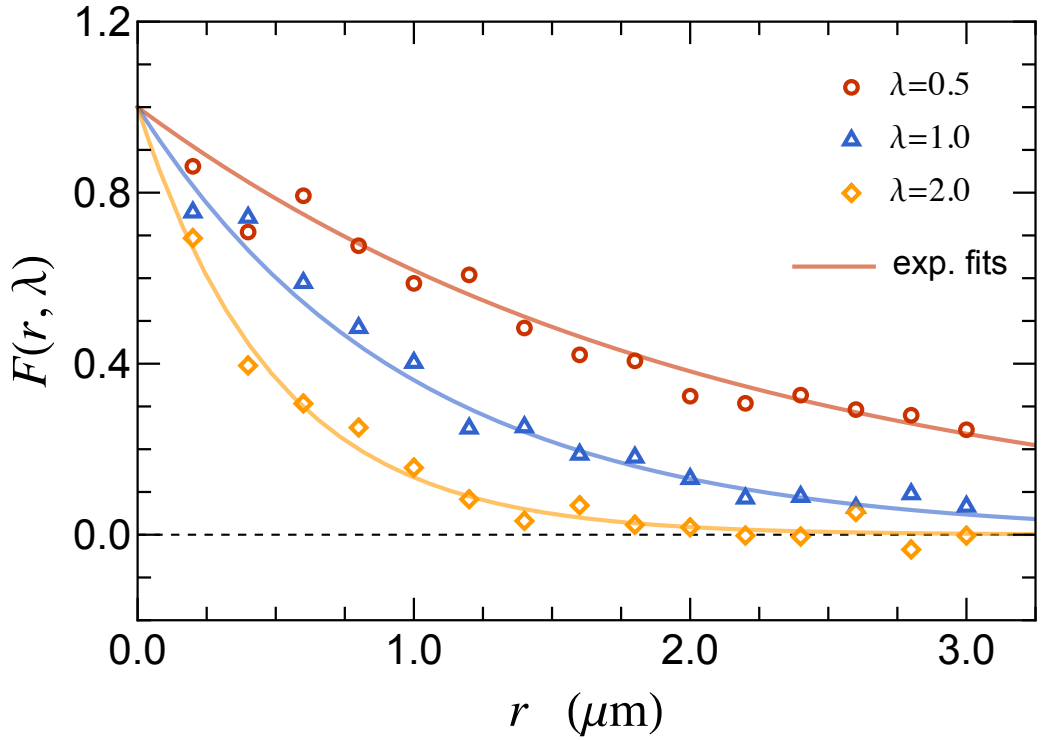

```
Show[
Plot[
{6x(1-x),4 x (3-13 x+20 x^2-10 x^3) }, {x, -0, 1},
(*size and range*)
PlotRange→{{0,1},{0,2.4}},
ImageSize→500,
AspectRatio→2/3,
(*line style*)
PlotStyle→{Directive[ColorData[112,1],Thick,Dashing[{}]],Directive[ColorData[112,2],Thickness[0.006],Dashing[Medium]]},
(*frame styles*)
Frame→True,Axes→False,FrameStyle→Directive[Black,Thickness[0.0025],FontSize→20],
FrameLabel→{Style[x,FontSize→32,FontFamily→"Times"],Style["ϕ_π(x; Q)  [MeV^-1]",FontSize→24,FontFamily→"Times"]},
FrameTicks→{{(*left*)Table[If[Mod[i,3]==0,{i/5,NumberForm[0.2*i,{2,1}],{0,-0.025}},{i/5,,{0,-0.015}}],{i,-20,20}],
(*right*)Table[If[Mod[i,3]==0,{i/5,,{0,-0.025}},{i/5,,{0,-0.015}}],{i,-20,20}]},
{(*bottom*)Table[If[Mod[i,2]==0,{i/10,NumberForm[0.1*i,{2,1}],{0,-0.025}},{i/10,,{0,-0.02}}],{i,-20,20}],
(*top*)Table[If[Mod[i,2]==0,{i/10,,{0,-0.025}},{i/10,,{0,-0.02}}],{i,-20,20}]}},
(*legend, additional texts*)
PlotLegends→Placed[LineLegend[{"Farnsworth-Wong","Turanga-Fry"},LabelStyle→Directive[FontSize→16],
LegendFunction→(Framed[#1, FrameMargins -> 5,RoundingRadius→8,FrameStyle→Gray] & )],Scaled[{.275,.825}]]

],
Graphics[{Text[Style["Q=10 GeV",FontSize→20,FontFamily→"Times"],Scaled[{0.85,0.875}]]}]
]
```

