6.4 Reworking Plots

2016-04-18

```
$Version
```

```
10.0 for Mac OS X x86 (64-bit) (September 10, 2014)
```

The size of a plot resulting from a command as described above may be changed in the process of editing. Also the position of the drawing within the area of the cell can be shifted. Such drawings may be stored as PDF-files; then they can be easily incorporated into TeX-Files so that one can display and print text, formulas and drawings in one step.

6.4.1 Changing the Size of Plot or its Position in a Cell

In order to change the size, guide the pointer to the corresponding cell; there appears a frame. Double click. Pressing the left button of the mouse and guiding the cursor, which assumes the shape of a cross consisting of two-pointed arrows, the image may be shifted. When the cursor is pointed to one of the small squares attached to the frame, it assumes the shape of a two-pointed arrow; guiding the mouse with the left button pressed down enlarges or diminishes the frame; thereafter a picture appears which accurately fits into the new frame. This may then be printed. These changes are lost as soon a the picture is cleared.

6.4.2 Use PDF-Files in Latex.

The plot may be stored in a pdf- file. This can be done in the following way: The file of the figure is best generated by a *Mathematica* graphics command attaching a name, e.g., "pict" to the output.This file is exported as a PDF-file by

Export["pictf.pdf", pict]

You will find this pdf-file in your home directory.

The Latex file must contain the two comands **\usepackage[]{} and \usepackage{color}** after the line with **\documentclass[]{}.**

```
\documentclass[11pt]{report}

\usepackage[final]{epsfig}                              <====
 \usepackage{color}                                     <====

.......

    \begin{figure}[h]
    \includegraphics[width = 12cm]{pictf}               <====
    \caption{text}
    \label{figxx}
    \end{figure}
or
    \begin{figure}[h]
    \includegraphics[height = 13cm]{pictf}              <====
    \caption{text}
    \label{figxx}
    \end{figure}
    ............

\end{document}
```

pictf is the name of the picture file. The width of the figure is the author's choice. "height" may be used in place of "width".

6.5 Some General Remarks on Options.

6.5.1 Getting Information on Options

Options for a certain *Mathematica* command can be printed by inputting the command preceeded by two question marks, e.g.

```
?? Integrate
```

Integrate[f, x] givestheindefiniteintegral $\int f\,dx$.

Integrate[f, $\{x, x_{min}, x_{max}\}$] givesthedefiniteintegral $\int_{x_{min}}^{x_{max}} f\,dx$.

Integrate[f, $\{x, x_{min}, x_{max}\}$, $\{y, y_{min}, y_{max}\}$, …] givesthemultipleintegral $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy \ldots f$. ≫

```
Attributes[Integrate] = {Protected, ReadProtected}

Options[Integrate] =
 {Assumptions :→ $Assumptions, GenerateConditions → Automatic, PrincipalValue → False}
```

Integrate[f, x] givestheindefiniteintegral $\int f\,dx$.

Integrate[f, $\{x, x_{min}, x_{max}\}$] givesthedefiniteintegral $\int_{x_{min}}^{x_{max}} f\,dx$.

Integrate[f, $\{x, x_{min}, x_{max}\}$, $\{y, y_{min}, y_{max}\}$, …] givesthemultipleintegral $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy \ldots f$.

Integrate[f, $\{x, y, \ldots\} \in reg$] integratesoverthegeometricregion reg. ≫

```
Attributes[Integrate] = {Protected, ReadProtected}

Options[Integrate] =
 {Assumptions :→ $Assumptions, GenerateConditions → Automatic, PrincipalValue → False}
Attributes[Integrate] = {Protected, ReadProtected}

Options[Integrate] :=
 {Assumptions :→ $Assumptions, GenerateConditions → Automatic, PrincipalValue → False}
```

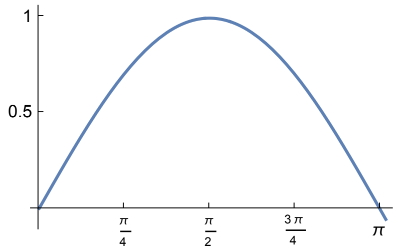
Options are also obtained with the help of the following command:

Options[*symbol*]	gives the list of default options assigned to a ***symbol*** (e.g.: ***NIntegrate[]***)
Options[*exp*]	gives the list of default options assigned to an ***expression*** (e.g.: a graphics object)
AbsoluteOptions[*expr*]	gives the absolute settings of options specified in an expression such as a graphics object.
AbsoluteOptions[*expr, name*]	gives the absolute settings of the specific option ***name***

```
Options[NIntegrate]

{AccuracyGoal → ∞, Compiled → Automatic,
 EvaluationMonitor → None, Exclusions → None, MaxPoints → Automatic,
 MaxRecursion → Automatic, Method → Automatic, MinRecursion → 0,
 PrecisionGoal → Automatic, WorkingPrecision → MachinePrecision}
```

```
bp = Plot[Sin[x], {x, 0, 3.2`},
            1
  Ticks → { ─ π {1, 2, 3, 4}, {0.5`, 1}}, ImageSize → 200]
            4
```



```
Options[bp]
```

```
                                                 1
{DisplayFunction → Identity, AspectRatio → ──────────────,
                                            GoldenRatio
 Axes → {True, True}, AxesLabel → {None, None}, AxesOrigin → {0, 0},
 DisplayFunction :→ Identity, Frame → {{False, False}, {False, False}},
 FrameLabel → {{None, None}, {None, None}},
 FrameTicks → {{Automatic, Automatic}, {Automatic, Automatic}},
 GridLines → {None, None}, GridLinesStyle → Directive[▨], ImageSize → 200,
 Method → {DefaultBoundaryStyle → Automatic, ScalingFunctions → None},
 PlotRange → {{0, 3.2}, {-0.0583741, 1.}}, PlotRangeClipping → True,
 PlotRangePadding → {{Scaled[0.02], Scaled[0.02]}, {Scaled[0.05], Scaled[0.05]}},
             π   π   3 π
 Ticks → {{ ─, ─, ───, π}, {0.5, 1}}}
             4   2    4
```

```
AbsoluteOptions[bp]
```

```
{AlignmentPoint → Center, AspectRatio → 0.618034,
 Axes → {True, True}, AxesLabel → {None, None}, AxesOrigin → {0., 0.},
 AxesStyle → {{■, AbsoluteThickness[0.25]}, {■, AbsoluteThickness[0.25]}},
 Background → None, BaselinePosition → Automatic, BaseStyle → {},
 ColorOutput → Automatic, ContentSelectable → Automatic,
 CoordinatesToolOptions → Automatic, DisplayFunction → Identity, Epilog → {},
 FormatType → TraditionalForm, Frame → {False, False, False, False},
 FrameLabel → {{None, None}, {None, None}}, FrameStyle → {{}, {}, {}, {}},
 FrameTicks → {None, None, None, None}, FrameTicksStyle → {},
 GridLines → {{}, {}}, GridLinesStyle → Directive[▨], ImageMargins → 0.,
 ImagePadding → All, ImageSize → 200., ImageSizeRaw → Automatic, LabelStyle → {},
 Method → {DefaultBoundaryStyle → Automatic, ScalingFunctions → None},
 PlotLabel → None, PlotRange → {{0., 3.2}, {-0.0583741, 1.}},
 PlotRangeClipping → True,
 PlotRangePadding → {{Scaled[0.02], Scaled[0.02]}, {Scaled[0.05], Scaled[0.05]}},
 PlotRegion → Automatic, PreserveImageOptions → Automatic,
 Prolog → {}, RotateLabel → True,
 Ticks → {{{0.785398, 0.785398, {0.00625, 0.}, {■, AbsoluteThickness[0.25]}},
     {1.5708, 1.5708, {0.00625, 0.}, {■, AbsoluteThickness[0.25]}},
     {2.35619, 2.35619, {0.00625, 0.}, {■, AbsoluteThickness[0.25]}},
     {3.14159, 3.14159, {0.00625, 0.}, {■, AbsoluteThickness[0.25]}}},
    {{0.5, 0.5, {0.00625, 0.}, {■, AbsoluteThickness[0.25]}},
     {1., 1., {0.00625, 0.}, {■, AbsoluteThickness[0.25]}}}}, TicksStyle → {}}
```

```
AbsoluteOptions[bp, PlotRange]
```

```
{PlotRange → {{0., 3.2}, {-0.0583741, 1.}}}
```

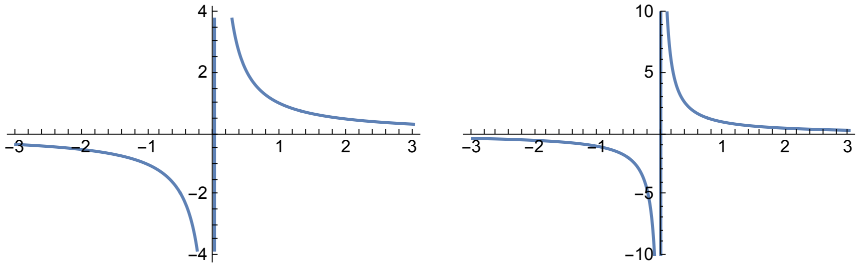
6.5.2 Setting Options.

