

Lecture 5: Introduction to Mathematica IV

Graphics

Graphics are an important part of exploring mathematics and conveying its results. An informative plot or graphic that conveys a complex idea succinctly and naturally to an educated observer is a work of creative art. Indeed, art is sometimes defined as “an elevated means of communication,” or “the means to inspire an observation, heretofore unnoticed, in another.” Graphics are art; they are necessary. And, I think they are fun.

For graphics, we are limited to two- and three-dimensions, but, with the added possibility of animation, sound, and perhaps other sensory input in advanced environments, it is possible to usefully visualize more than three dimensions. Mathematics is not limited to a small number of dimensions; so, a challenge—or perhaps an opportunity—exists to use artfulness to convey higher dimensional ideas graphically.

The introduction to basic graphics starts with two-dimensional plots.

[3.016 Home](#)[Full Screen](#)[Close](#)[Quit](#)

Two-dimensional Plots I

notebook (non-evaluated)

pdf (evaluated)

html (evaluated)

Examples of simple x - y plots and how to decorate them.

- When `Plot` gets a list of expressions as its first argument, it will superpose the curves obtained from each. In this example, the y -variable's display is controlled with `PlotRange`, and the curves' colors and thicknesses are controlled with a list for `PlotStyle`. Note that the rule `PlotStyle` is a list of descriptions such as `Hue`, `RGBColor`, `Thickness`, `Dashing`, etc.
- To plot a curve of the form, $(x(t), y(t))$ as a function of a parameter t , `ParametricPlot` is called with its first argument being a list of x - and y -functions.
- Superposition of parametric plots is obtained with a list of two-member lists.
- With the physical constants package loaded, a function to convert degrees-Celsius to degrees Kelvin, and a function to calculate the Arrhenius function, an Arrhenius plot can be obtained with `ParametricPlot`.
- By naming a plot, it can be referenced and combined with more *Graphics Objects* with `Show`. In this case, a specialized “tick-scheme” is employed with “smart” labels for the $1/T$ axis.
- `LogPlot` needs `Needs["Graphics`"]`. Here is an example of an annually compounded bank account.

```

1 Plot[(Sin[x]/x, Tan[x]/x), {x, -5 Pi, 5 Pi},
   PlotRange -> {-0.25, 1.25}, PlotStyle ->
   {{Thickness[0.01], Hue[1]}, {Thickness[0.005], Hue[2/3]}}]

2 LuckyClover[t_, n_] :=
  {(1/(n+1)) (Cos[(n+1)t - Pi/4] - (n+1) Cos[t - Pi/4],
   Sin[(n+1)t - Pi/4] - (n+1) Sin[t - Pi/4])}

3 ParametricPlot[LuckyClover[t, 4], {t, 0, 2 Pi}, AspectRatio -> 1]

4 ParametricPlot[Evaluate[Table[LuckyClover[t, i], {i, 2, 7}]],
  {t, 0, 2 Pi}, AspectRatio -> 1]

5 ParametricPlot[Evaluate[Table[LuckyClover[t, i], {i, 2, 7}]],
  {t, 0, 2 Pi}, AspectRatio -> 1, PlotStyle ->
  Table[{Thickness[0.005], Hue[(2/3)*(i-2)/5]}, {i, 2, 7}]]

6 << Miscellaneous`PhysicalConstants`

7 Kelvin[TempCelsius_] := 273.15 + TempCelsius

8 Arrhenius[EnergyEV_, TempCelsius_] :=
  Exp[-(EnergyEV * Joule * ElectronCharge)/
    (Kelvin[TempCelsius] * BoltzmannConstant * Kelvin * Coulomb)]

9 ParametricPlot[1/Kelvin[T], Log[Arrhenius[1.0, T]]],
  {T, 0, 1000}]

10 arrhenplot = ParametricPlot[
  Evaluate[Table[{1/Kelvin[T], Log[Arrhenius[ev, T]]},
    {ev, 1, 5, 1}]], {T, -200, 1000}, PlotStyle -> Table[
    {Thickness[0.005], Hue[(2/3)*(5-i)/4]}, {i, 1, 5, 1}]]

11 Show[arrhenplot,
  Ticks -> {Table[{1/Kelvin[T], StringJoin["1/", ToString[T]]},
    {T, -200, 500, 100}], Automatic}]

12 BankAccount[InitialInvestment_,
  AnnualInterest_, NYears_] :=
  InitialInvestment * (1 + AnnualInterest/100)^NYears
Plot[BankAccount[100, 8.5, t], {t, 0, 50}]
Needs["Graphics`"]
LogPlot[BankAccount[100, 8.5, t], {t, 0, 50}]

```

3.016 Home



Full Screen

Close

Quit

Two-dimensional Plots II

notebook (non-evaluated)

pdf (evaluated)

html (evaluated)

Examples of incorporating data into x - y plots. Sometimes you will want to plot numbers that come from elsewhere—otherwise known as data. Presumably, data will be imported with file I/O.