ListPlot with default style



ListPlot with publication style




```
Show[
ListPlot[{
data1[[All,{1,2}]],
data2[[All,{1,2}]],
data3[[All,{1,2}]]},PlotRange→All, PlotMarkers→Automatic,
AxesLabel→{ r,F[r, λ]},
PlotLegends→LineLegend[{0.5,1.0,2.0},LegendLabel→"λ"]],
Plot[{fit1[x],fit2[x],fit3[x]},{x,0,3}]
]
```

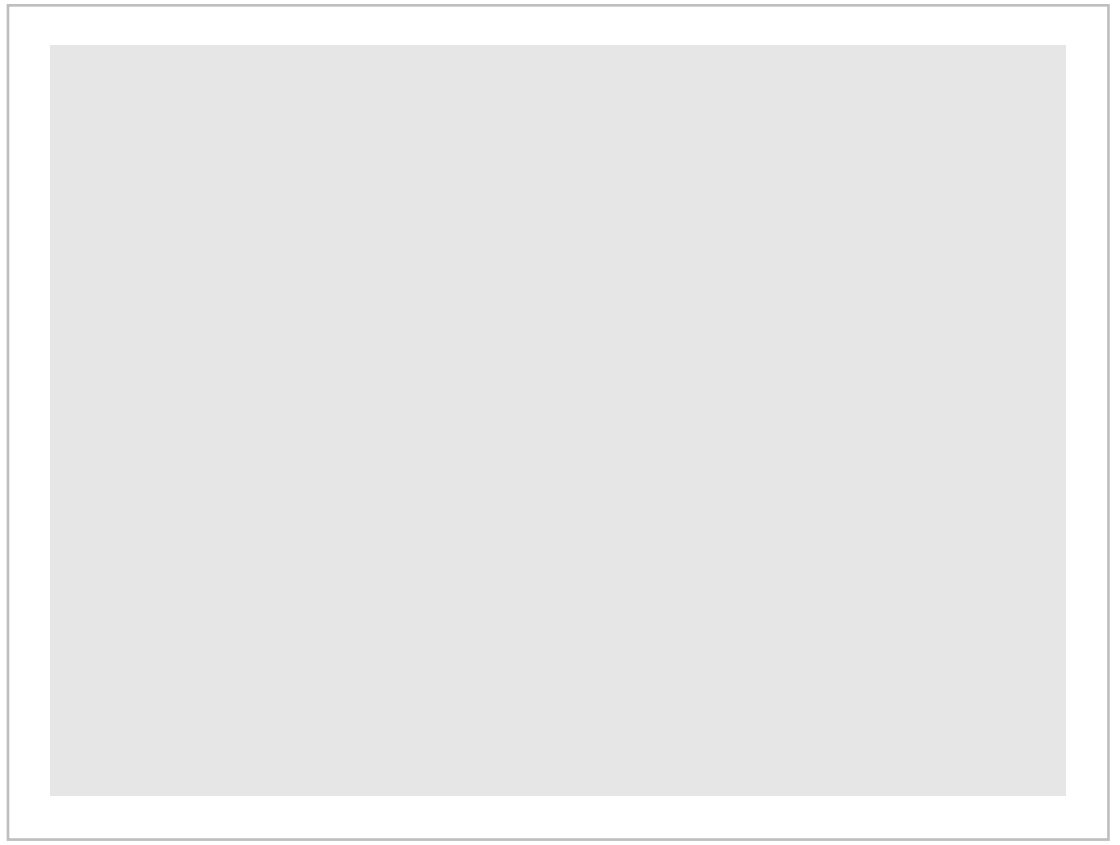
```
Show[
ListPlot[{
data1[[All,{1,2}]],
data2[[All,{1,2}]],
data3[[All,{1,2}]]},
(*size and range*) PlotRange→{{0,3.25},{-0.2,1.2}}, ImageSize→500, AspectRatio→2/3,
(*marker style*) PlotMarkers→(PlotMarkers/.Charting`ResolvePlotTheme["OpenMarkersThick",ListLinePlot]), PlotStyle→112,
(*frame styles*) Frame→True,Axes→False,FrameStyle→Directive[Black,Thickness[0.0025],FontSize→20],
FrameLabel→{Style["r    (μm)",FontSize→24,FontFamily→"Times"],Style["F(r, λ)",FontSize→24,FontFamily→"Times"]},
FrameTicks→{
{(*left*)Table[If[Mod[i,4]==0,{i/10,NumberForm[0.1*i,{2,1}],{0,-0.025}},{i/10,,{0,-0.015}}],{i,-20,20}],
(*right*)Table[If[Mod[i,4]==0,{i/10,,{0,-0.025}},{i/10,,{0,-0.015}}],{i,-20,20}]},
{(*bottom*)Table[If[Mod[i,4]==0,{i/4,NumberForm[0.25*i,{2,1}],{0,-0.025}},{i/4,,{0,-0.015}}],{i,-20,20}],
(*top*)Table[If[Mod[i,4]==0,{i/4,,{0,-0.025}},{i/4,,{0,-0.015}}],{i,-20,20}]}},
(*legend, additional texts*)
PlotLegends→Placed[LineLegend[{"λ=0.5","λ=1.0","λ=2.0"},LabelStyle→Directive[FontSize→16,FontFamily→"Times"]], Scaled[{.85,.8}]],
(*Grids*) GridLines→{None,{0}}, GridLinesStyle→Directive[Dashed, Black,Thickness[Medium]]
],
(*fits*)
Plot[{fit1[x],fit2[x],fit3[x]},{x,0,4},
PlotStyle→{Directive[ColorData[112,1],Thick,Opacity[0.6],Dashing[{}]],Directive[ColorData[112,2],Thick,
Opacity[0.6],Dashing[{}]],Directive[ColorData[112,3],Thick,Opacity[0.6],Dashing[{}]]},
PlotLegends→Placed[LineLegend[{"exp. fits"},LabelStyle→Directive[FontSize→16]],Scaled[{.85,.6}]]]
]
```