Options may be inserted in commands for plots

```
Plot[],      ParametricPlot[],   ListPlot[], etc.
Plot3D[],    ParametricPlot3D[],     etc.
```

in the shape of rules as optional arguments. There are default values for each option. These are used as long as one does not assign an explicit value to a particular option in the plot statement just used. An explicit option replaces the default value of this option.

```
p1 = Plot[1/x, {x, -3, 3}];
p2 = Plot[1/x, {x, -3, 3}, PlotRange → {-10, 10}];
Show[GraphicsRow[{p1, p2}], ImageSize → 450]
```



In the left drawing (p1) the default value (**Automatic**) for the option **PlotRange** is used. This option determines the range of the scale of the ordinate. The default value **Automatic** of this option induces a built-in routine to choose the range. In the drawing at the right hand side (p2) the plot range is selected at the will of the user.

The present status of the default values assigned to the various options is shown when the particular plot command is preceeded by two question marks.

```
?SetOptions
```

SetOptions[*s*, *name*$_1$ → *value*$_1$, *name*$_2$ → *value*$_2$, …] sets the specified default options for a symbol *s*.
SetOptions[*stream*, …] or SetOptions["*name*", …] sets options associated with a particular stream.
SetOptions[*object*, …] sets options associated with an external object such as a NotebookObject. >>

The default values for the options may be changed by the commands above :

```
?? Plot
```

Plot[f, {x, x_{min}, x_{max}}] generatesa plotof f as a functionof x fromx_{min} to x_{max}.
Plot[{f_1, f_2, …}, {x, x_{min}, x_{max}}] plotsseveralfunctionsf_i. »

```
Attributes[Plot] = {HoldAll, Protected, ReadProtected}

                                                       1
Options[Plot] = {AlignmentPoint → Center, AspectRatio → ——————————— , Axes → True,
                                                     GoldenRatio
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
  EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
  Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm,
  Frame → False, FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic,
  FrameTicksStyle → {}, GridLines → None, GridLinesStyle → {}, ImageMargins → 0.,
  ImagePadding → All, ImageSize → Automatic, ImageSizeRaw → Automatic,
  LabelStyle → {}, MaxRecursion → Automatic, Mesh → None, MeshFunctions → {#1 &},
  MeshShading → None, MeshStyle → Automatic, Method → Automatic,
  PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None,
  PlotPoints → Automatic, PlotRange → {Full, Automatic}, PlotRangeClipping → True,
  PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic, Prolog → {},
  RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

6.5.2.1 Example for a fixed and a running option

An example involving both: 1. Setting a new option (yellow background) ;
2. An option coupled to the running index of a list of drawings.

```
SetOptions[Plot, Background -> Yellow]

                                               1
{AlignmentPoint → Center, AspectRatio → ——————————— , Axes → True,
                                           GoldenRatio
 AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → 🟨,
 BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
 ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
 ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
 DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
 EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
 Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm,
 Frame → False, FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic,
 FrameTicksStyle → {}, GridLines → None, GridLinesStyle → {}, ImageMargins → 0.,
 ImagePadding → All, ImageSize → Automatic, ImageSizeRaw → Automatic,
 LabelStyle → {}, MaxRecursion → Automatic, Mesh → None, MeshFunctions → {#1 &},
 MeshShading → None, MeshStyle → Automatic, Method → Automatic,
 PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None,
 PlotPoints → Automatic, PlotRange → {Full, Automatic}, PlotRangeClipping → True,
 PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
 PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic, Prolog → {},
 RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
 Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}

curves  = Table[Sin[n x], {n, 3}]

{Sin[x], Sin[2 x], Sin[3 x]}
```

```
plosty = Thread[Table[PlotStyle, {3}] → Table[Dashing[{3 (n - 1) 0.01}], {n, 3}]]
```

```
{PlotStyle → Dashing[{0.}],
 PlotStyle → Dashing[{0.03}], PlotStyle → Dashing[{0.06}]}
```

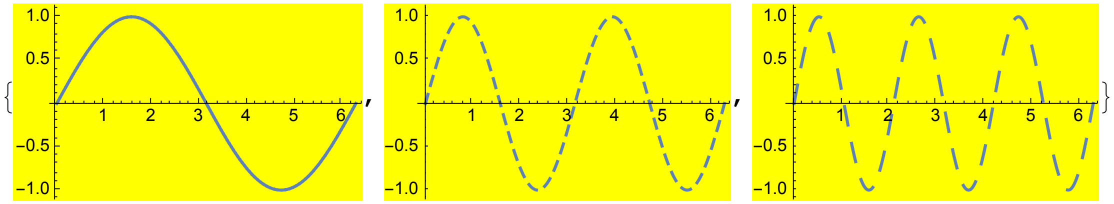
```
Table[Plot[curves[[n]], {x, 0, 2 π}, plosty[[n]]], {n, 3}]
```

Error message:

Plot::nonopt
  Options expected (instead of plosty⟦n⟧) beyond position 2 in Plot[curves⟦n⟧, {x, 0, 2π}, plosty⟦n⟧]. An option must
        be a rule or a list of rules ≫

Remedy: Insert option (here: curves⟦n⟧) into command **Evaluate[]**

```
Table[Plot[curves[[n]], {x, 0, 2 π}, Evaluate[plosty[[n]]] ], {n, 3}]
```



```
SetOptions[Plot, Background →  None];
```

6.5.3 Style Directives

There are several options, whose name ends with the word **Style** as, e.g.,

AxesStyle, BoxStyle, FrameStyle, PlotStyle,...

which influence the style of the drawing or that of a particular part. For example, in **Plot PlotStyle** will influence the features of the curves, their thickness, color, continuity (dashing). To specify the style one uses the following **style directives:**

`AbsoluteDashing[`*list*`]` determines the dashing of lines. The units used in *list* are in printers points; 1 pt ≈ 0.35 mm.

`____________________` ***iist = {}***

`--------------------` ***list = {3,1}***

`-.-.-.-.-.-.-.-.-.-.-` ***list = {3,0.5,0.1,0.5}***

All elements of ***list*** are used; then the cycle starts again.

`AbsoluteThickness[`*num*`]` determines the thickness of lines.
The unit used in ***num*** is in printers points; 1 pt ≈ 0.35 mm.

`Dashing[`*list*`]` determines the dashing of lines. The units used in ***list*** are in fractions of the total length of the drawing; so 0.01 is 1 percent of the this length.

`Thickness[`*num*`]` determines the thickness of lines. The unit used in ***num*** is fractions of the total length of the drawing;

`Thin` Plot a rather thin line

`Thick` Plot a thick line

`RGBColor[]` see § 6.5.4
`Hue`
`Graylevel[]`

A style directives may be a list comprising several of the above directives, e.g., a directive may prescribe the thickness,
the dashing and the color of a curve.