- 2: The chemical elements and information about them is accessible via the package `Miscellaneous`ChemicalElements``. Here, this data will be used to make plots for the 1st–90th elements.
- 6: Subsequent to extracting the melting points by `Maping` the function `MeltingPoint` onto the element-list, the trends in melting point with atomic number is visualized. Using `PlotJoined` set to true in `ListPlot`, makes the trend more visible.
- 8: `Dens` and `mps` are each lists with 90 number-like objects. Therefore `{Dens, mps}` is a list of two lists—it has dimensions 2×90 . `ListPlot` will take data of the form $\{\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_N, y_N\}\}$, i.e., dimensions 90×2 . `Transform` will convert the data to the correct form for `ListPlot`.
- 10: By joining the data points with line-segments, a relationship between density and melting point becomes visible.

```
ListPlot, PieChart, Histogram, Barchart, etc
1 << Miscellaneous`ChemicalElements`
2 Elements
3 e190 = Elements[Table[i, {i, 1, 90}]]
4 mps = Map[MeltingPoint[#] &, e190] /. Kelvin -> 1
The next plot illustrates the variation of melting temperature as a
function of atomic number...
5 ListPlot[mps]
6 ListPlot[mps, PlotJoined -> True]
7 Dens = Map[Density[#] &, e190] /. {Kilogram -> 1, Meter -> 1}
The next line matches up values of density with melting
temperature...
8 dmdata = Transpose[{Dens, mps}]
9 ListPlot[dmdata]
10 ListPlot[dmdata, PlotJoined -> True]
```

3.016 Home



Full Screen

Close

Quit

Three Dimensional Graphics

notebook (non-evaluated)

pdf (evaluated)

html (evaluated)

It would be better to say, 3D graphics projected onto a 2D screen.

Using different **ViewPoints** and perspective, one can obtain informative 3D information on a screen. Unfortunately, MATHEMATICA®'s front end does not have the capability of spinning or flying-around 3D graphics (yet). But, such things are possible by exporting this information into other formats. An example of such can be found here: http://pruffle.mit.edu/~carter/talks/Stuttgart_INCEMS/node68.html

3.016 Home



Full Screen

Close

Quit

- 2: This is a function that computes the electrostatic potential over a 11×11 square-lattice of point-charges centered on the z -plane as a function of x , y , and z . In this example, some simpler methods of visualizing this four-dimensional object will be examined.
- 4: With sufficiently many **PlotPoints** the structure of the potential at a fixed distance $z = 0.25$ is made apparant.
- 5: Without recomputing all the data, the **ViewPoint** can be changed if the **SurfaceGraphics** object is assigned to a symbol that can be passed to **Show**. There is a handy *3D ViewPoint Selector* in MATHEMATICA®'s Input menu.
- 6: By computing *isopotentials* or *contours of constant potential* for $z = 0.25$, and using color. The **ContourPlot** function produces something like a *topographic map*.
- 7: The **ContourPlot** can be easily colored by setting the **ColorFunction**-option to **Hue**...
- 8: But, the **Hue** cycles from red to green to blue—and then back to red again. Here is a method to remove redundant colors.

```

Plot3D, ContourPlot, DensityPlot, etc

1 EPot[x_, y_, z_, xo_, yo_] :=
  1 /
  Sqrt[(x - xo)^2 + (y - yo)^2 + z^2]

2 SheetOLatticeCharge[x_, y_, z_] :=
  Sum[EPot[x, y, z, xo, yo], {xo, -5, 5}, {yo, -5, 5}]

SheetOLatticeCharge represents the electric field produced by
an 11 by 11 array of point charges arranged on the x-y plane at z
= 0. The following command evaluates and plots the field
variation in the plane z = 0.25:

3 Plot3D[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
  {x, -6, 6}, {y, -6, 6}]

Note below how theplot is set to contain the output of the Plot3D
command.

4 theplot = Plot3D[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
  {x, -6, 6}, {y, -6, 6}, PlotPoints -> 120]

Now we can adjust the viewpoint of theplot, without
recalculating the entire plot, using the Show command:

5 Show[theplot, ViewPoint -> {0, -5, 2}]

6 theconplot =
  ContourPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
  {x, -6, 6}, {y, -6, 6}, PlotPoints -> 120]

7 theconplot =
  ContourPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
  {x, -4, 4}, {y, -4, 4}, PlotPoints -> 120,
  ColorFunction -> Hue, Contours -> 24]

8 thedenplot =
  DensityPlot[Evaluate[SheetOLatticeCharge[x, y, 0.25]],
  {x, -4, 4}, {y, -4, 4}, PlotPoints -> 120,
  ColorFunction -> (Hue[1 - #*0.66] &)]

9 Show[thedenplot, Mesh -> False]