Redesigning your Plot

- The default style is not suitable for publication quality figures
- Adjust the plot options:
  - PlotRange, PlotPadding, Background, ImageSize etc (see Canvas)
  - thickness, dashing and other plot style (see PlotStyle, Colors)
  - labels, font and size (see Labels, Font and Size)
  - redesign ticks and markers (see Plot Markers)
  - customize PlotLegends (see Plot Legend)
- Combine plots: Show, Epilog, GraphicsRow, GraphicsGrid
  - `Show[g1, g2, g3, options]`
  - layout and layers: in Show, g_3 over g_2 over g_1
  - style control: global and local
  - multipanel figures: DIY, external package, e.g. *SciDraw* of Mark Caprio, …

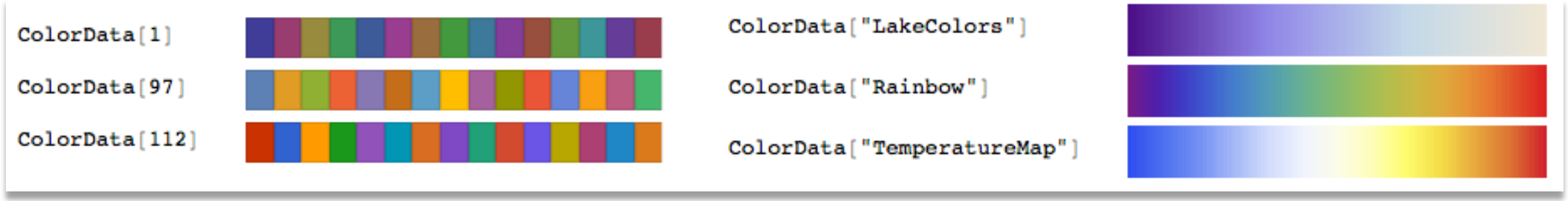
Canvas

- PlotRange
- Padding
- ImageSize
- AspectRatio
- Framed, Labeled, Background
- Outlined PDF



Colors

- *Mathematica* has some cool color collections
  - indexed colors: `ColorData[n]`
  - gradient colors: `ColorData["name"]`



- I personally prefer to more *restrained* color tones
  - example: Red, Blue, ~~Green~~ / Darker[Green], ~~Yellow~~ / Orange / Brown
- Use, e.g., color temperatures to distinguish adjacent curves
- Use Opacity[*op*] to further adjust color style (0<*op*<1; 0: transparent; 1: opaque)
- Do not abuse colors – use colors to illustrate not to distract!