```
PlotStyle -> {Thickness[0.01], Dashing[{0.05}], RGBColor[1,0,0]}
```

If the plot command contains a list of curves to be ploted, the style directive may contain a correpending list, so that each curve gets its own characteristics. If the list of style directives is shorter than that of the curves, the style list is used again from its beginning. So for 3 curves, each having its own characteristics, one has:

```
    PlotStyle ->  { Dashing[{}], Dashing[{0.01}], Dashing[{0.02}] }
```

6.5.4 GrayLevel and Color Codes

In the style directives described in the preceeding section as well as in some other commands as e.g.

AmbientLight, Background, DefaultColor, ...

Graylevel or color codes are used. These will be described in this section.

6.5.4.1 GrayLevel

GrayLevel[*num***]**	*0 ≤ num ≤ 1*	determines the shade of gray of a line, curve or area.
GrayLevel[*0* **]**	black	
GrayLevel[*1* **]**	white	

6.5.4.2 Colors

Colors (including white and black) are obtained by mixing three or four fundamental colors. There are two types of mixing, additive and subtractive.

Additive mixing

is used on color screens. The fundamental colors are Red, Green, Blue. Thus any color may be coded in the command:

RGBColor[*numr,numg,numb***]**	*numr*	determines the amount of red in the mixture.
	numg	determines the amount of green in the mixture.
0 ≤ *numi* ≤ 1,	*numb*	determines the amount of blue in the mixture.

All colors can be obtained by mixing these three basic colors. For example:

RGBColor[1, 1, 0] yellow

Subtractive mixing

is used in slides and prints on paper. The four fundamental colors are Cyan, Magenta, Yellow, Black. The first three colors are just the colors complementary to the fundamental colors used in additive mixing.

6.5.4.3 Hue

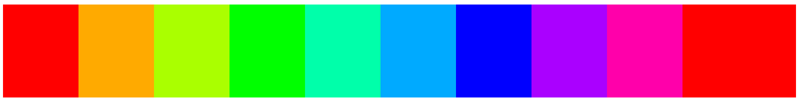
Hue[*h*] is a graphics directive which specifies that graphical objects should be displayed in a color corresponding to *h* .

0 ≤ h ≤ 1 Values outside this range are clipped, i.e. only the decimal part of h is considered.

	Red	Yellow	Green	Blue	Violet	Red
h =	0	0.16	0.4	0.56	0.68	1

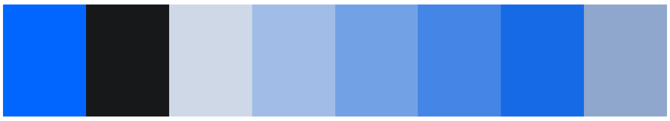
```
nu = Table[k / 9 // N, {k, 0, 9}];
Print["   ", NumberForm[nu, {3, 2}] ]
Show[Graphics[Table[{Hue[k/9], Rectangle[{k, 0}, {k + 1.5, 1}]}, {k, 0, 9}]],
 AspectRatio → 0.15`, ImageSize → 430]
```

```
 {0.00, 0.11, 0.22, 0.33, 0.44, 0.56, 0.67, 0.78, 0.89, 1.00}
```



Hue[*h*, *s*, *b*]   0 ≤ s ≤ 1 specifies the saturation, 0 ≤ b ≤ 1 the brightness of the color.

```
Show[Graphics[{Hue[0.6`], Rectangle[{0, 0}, {1, 1}],
   Hue[0.6`, 0.1`, 0.1`], Rectangle[{1, 0}, {2, 1}], Hue[0.6`, 0.1`, 0.9`],
   Rectangle[{2, 0}, {3, 1}], Hue[0.6`, 0.3`, 0.9`], Rectangle[{3, 0}, {4, 1}],
   Hue[0.6`, 0.5`, 0.9`], Rectangle[{4, 0}, {5, 1}], Hue[0.6`, 0.7`, 0.9`],
   Rectangle[{5, 0}, {6, 1}], Hue[0.6`, 0.9`, 0.9`], Rectangle[{6, 0}, {7, 1}],
   Hue[0.6`, 0.3`, 0.8`], Rectangle[{7, 0}, {8, 1}]}], AspectRatio → 0.2`]
```

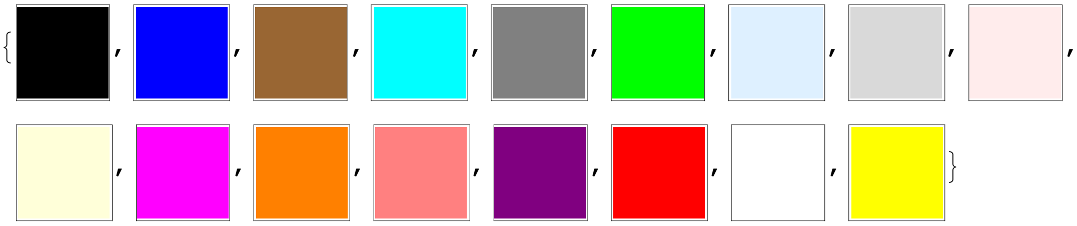


6.5.4.4 Colors

17 Colors can be assigned by their name. These are given in the following list:

```
colist =
 {Black,
Blue,
Brown,
Cyan,
Gray,
Green,
LightBlue,
LightGray,
LightPink,
LightYellow,
Magenta,
Orange,
Pink,
Purple,
Red,
White,
Yellow}
```

```
Table[Show[Graphics[{colist[[k]], Rectangle[{k - 1, 0}, {k, 1}]}],
  ImageSize → 50, Frame → True, FrameTicks → None], {k, Length[colist]}]
```



6.5.4.5 A list of all color names

There is a package with many more colors in the kernel. Their names are given in the following list. These colors can be called
by their RGBColorCode after the laters has ben found via **ColorData["Legacy","ColorName"]** .
The color names can be found in the following list.

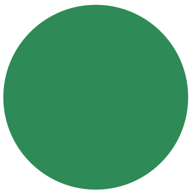
```
OldColorNames =
{"AliceBlue", "AlizarinCrimson", "Antique", "Apricot", "Aquamarine",
   "AureolineYellow", "Azure", "Banana", "Beige", "Bisque", "Black",
   "BlanchedAlmond", "Blue", "BlueViolet", "Brick", "Brown", "BrownMadder",
   "BrownOchre", "Burlywood", "BurntSienna", "BurntUmber", "CadetBlue",
   "CadmiumLemon", "CadmiumOrange", "CadmiumYellow", "Carrot", "Cerulean",
   "Chartreuse", "Chocolate", "ChromeOxideGreen", "CinnabarGreen", "Cobalt",
   "CobaltGreen", "ColdGray", "Coral", "CornflowerBlue", "Cornsilk", "Cyan",
   "CyanWhite", "DarkGoldenrod", "DarkGreen", "DarkKhaki", "DarkOliveGreen",
   "DarkOrange", "DarkOrchid", "DarkSeaGreen", "DarkSlateBlue", "DarkSlateGray",
   "DarkTurquoise", "DarkViolet", "DeepCadmiumRed", "DeepCobaltViolet",
   "DeepMadderLake", "DeepNaplesYellow", "DeepOchre", "DeepPink",
   "DeepSkyBlue", "DimGray", "DodgerBlue", "Eggshell", "EmeraldGreen",
   "EnglishRed", "Firebrick", "Floral", "ForestGreen", "Gainsboro",
   "GeraniumLake", "Ghost", "Gold", "Goldenrod", "GoldOchre", "Gray", "Green",
   "GreenishUmber", "GreenYellow", "Honeydew", "HotPink", "IndianRed",
   "Indigo", "Ivory", "IvoryBlack", "Khaki", "LampBlack", "Lavender",
   "LavenderBlush", "LawnGreen", "LemonChiffon", "LightBeige", "LightBlue",
   "LightCadmiumRed", "LightCadmiumYellow", "LightCoral", "LightGoldenrod",
   "LightGray", "LightPink", "LightSalmon", "LightSeaGreen", "LightSkyBlue",
   "LightSlateBlue", "LightSlateGray", "LightSteelBlue", "LightViridian",
   "LightYellow", "LimeGreen", "Linen", "Magenta", "ManganeseBlue", "Maroon",
   "MarsOrange", "MarsYellow", "MediumAquamarine", "MediumBlue", "MediumOrchid",
   "MediumPurple", "MediumSeaGreen", "MediumSlateBlue", "MediumSpringGreen",
   "MediumTurquoise", "MediumVioletRed", "Melon", "MidnightBlue", "Mint",
   "MintCream", "MistyRose", "Moccasin", "Navajo", "Navy", "NavyBlue",
   "Oak", "OldLace", "Olive", "OliveDrab", "Orange", "OrangeRed", "Orchid",
   "PaleGoldenrod", "PaleGreen", "PaleTurquoise", "PaleVioletRed", "PapayaWhip",
   "Peach", "PeachPuff", "Peacock", "PermanentGreen", "PermanentRedViolet",
   "Peru", "Pink", "Plum", "PowderBlue", "PrussianBlue", "Purple", "Raspberry",
   "RawSienna", "RawUmber", "Red", "RoseMadder", "RosyBrown", "RoyalBlue",
   "SaddleBrown", "Salmon", "SandyBrown", "SapGreen", "SeaGreen", "Seashell",
   "Sepia", "Sienna", "SkyBlue", "SlateBlue", "SlateGray", "Smoke", "Snow",
   "SpringGreen", "SteelBlue", "TerreVerte", "Thistle", "Titanium", "Tomato",
   "Turquoise", "TurquoiseBlue", "Ultramarine", "UltramarineViolet",
   "VanDykeBrown", "VenetianRed", "Violet", "VioletRed", "WarmGray", "Wheat",
   "White", "Yellow", "YellowBrown", "YellowGreen", "YellowOchre", "Zinc"};
```

The RCB code for a color given in the list above is obtained in the following way:

```
ColorData["Legacy", "SeaGreen"]
```

🟩

```
Show[Graphics[{%, Disk[{0, 0}, 1]}], ImageSize → 100]
```



6.5.4.6 Opacity

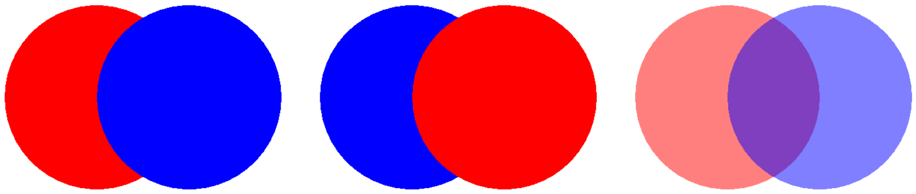
```
? Opacity
```

```
Opacity[a] is a graphicsdirectivewhichspecifies
     thatgraphicalobjectswhichfollowaretobe displayed,if possible,withopacitya.
Opacity[a, color] usesthespecifiedcolorwithopacitya. >>
```

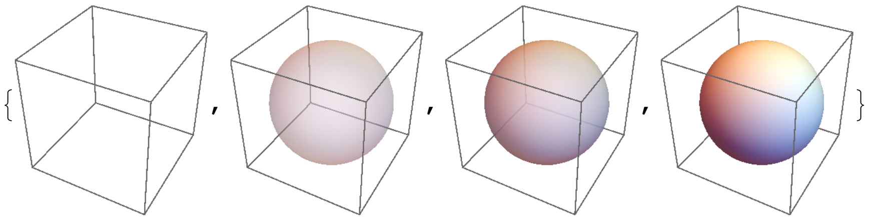
Opacity[0] = Invisble

Opacity[1] = Intransparent = opaque

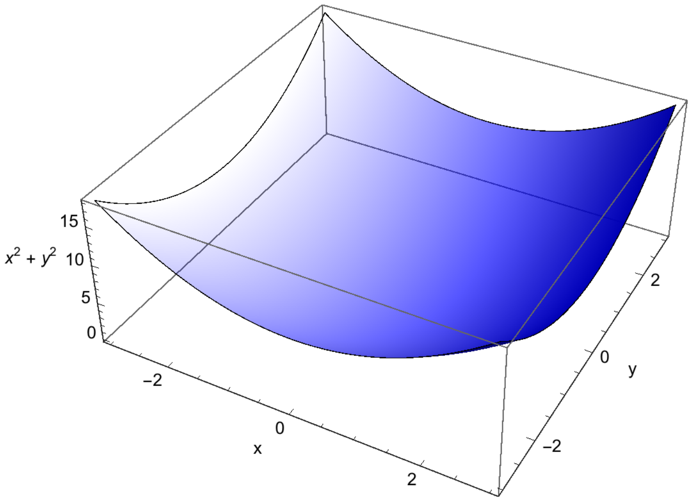
```
p1 = Graphics[{Red, Disk[], Blue, Disk[{1, 0}]}, ImageSize → 150];
p2 = Graphics[{Blue, Disk[], Red, Disk[{1, 0}]}, ImageSize → 150];
p3 = Graphics[{Opacity[0.5, Red], Disk[],
    Opacity[0.5, Blue], Disk[{1, 0}]}, ImageSize → 150];
GraphicsRow[{p1, p2, p3}]
```



```
Table[Graphics3D[{Opacity[a], Sphere[]}, ImageSize → 100], {a, 0, 1, 1 / 3}]
```



```
Plot3D[x^2 + y^2, {x, -3, 3}, {y, -3, 3}, AxesLabel → {"x", "y", "  x^2 + y^2"},
 ColorFunction → (Directive[Opacity[#], Blue] &), PlotPoints → 40, Mesh → None]
```



6.5.5 Label

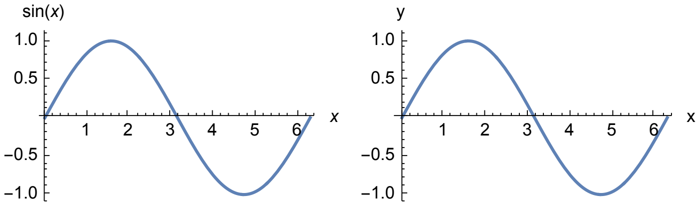
There are several options, whose name ends with the word **`Label`** as, e.g.,

AxesLabel, FrameLabel, PlotLabel

which are used to attach text to the coordinate axes, to an (optional frame) or as a head of a drawing. Text enclosed by quotation marks (= String) is treated as a pure, isolated text. Other text may be influenced by previous definitions.

```
SetOptions[Plot, Background -> None];
```

```
y = Sin[x];
p1 = Plot[Sin[x], {x, 0, 2 π}, AxesLabel → {x, y}];
p2 = Plot[Sin[x], {x, 0, 2 π}, AxesLabel → {"x", "y"}];
Show[GraphicsRow[{p1, p2}]]
```



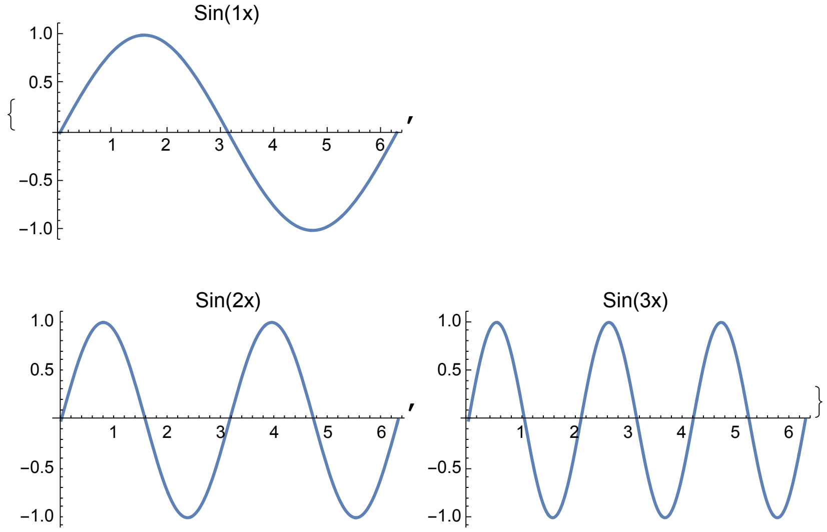
6.5.5.1 Row

The text of a label may consist of several parts of different nature, e.g. of strings, of common numbers (e.g.running indices)
or of information depending on previous input or output. For this, the command
`Row[`*Exp1,Exp2,Exp3,...*`]` is needed; without this the output will be scrambled.