```

Graphics Primitives and Graphical Constructions

notebook (non-evaluated)

pdf (evaluated)

html (evaluated)

Examples of placing *Graphics Primitives* onto the display are developed and a tidy graphical demonstration of a *Wulff construction* is presented. Because PostScript is one of the graphics primitives, you can draw anything that can be imaged in another application. You can also import your own drawing and images into MATHEMATICA®.

- 3: A **Circle** is a graphics primitive. **Graphics** takes graphics primitives as arguments and converts them to a *graphics object*. **Show** takes the graphics object and sends it to the display. As MATHEMATICA® will pick an intersection for the axis display and a convenient scaling for both axes, **AxesOrigin** and **AspectRatio** should be specified in order to obtain the desired rendering.
- 5: Graphics primitives, such as **Text**, can be combined with two-dimensional plots to improve their *graphical content or exposition*.
- 6: The Wulff construction is a famous thermodynamic construction that predicts the equilibrium enclosing surface of an anisotropic isolated body. The anisotropic surface tension, $\gamma(\hat{n})$, is the amount of work (per unit area) required to produce a planar surface with outward normal $\hat{n}$. The construction proceeds by drawing a bisecting plane at each point of the polar plot $\gamma(\hat{n})\hat{n}$. The interior of all bisectors is the resulting *Wulff shape*.
- 8: This is an example $\gamma(\hat{n})$ with the surface tension being smaller in the $\langle 11 \rangle$ -directions.
- 9: By combining the graphics primitives from the *wulffline* function with the γ -plot, the equilibrium shape is visualized.

It can be useful to be able to build up arbitrary graphics objects piece-by-piece using simple "graphics primitives" like **Circle**:

```
1 Show[Graphics[Circle[2, 2], 1.5]]
2 Show[Graphics[Circle[2, 2], 1.5], Axes -> True]
3 Show[Graphics[Circle[2, 2], 1.5],
  Axes -> True, AxesOrigin -> {0, 0}, AspectRatio -> 1]
```

Now we take a simple plot...

```
4 cosplot = Plot[Cos[x], {x, 0, 4 Pi}]
```

and overlay some text in places of our own choosing...

```
5 Show[cosplot, Graphics[Text["One Wavelength", {2 Pi, 1.1}],
  Graphics[Text["Two Wavelengths", {4 Pi, 1.1}],
  PlotRange -> All]]
```

```
6 wulffline[{x_, y_}, wulfflength_] :=
Module[{theta, wulffhalflength = wulfflength * 0.5,
  x1, x2, y1, y2}, theta = ArcTan[x, y];
  x1 = x + wulffhalflength * Cos[theta + Pi/2];
  x2 = x + wulffhalflength * Cos[theta - Pi/2];
  y1 = y + wulffhalflength * Sin[theta + Pi/2];
  y2 = y + wulffhalflength * Sin[theta - Pi/2];
  Graphics[Line[{x1, y1}, {x2, y2}]]]
```

```
7 gammaplot[{theta_, anisotropy_, nfold_}] :=
(Cos[theta] + anisotropy * Cos[(nfold + 1) * theta],
Sin[theta] + anisotropy * Sin[(nfold + 1) * theta])
```

```
8 GammaPlot = ParametricPlot[
  gammaplot[t, 0.1, 4], {t, 0, 2 Pi}, AspectRatio -> 1,
  PlotStyle -> {{Thickness[0.005], RGBColor[1, 0, 0]}}]
```

```
9 Show[Table[wulffline[gammaplot[t, 0.1, 4], 2],
  {t, 0, 2 Pi, 2 Pi/100}], GammaPlot, AspectRatio -> 1]
```

3.016 Home



Full Screen

Close

Quit

Animation

notebook (non-evaluated)

pdf (evaluated)

html (evaluated)

A *random walk* process is an important concept in diffusion and other statistical phenomena. An animation of a 2D random walk process is developed.