Plot Style

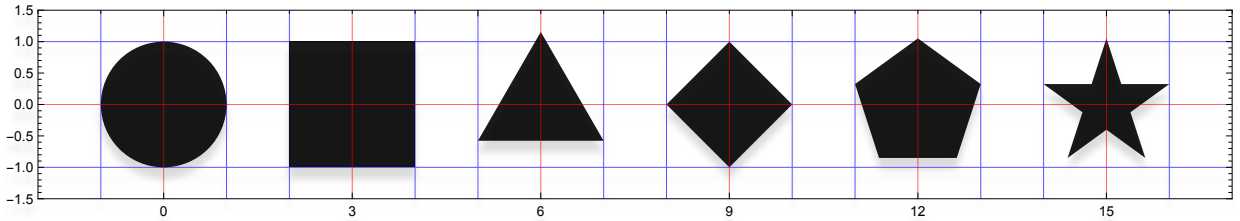
- PlotStyle->{*style_of_data*$_1$, *style_of_data*$_2$, *style_of_data*$_3$, …}
  - Global specification: PlotStyle->style
- Color (cf. Colors); Font, size and thickness (cf. Font and Size); …
- Dashing

- Filling, FillingStyle
- Directive[*style_dir*$_1$, *style_dir*$_2$, *style_dir*$_3$, …]
  - example: PlotStyle->{Directive[Thick, Dashed, Red, Opacity[0.6], EdgeForm[Thin]], Directive[…], …}
- Sampling points, Quality/efficienty,
- Singularity, special region,

Plot Markers

- Possible issues with the PlotMarkers
  - alignment issues with the font based plot markers (default choice)

  solution: use graphics based markers – customized or built-in, e.g., PlotMarkers/.Charting`ResolvePlotTheme["OpenMarkersThick",ListLinePlot]
  - size issues: markers appear visually different in size



Symbols with the same linear size appear visually different in size

  solution: adjust the size of the markers by hand to keep the areas roughly the same. (the visual appearance may vary from person to person)

- Open markers usually provide a balanced visual impression
  - use `Line[{{x1, y1}, {x2, y2}, …}]`
    - the gap at the vertices:
  - use filled polygon with `EdgeForm`
    - opacity, labeling the center, …

Plot Legend

- Place the legend (PlotLegens -> Placed[*legend*, {*x*, *y*}])
  - absolute: {0.1, 1.5}, {Log[0.01], 10}
  - relative: {Left, Bottom}, Scaled[{0.1, 0.9}]
  - legends of combined figures
- Styles of legend
  - LineLegend, PointLegend, SwatchLegend, BarLegend, …
- Elements
  - LabelStyle
  - LegendMarkers
  - LegendLayout
  - LegendFunction

Font and Size

- FrameStyle, PlotStyle, LabelStyle, … (cf. Style)
  - example: FrameStyle->Directive[Thickness[Medium], Black, FontSize->24, FontFamily->"Times"]
- FontSize (`FontSize->`*`s`*)
  - relative to the ImageSize
  - recommendation: for 640x480 (Medium, Large) figures, font size 16~32 is usually good
- FontFamily (`FontFamily->"font"`)
  - text-based vs professional fonts (Times, FreeSerif, …)
  - do not put too much texts and math expressions in a figure – use caption!
- Thickness (Thickness[*absolute_or_relative_size*], Thick, Thin)
  - the default frame lines are too thin (PDF vs PNG vs screen display)
  - recommendation: Thickness[Medium], Thickness[0.0025] for regular sized figure in PDF format
- FrameTicks (FrameTicks->{{*left_ticks*, *right_ticks*}, {*bottom_ticks*, *top_ticks*}})
  - tick marks by default is too small and too thin
  - each tick set (e.g., left_tick) is a list: { {tick_position$_1$, tick_label$_1$, {0, size$_1$}}, {…}, …}

Labels

- Labeling:
  - axes, frame axes, legends, curves, other necessary info, …
  - be concise: do not put too much texts and math expressions in a figure – use caption!
- Text and numbers (cf. Characters and Strings)
  - examples: "alpha = "<>ToString[alpha]<>" eV"; `NumberForm[number, {n, m}]`
- Symbols (e.g. Greek letters) and typesetting (cf. Symbols and Typesetting, Font and Size)
- Font, and size (cf. Font and size)
  - professional fonts (Times, FreeSerif) vs text-based fonts (Arial, Helvetica)
- Predefined vs customized
  - AxesLabel->{*horizonal_label*, *vertical_label*}, PlotLabel, FrameLabel, LegendLabel, …
  - Graphics[Text[Style[*texts_or_expressions*, *styles*], {*x*, *y*}]
  - Epilog->Inset[*graphics*,{pos_x, pos_y}] ; Labeled[*figure, labels, positions, spacing*]
- *Mathematica* cannot handle complicated typesetting—one often needs work-around
  - example 1: "\!\(*SuperscriptBox[\(10\), \(10\), \(" <>ToString[i]<> "\)\0\)" (cf. FullForm)
  - example 2: use pre-produced image for complex expressions