```
? Row
```

Row[{*expr*$_1$, *expr*$_2$, …}] is an objectthatformatswiththe *expr$_i$* arrangedin a row, potentiallyextendingoverseverallines
Row[*list*, *s*] inserts*s* as a separatorbetweensuccessiveelements »

```
Table[Plot[Sin[n x], {x, 0, 2 π},
  PlotLabel → Row[{"Sin(", n, "x)"}], ImageSize → 200], {n, 3}]
```



```
Row[Range[50], "."]
```

```
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.21.22.23.24.25.26.27.
 28.29.30.31.32.33.34.35.36.37.38.39.40.41.42.43.44.45.46.47.48.49.50
```

```
Row[Range[50], "+"]
```

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 +
 18 + 19 + 20 + 21 + 22 + 23 + 24 + 25 + 26 + 27 + 28 + 29 + 30 + 31 + 32 + 33 +
 34 + 35 + 36 + 37 + 38 + 39 + 40 + 41 + 42 + 43 + 44 + 45 + 46 + 47 + 48 + 49 + 50
```

This is neither an expression nor a string as appears in the following:

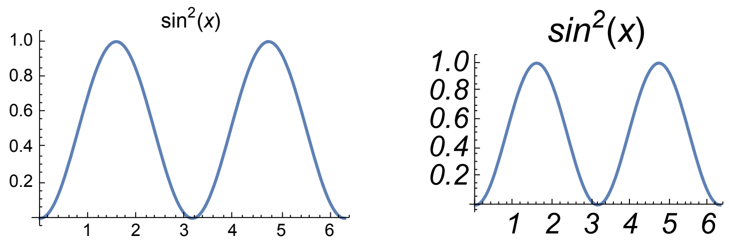
```
ToExpression[%]
```

```
ToExpressionnotstrbox:
  1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+26+27+28+29+30+3
        1+32+33+34+35+36+37+38+39+40+41+42+43+44+45+46+47+48+49+50 is
     not a string or a box ToExpression can only interpret strings or boxes as Wolfram Language input ≫
$Failed
```

6.5.6 Choice of Fonts

BaseStyle = rule or list of rules an option for the text style in a particular graphic

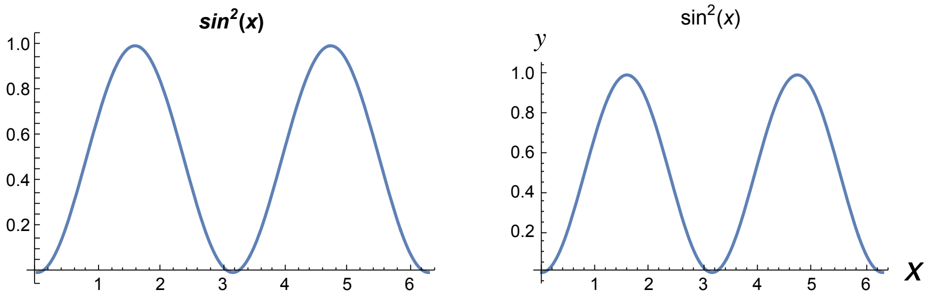
```
p1 = Plot[Sin[x]^2, {x, 0, 2 π}, PlotLabel → Sin[x] ^2];
p2 = Plot[Sin[x]^2, {x, 0, 2 π}, PlotLabel → Sin[x]^2,
    BaseStyle → {FontSlant → Italic, FontSize → 15}];
Show[GraphicsRow[{p1, p2}], ImageSize → 400]
```



FontSize	**-> n**	the size of font to use in printer's points
FontSlant	**-> "Italic"**	use an italic font
FontWeight	**-> "Bold"**	use a bold font
FontFamily	**-> "name"**	specify the nam of the font family to use (e.g. "Times", "Courier", "Helvetica")

The style of a text may also be defined locally by the command **Style[].**

```
Clear[x, y]
p1 = Plot[Sin[x]^2, {x, 0, 2 π}, PlotLabel →
    Style[Sin[x]^2, FontSize → 11, FontSlant → "Italic", FontWeight → Bold]];
p2 = Plot[Sin[x]^2, {x, 0, 2 π}, PlotLabel → sin^2(x),
   AxesLabel → {Style[x, FontSize → 18, FontFamily → "Helvetica"],
     Style[y, FontSize → 14, FontFamily → "Times"]}];
Show[GraphicsRow[{p1, p2}], ImageSize → 500]
```



```
? Style
```

Style[*expr*, *options*] displayswith*expr* formattedusingthespecifiedoptionsettings
Style[*expr*, "*style*"] usestheoptionsettingsforthespecifiedstylein thecurrentnotebook
Style[*expr*, *color*] displaysusingthespecifiedcolor
Style[*expr*, Bold] displayswithfontsmadebold
Style[*expr*, Italic] displayswithfontsmadeitalic
Style[*expr*, Underlined] displayswithfontsunderlined
Style[*expr*, Larger] displayswithfontsmadelarger
Style[*expr*, Smaller] displayswithfontsmadesmaller
Style[*expr*, *n*] displayswithfontsize *n*.
Style[*expr*, Tiny], Style[*expr*, Small], etc. displaywithfontsthataretiny, small, etc. >>

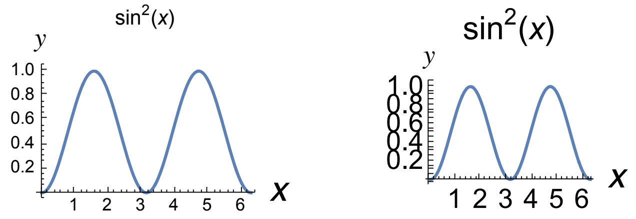
```
? BaseStyle
```

BaseStyleis an optionforformattingandrelatedconstructsthatspecifiesthebasestyletouseforthem >>

The option **Style** above permits one to choose the individual stylel of each letter or lettering in a plot. The option **BaseStyle** may be used to change the style of **all** lettering in an existing plot.

```
p3 = Show[p2, BaseStyle → {FontSize → 14}];

Show[GraphicsRow[{p2, p3}]]
```



6.6 Arrays of Graphics

Several figures are drawings may be combined by the command

`GraphicsRow[list]`	One row of pictures
`GraphicsGrid[list]`	An array of pictures

```
? GraphicsRow
```

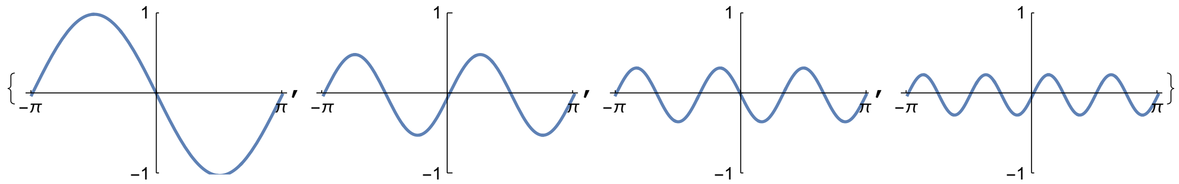
GraphicsRow[{g_1, g_2, …}] generates a graphic in which the g_i are laid out in a row.
GraphicsRow[*list*, *spacing*] leaves the specified spacing between successive elements »

```
? GraphicsGrid
```

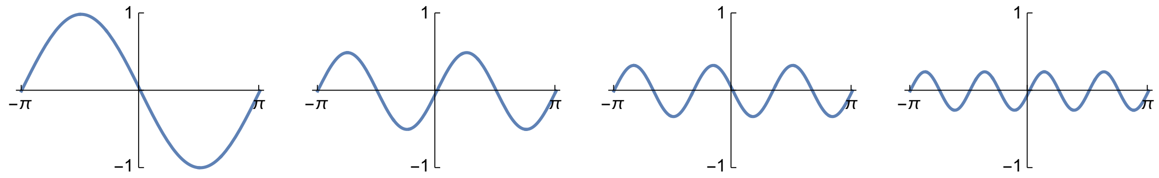
GraphicsGrid[{{g_{11}, g_{12}, …}, …}] generates a graphic in which the g_{ij} are laid out in a two-dimensional grid »

The argument is a list comparable to that for a matrix, whose elements are the names of the Graphics objects.