- 1: This is a recursive function that simulates a random walk process. Each step in the random walk is recorded as a list structure, {iteration number, $\{x, y\}$, and assigned to *randomwalk* [iteration number]. For each step (or iteration), a number between 0 and $1/2$ is (for the magnitude of the displacement) and an angle between 0 and 2π (for the direction) are selected randomly from a uniform distribution.
- 2: This shows the history of a random walk after 50 iterations by using graphics primitives.
- 3: This will produce a sequence of images which can be grouped together and then collapsing the cell by double-clicking it. The collapsed cell can be animated by using a menu item under the Cell-menu. Here, the step is depicted with a number at the current position and a line segment to the subsequent position.
- 4: This is a similar animation, but the history of each previous step is included in the graphical display.

```

randomwalk[0] = {0, {0, 0}};
randomwalk[nstep_Integer?Positive] :=
  randomwalk[nstep] = {nstep, randomwalk[nstep - 1][[2]] +
    Random[Real, {0, 0.5}] *
    {Cos[theta = 2 Pi Random[]], Sin[theta]}}

Show[
  Table[Graphics[Text[ToString[randomwalk[[1]],
    randomwalk[[2]]], {i, 0, 50}],
  Table[Graphics[Line[{randomwalk[j - 1][[2]],
    randomwalk[j][[2]]}], {j, 1, 50}],
  PlotRange -> All, AspectRatio -> 1, AxesOrigin -> {0, 0}]

<< Graphics` Animation`
ShowAnimation[
  Table[
    Graphics[
      {Text[
        ToString[randomwalk[[1]],
        randomwalk[[2]]],
        Line[randomwalk[[2]], randomwalk[[1]][[2]]]}],
    {i, 0, 49}
  ],
  PlotRange -> {{-3, 3}, {-3, 3}},
  AspectRatio -> 1, AxesOrigin -> {0, 0}]

ShowAnimation[
  Table[
    Graphics[
      Table[
        {Text[
          ToString[randomwalk[[1]],
          randomwalk[[2]]],
          Line[randomwalk[[2]], randomwalk[[1]][[2]]]}],
        {j, 0, i}
      ],
    {i, 0, 49}
  ],
  PlotRange -> {{-3, 3}, {-3, 3}},
  AspectRatio -> 1, AxesOrigin -> {0, 0}]

```

3.016 Home



Full Screen

Close

Quit

Index

3D ViewPoint Selector, [40](#)

animation
of random walk, [42](#)

array of charges
visualization example, [40](#)

Arrhenius plots
functions to create, [38](#)

AspectRatio, [41](#)

AxesOrigin, [41](#)

chemical element data
examples of plotting, [39](#)

Circle, [41](#)

ColorFunction, [40](#)

colors
in contour plots, [40](#)

ContourPlot, [40](#)

contours of constant potential, [40](#)

Dashing, [38](#)

data
in plots, [39](#)

Example function
randomwalk, [42](#)
wulffline, [41](#)

Graphics, [41](#)
graphics

graphics primitives, [41](#)
graphics in mathematica
examples, [37](#)
graphics object, [41](#)
Graphics Objects, [38](#)
Graphics Primitives, [41](#)
Graphics', [38](#)

Hue, [38](#), [40](#)

isopotentials, [40](#)

ListPlot, [39](#)

LogPlot, [38](#)

Map, [39](#)

Mathematica function

AspectRatio, [41](#)
AxesOrigin, [41](#)
Circle, [41](#)
ColorFunction, [40](#)
ContourPlot, [40](#)
Dashing, [38](#)
Graphics, [41](#)
Hue, [38](#), [40](#)
ListPlot, [39](#)
LogPlot, [38](#)
Map, [39](#)
MeltingPoint, [39](#)
Needs, [38](#)

ParametricPlot, 38

PlotJoined, 39

PlotPoints, 40

PlotRange, 38

PlotStyle, 38

Plot, 38

RGBColor, 38

Show, 38, 40, 41

SurfaceGraphics, 40

Text, 41

Thickness, 38

Transform, 39

ViewPoint, 40

Mathematica package

Graphics', 38

Miscellaneous'ChemicalElements', 39

MeltingPoint, 39

Miscellaneous'ChemicalElements', 39

Needs, 38

ParametricPlot, 38

Plot, 38

PlotJoined, 39

PlotPoints, 40

PlotRange, 38

plots

changing the appearance of curves, 38

example using chemical elements, 39

from data, 39

log, 38

parametric 2D, 38

superposition of curves, 38

three dimensional

contours, 40

density, 40

two dimensions

examples, 38

options, 38

PlotStyle, 38

random walk, 42

randomwalk, 42

reducing Hue range in a ColorFunction, 40

RGBColor, 38

Show, 38, 40, 41

SurfaceGraphics, 40

Text, 41

Thickness, 38

topographic map, 40

Transform, 39

ViewPoint, 40

visualization example

random walk, 42

Wulff construction, 41

Wulff construction, 41

example mathematica function to draw, 41

Wulff shape, 41

wulffline, 41

[3.016 Home](#)



[Full Screen](#)

[Close](#)

[Quit](#)