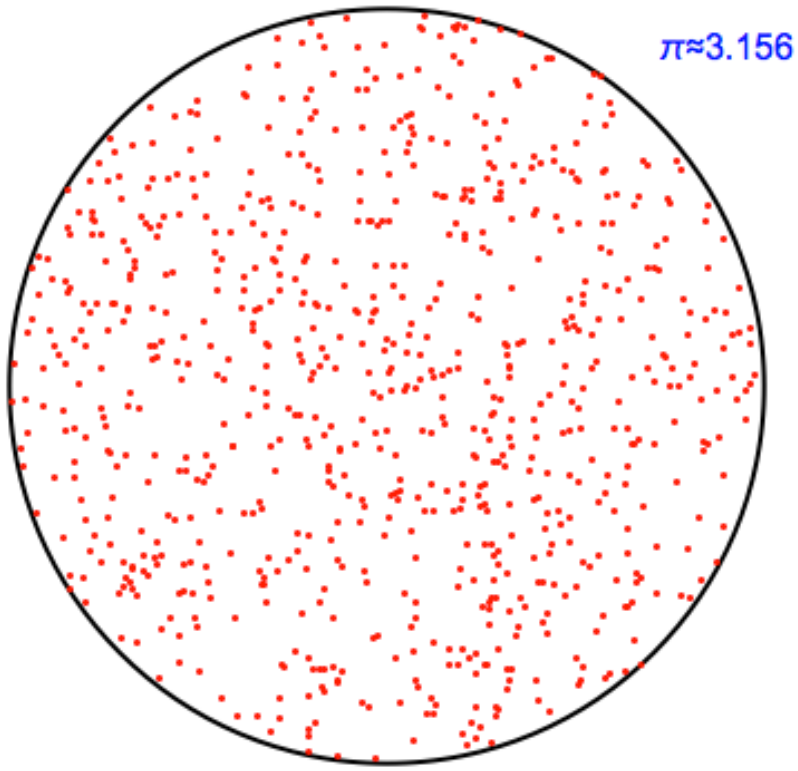
Miscellaneous

- Frame->True, Axes->False, otherwise, axes (without ticks) may still appear
- Adding arrows on axes: AxesStyle-> Directive[Arrowhead[Medium], …]
- Solid line is, in principle, equivalent to Dashing[{}], however, …
- PlotTheme->”Monochrome”

Graphics

- Graphics is used to show graphic primitives:
  - Line, Circle, Rectangle, Triangles, Polygon, Disk, Arrow, BSplines, BezierCurves, Parallelogram, Point, Text, …
- Styles are added in the same list of primitives
- Share many options with `Plot`, `ListPlot`
  - `Frame`, `Axes`, …
- Can be used to generate graphics elements

```
Graphics[
 {Black, Thick, Circle[],
  Red,
  Point[
   (pts = Select[RandomReal[{-1, 1}, {1000, 2}],
      Norm[#] ≤ 1 &])],Blue,
  Text[Style["π≈" <> ToString[Length[pts] / 1000. * 4],
    16], {0.9, 0.9}]}
]
```