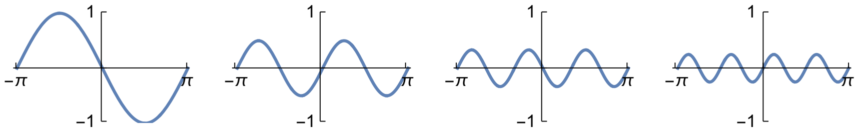
```
pp = Table[Plot[(-1)^n Sin[n x]/n,{x,-Pi,Pi}, ImageSize -> 140,
     PlotRange -> {-1,1}, Ticks -> {{-Pi,0,Pi}, {-1,0,1} }],{n,4} ]
```



```
GraphicsRow[pp]
```

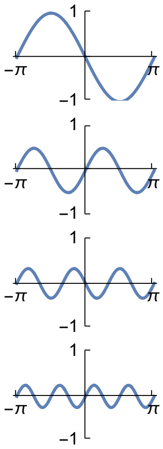


```
Show[GraphicsRow[pp], ImageSize → 450]
```

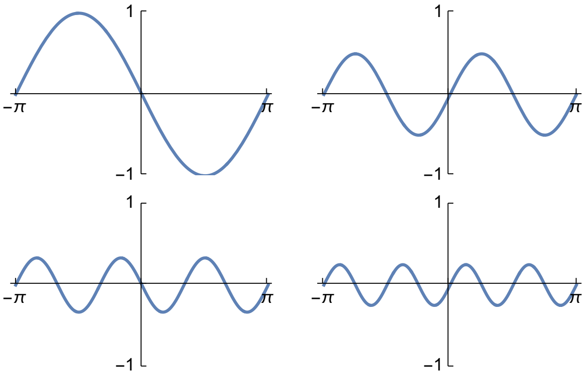


```
pt = Table[{pp[[k]]}, {k,Length[pp] }] ;
```

```
Show[GraphicsGrid[pt], ImageSize → 100]
```



```
pp22 = {{pp⟦1⟧, pp⟦2⟧}, {pp⟦3⟧, pp⟦4⟧}}; p1 = Show[GraphicsGrid[pp22]]
```



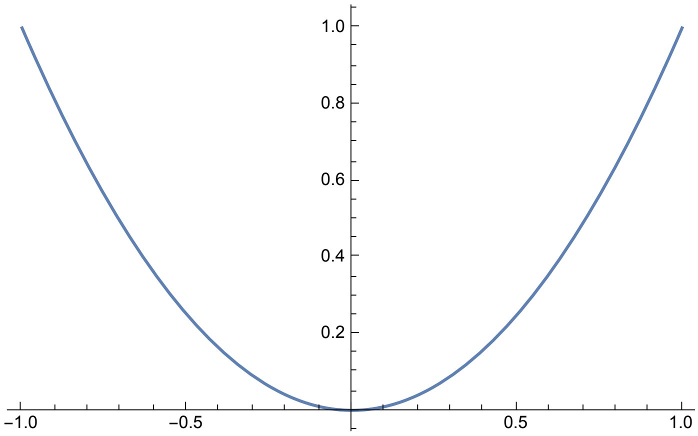
```
? Spacings
```

Spacings is an option to Grid and related constructs that specifies the spacings to leave between successive objects >>

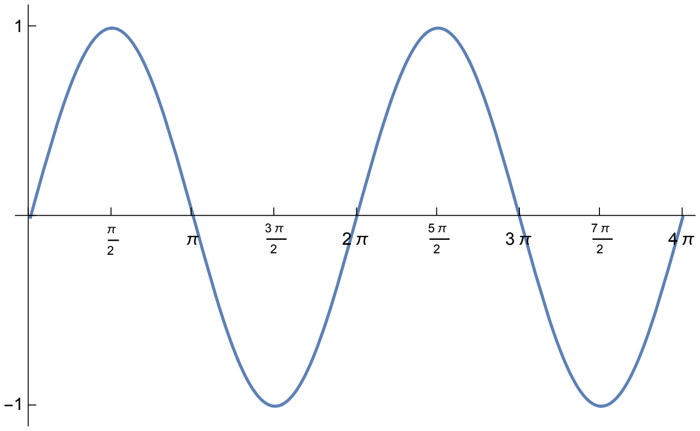
```
Spacings -> {Scaled[h], Scaled[v]}
```

Graphics not having the same size may be put into something resembling a graphics array in the following way:

```
graph1 = Plot[x^2, {x, -1, 1}]
```

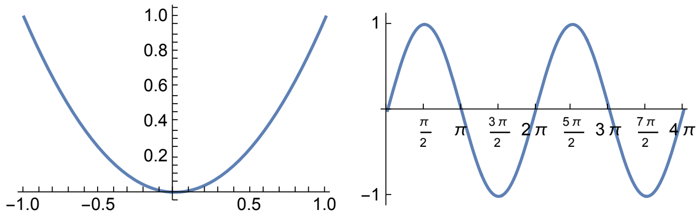


```
graph2 = Plot[Sin[x], {x, 0, 4 π}, Ticks → {π Range[0, 4, 1 / 2], {-1, 0, 1}}]
```

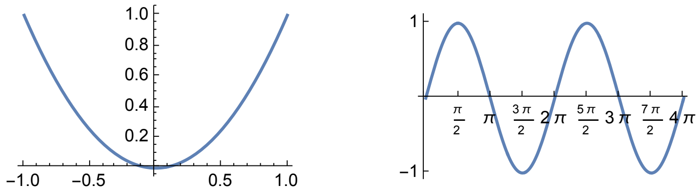


```
graph3 = Plot[Sin[x], {x, 0, 4 π},
   Ticks → {π Range[0, 4, 1 / 2], {-1, 0, 1}}, AspectRatio → 0.7];
```

```
Show[GraphicsRow[{graph1, graph2}, Spacings → {Scaled[0.1], Scaled[1]}]]
```



```
Show[GraphicsRow[{graph1, graph2}, Spacings → {Scaled[0.5], Scaled[1]}]]
```



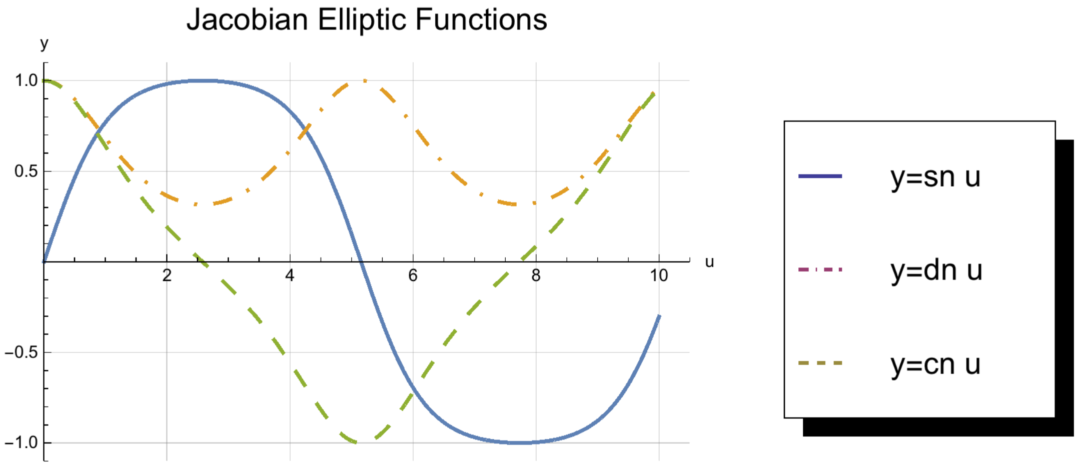
In order to achieve proper matching, it may be necessary to play with the number determining the **`AspectRatio`** and with **`Spacings`**.

6.7 Legends

Sometimes it is convenient to have specifications to a drawing in an own frame besides the drawing, called a legend. For this, some options in the **Plot[]** statement and the package **Graphics`Legend`** are needed. There is also a proper command **Legend[]** shown below.

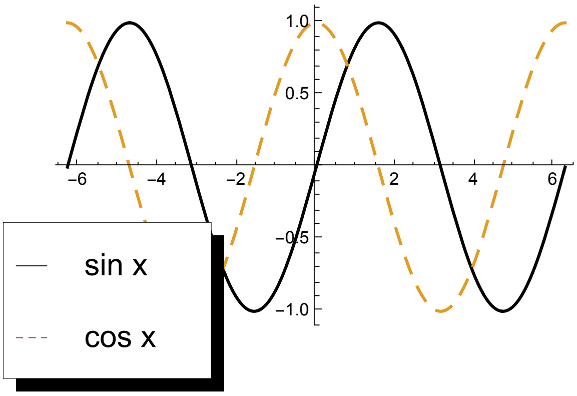
```
<< "PlotLegends`"

th = Thick;
Plot[{JacobiSN[t, 0.9`], JacobiDN[t, 0.9`], JacobiCN[t, 0.9`]},
 {t, 0, 10`}, PlotRange → {{0, 10.5`}, {-1.1`, 1.1`}},
 PlotPoints → 30, GridLines → Automatic, AxesLabel → {"u", "y"},
 PlotStyle → {{th, Dashing[{}]}, {th, Dashing[{0.03`, 0.025, 0.005, .025`}]},
   {th, Dashing[{0.03`}]}}, PlotLabel → Style["Jacobian Elliptic Functions", 16],
 LegendPosition → {1.1`, -0.5`}, ImageSize → 580,
 PlotLegend → {Style["y=sn u", 16], Style["y=dn u", 16], Style["y=cn u", 16]}]
```



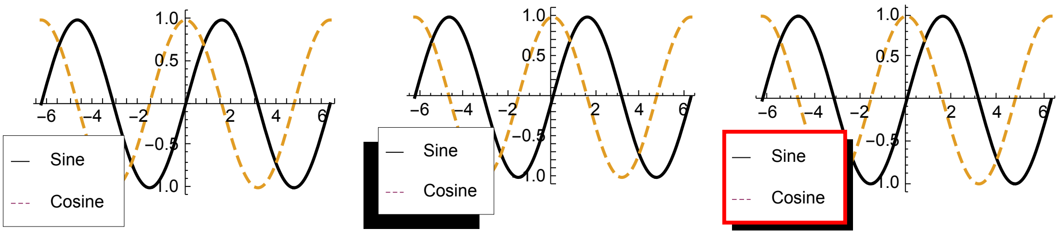
The content and the style of the legend may also be prescribed in the following way:

```
Plot[{Sin[x], Cos[x]}, {x, -2 π, 2 π},
 PlotStyle → {GrayLevel[0], Dashing[{0.03`}]}, PlotLegend →
  Map[(Style[#1, FontSize → 16] &), {"sin x", "cos x"}], ImageSize → 300]
```



The legend may be liberated from its shadow by the substitution shown in the Show[] below:

```
p1 = Plot[{Sin[x], Cos[x]}, {x, -2 π, 2 π},
    PlotStyle → {GrayLevel[0], Dashing[{0.03`}]},
    PlotLegend → {"Sine", "Cosine"}, LegendShadow → None];
p2 = Plot[{Sin[x], Cos[x]}, {x, -2 π, 2 π},
    PlotStyle → {GrayLevel[0], Dashing[{0.03`}]},
    PlotLegend → {"Sine", "Cosine"}, LegendShadow → {-0.1, -0.1}];
p3 = Plot[{Sin[x], Cos[x]}, {x, -2 π, 2 π},
    PlotStyle → {GrayLevel[0], Dashing[{0.03`}]},
    PlotLegend → {"Sine", "Cosine"}, LegendBorder → Directive[Thick, Red]];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 550]
```

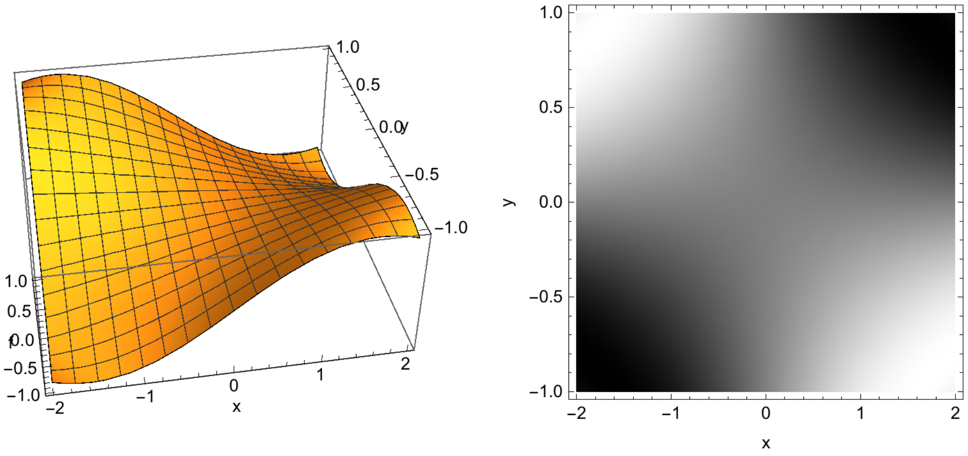


Option	Default value	Meaning
LegendPosition	**{-1, -1}**	**position of legend in relation to graph**
LegendSize	**Automatic**	**length of {*x*, *y*} dimensions**
LegendShadow	**Automatic**	**specify shadow**
LegendTextSpace	**Automatic**	**space in the legend box for text**
LegendTextDirection	**Automatic**	**direction text is rotated, as in Text graphic**
LegendTextOffset	**Automatic**	**offset of text, as in Text graphics primi**
LegendLabel	**None**	**label for legend**
LegendLabelSpace	**Automatic**	**specify space for LegendLabel**
LegendOrientation	**Vertical**	**direction in which key boxes are laid o**
LegendSpacing	**Automatic**	**specify the amount of space around each ke**
LegendBorder	**Automatic**	**style of border of key boxes and text**
LegendBorderSpace	**Automatic**	**specify space around all boxes and tex**
LegendBackground	**Automatic**	**style of the background**

6.7.1 Application of Legends

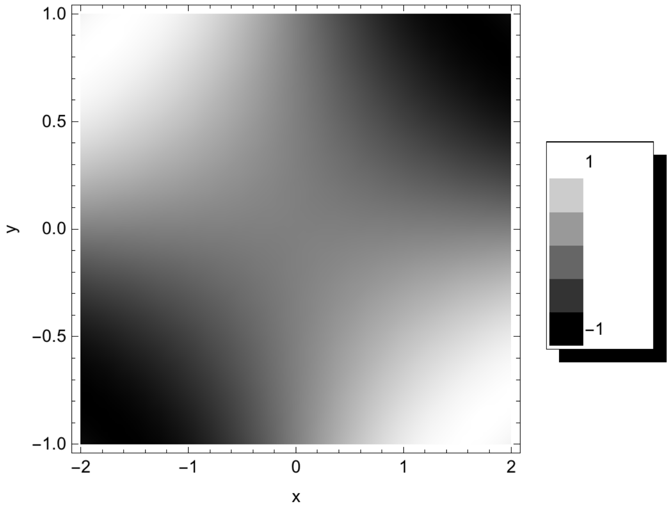
The Legend package may be used to create a shaded or colored density or contour plot accompanied by a legend,
in which the various gray-levels or colors have a numerical value specified.This is a very usual way of specifying
the``z-value" associated with a gray-levels or colors (a sort of height scale as used e.g. in maps). This is shown
below for gray-levels and colors.

```
p3 = Plot3D[-Sin[x y], {x, -2, 2}, {y, -1, 1},
   AxesLabel → {"x", "y", "f"}, ViewPoint → {-0.469`, -2.689`, 2.`}];
dp = DensityPlot[Sin[x y], {x, -2, 2}, {y, -1, 1}, ColorFunction →
     (GrayLevel[1 - #1] &), PlotPoints → 28, FrameLabel → {"x", "y"}];
Show[GraphicsRow[{p3, dp}], ImageSize → 500]
```