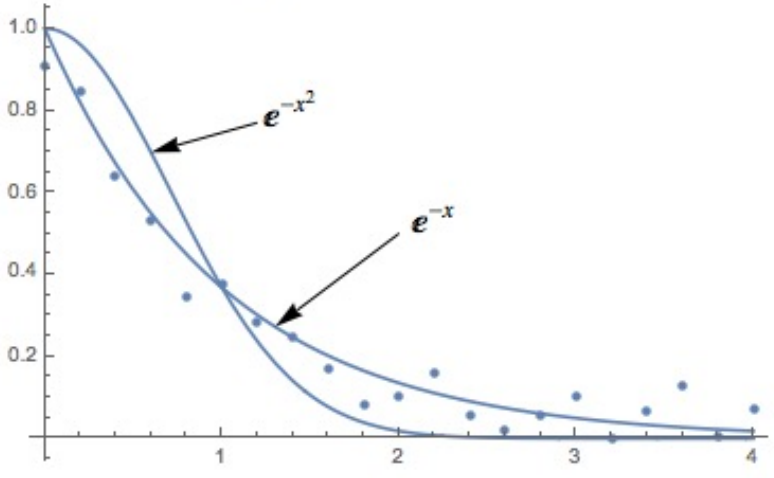


Combining multiple graphs

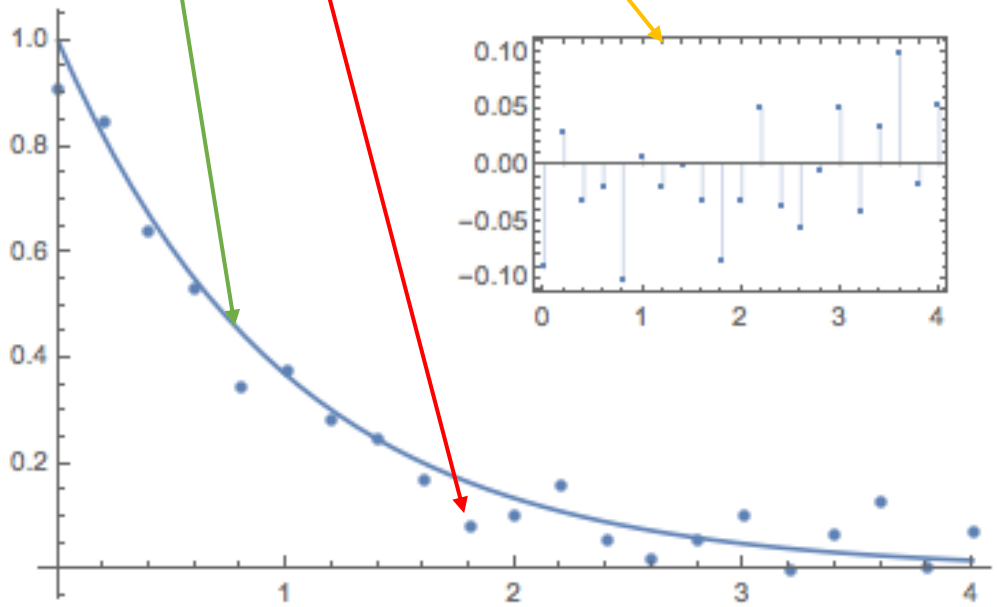
- Show[g_1, g_2, g_3, …, op_1, op_2, …] combines graphs g_i and applies options
  - graphs can be output of Plot, ListPlot, Graphics, …
  - canvas option priority: options of g_1 > options of Show > local options
  - overlay: frame line, axes beneath g_1 beneath g_2 beneath g_3 …
  - graphs share the same coordinate system – can also use Scaled coordinates
- Epilog and Inset
  - Plot[…, Epilog->{Inset[g_1, {x_1, y_1}], Inset[g_2, {x_2, y_2}], … }, …]
  - inset graphs do not share options or the coordinate system of the mother graph
- GraphicsRow, GraphicsColumn, GraphicsGrid
- Built-in decorative options
  - GraphicsLines, PlotLabel, PlotLegends, …

Examples:

```
Show[
 ListPlot[pts],
 Plot[Exp[-x], {x, 0, 4}],
 Plot[Exp[-x^2], {x, 0, 4}],
 Graphics[{Text[Style["e^-x", 16,
     FontFamily → "Times"], {2.2, .525}],
   Arrow[{{2, 0.5}, {1.3, E^-1.3}}],
   Text[Style["e^-x^2", 16,
     FontFamily → "Times"], {1.4, .8}],
   Arrow[{{1.2, 0.77}, {0.6, E^-0.36}}]}],
 PlotRange → All]
```



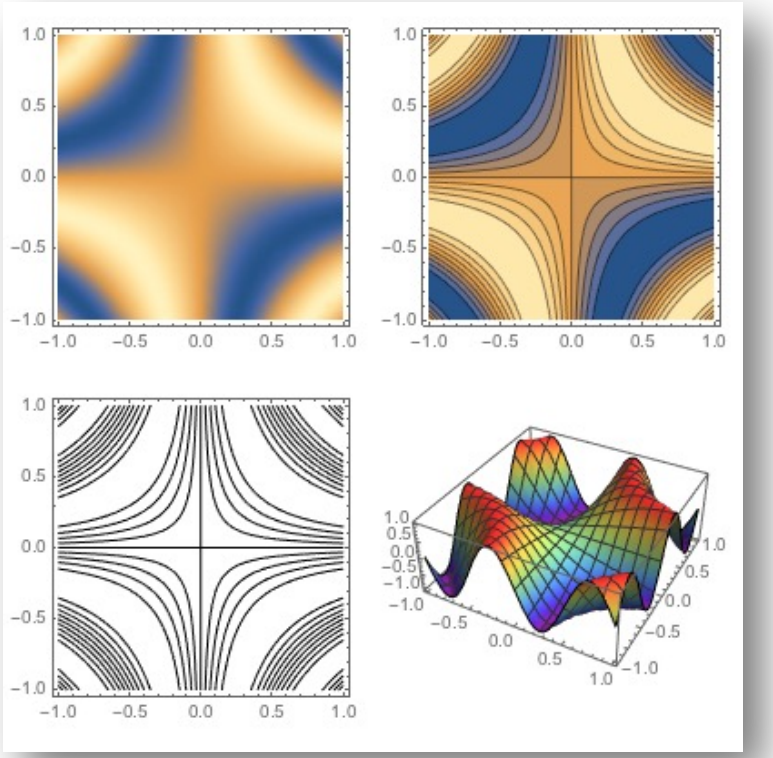

```
Show[
 ListPlot[pts, Epilog → Inset[
    ListPlot[Table[{pt[[1]], pt[[2]] - Exp[-pt[[1]]]},
      {pt, pts}], Frame → True, Filling → Axis],
    Scaled[{.7, .7}]]],
 Plot[Exp[-x], {x, 0, 4}],
 PlotRange → All
]
```



Two-variable visualization

- DensityPlot, Plot3D, ContourPlot, ParametricPlot
  - ListDensityPlot, ListPlot3D, ListContourPlot vs Interpolation

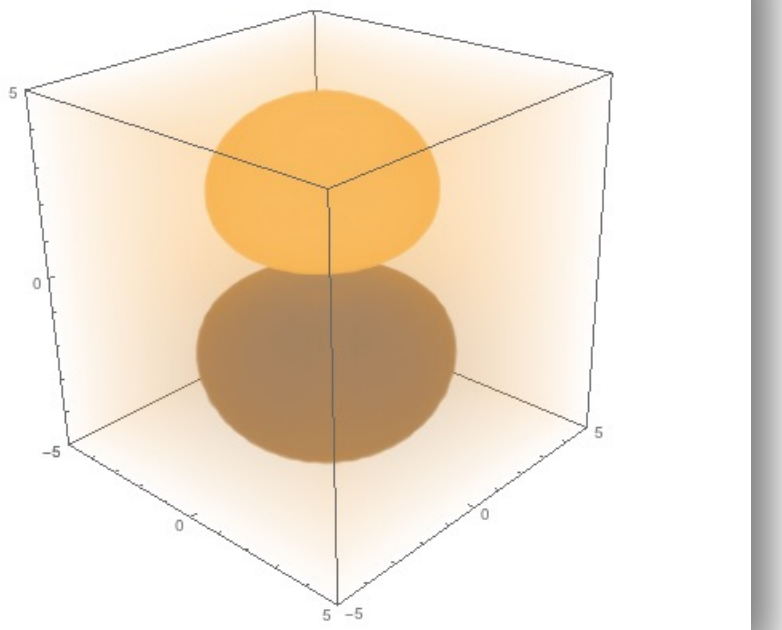
```
GraphicsGrid[
 {{DensityPlot[
    Sin[2 Pi * x * y], {x, -1, 1}, {y, -1, 1},
    ImageSize → Small
   ],
   ContourPlot[
    Sin[2 Pi * x * y], {x, -1, 1}, {y, -1, 1}
   ]},
  {ContourPlot[
    Sin[2 Pi * x * y], {x, -1, 1}, {y, -1, 1},
    ContourShading → None
   ],
   Plot3D[
    Sin[2 Pi * x * y], {x, -1, 1}, {y, -1, 1},
    ColorFunction → "Rainbow"
   ]}}]
```



Three-variable visualization

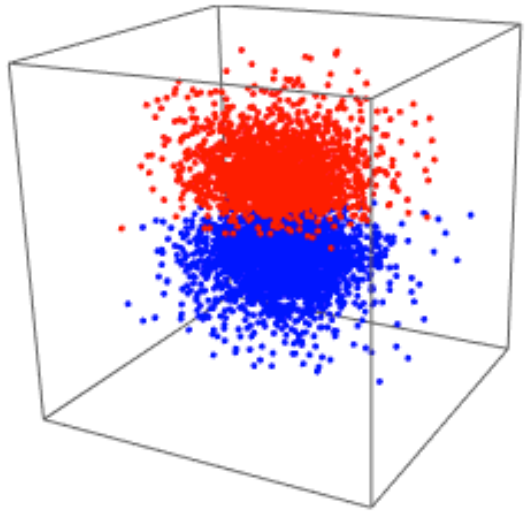
- ContourPlot3D, DensityPlot3D, SliceContourPlot3D, SliceDensityPlot3D, ParametricPlot3D
  - DensityPlot3D and *slice plots* require *Mathematica* 10 or later
- Use random points

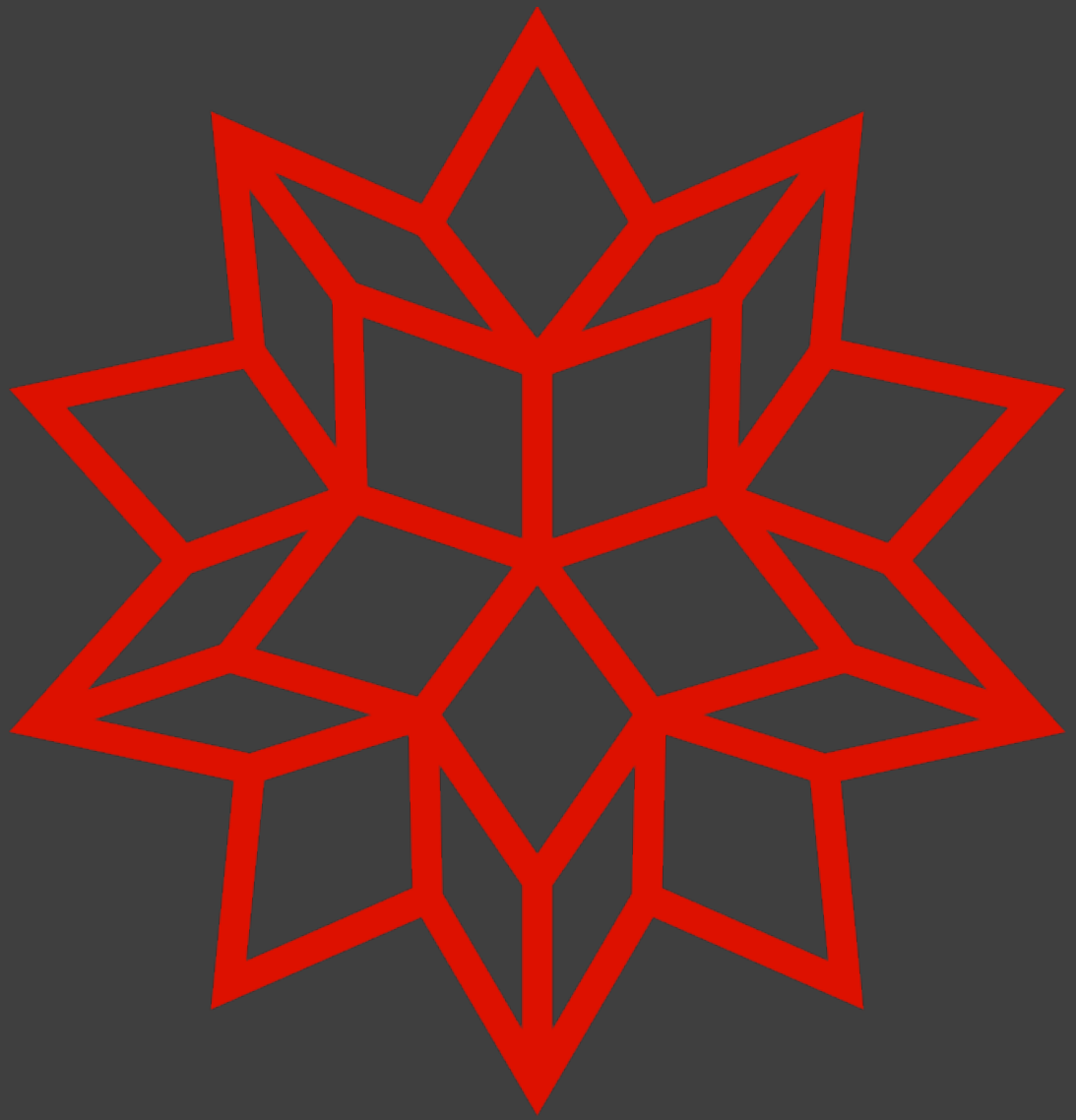
```
DensityPlot3D[ρ[x, y, z], {x, -5, 5}, {y, -5, 5}, {z, -5, 5}]
```




```
list = NestList[Metropolis, {0.1, 0.1, 0.1, .1}, 100 000];

Graphics3D[{Red, Point[Select[list[[10 000 ;;]], #[[3]] > 0 &][[1 ;;  ;; 20, 1 ;; 3]]], Blue,
   Point[Select[list[[10 000 ;;]], #[[3]] <= 0 &][[1 ;;  ;; 20, 1 ;; 3]]]}, BoxRatios → 1]
```





fin