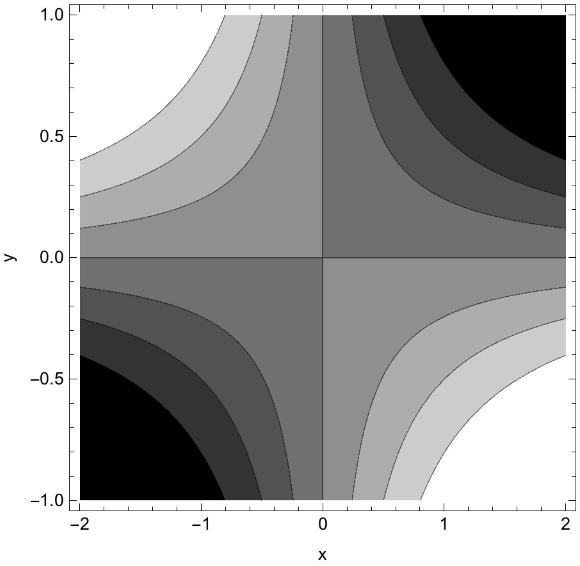


The maximum (maxf) and minimum (minf) values of the function displayed are needed to specify the values of the gray-levels in the legend. These may be found by inspecting the graphics generated by Plot3D[]. Looking for these extreme values by use of **FindMinimum[]** (or **ConstrainedMin[]**, **ConstrainedMax[]**, which are applicable to linear functions only!) may be onerous. Below the lows of the function are black, the heights white. Six levels of gray are used; more levels are hard to discern. Recall that the mesh may be omitted by the option **Mesh -> None.**

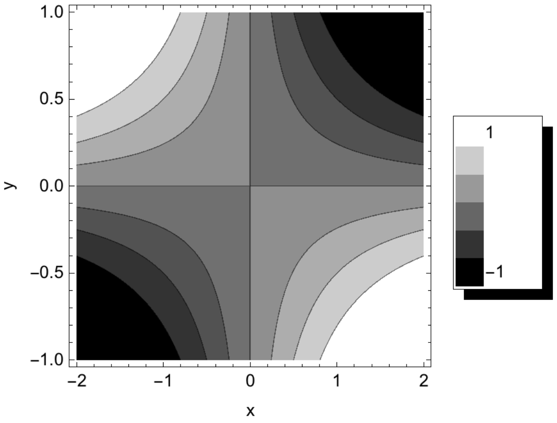
```
maxf = 1; minf = -1;
ShowLegend[dp, {GrayLevel[1 - #] &, 6, ToString[maxf],
  ToString[minf], LegendPosition → {1.1, -.4}}, FrameLabel -> {"x", "y"}]
```



```
c = ContourPlot[Sin[x y], {x, -2, 2}, {y, -1, 1},
  ColorFunction → (GrayLevel[1 - #1] &), PlotPoints → 28,
  Contours → 7, FrameLabel → {"x", "y"}, ImageSize → 300]
```



```
maxf = 1; minf = -1;
ShowLegend[c, {GrayLevel[1 - #] &, 6,
  ToString[maxf], ToString[minf], LegendPosition → {1.1, -.4}}]
```

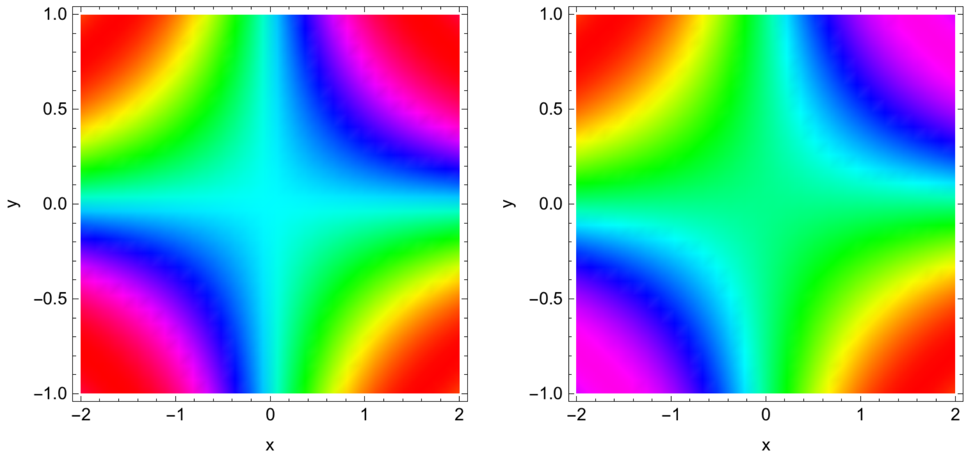


```
maxf = 1; minf = -1;
dd = DensityPlot[Sin[x y], {x, -2, 2}, {y, -1, 1},
   ColorFunction → (Hue[# ] &), PlotPoints → 28, FrameLabel -> {"x", "y"}];
```

```
dm = DensityPlot[Sin[x y], {x, -2, 2},
   {y, -1, 1}, ColorFunction → (Hue[#/ (maxf - minf - .82)] &),
   PlotPoints → 28, FrameLabel -> {"x", "y"}];
```

The color function **Hue[]** is red for **Hue[0]** and **Hue[1]**, so for the maximum and the minimum of the function displayed.
This gives an ambiguity to be seen in the left image below, where the maxima in the corners of theprincipal diagonal and the minima in the corners of the secondary diagonal both are colored in red. This ambiguity is removed by changing the argument of Hue[] as shown above in d and in the graphics below at the rhs. This function, in particular the value .82, have been found by trial and error.

```
Show[GraphicsRow[{dd, dm}], ImageSize → 500]
```

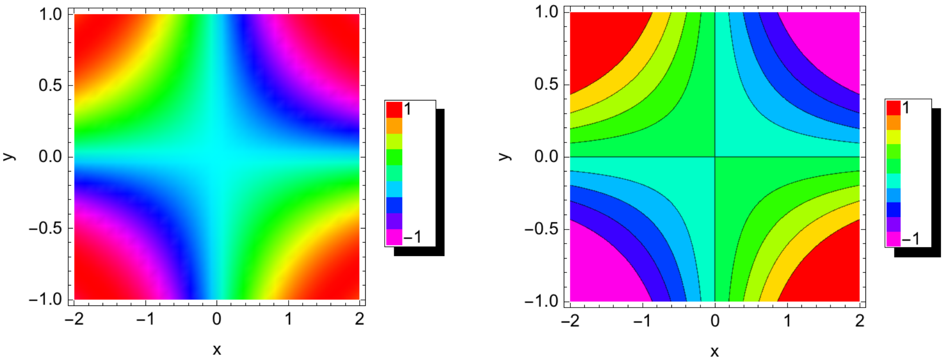


```
a1 = ShowLegend[dd, {Hue[#/ (maxf - minf - .82)] &, 9, ToString[maxf],
    ToString[minf], LegendPosition → {1.1, -.4}}, FrameLabel -> {"x", "y"}];
```

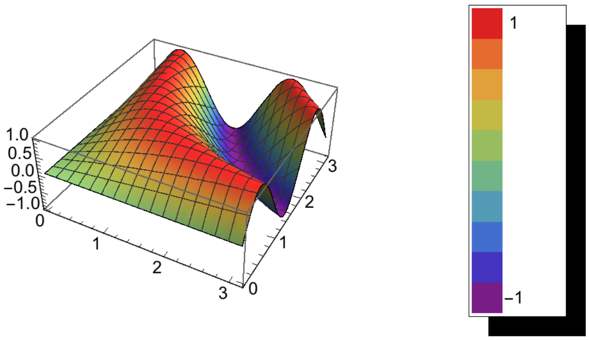
```
c = ContourPlot[Sin[x y], {x, -2, 2},
   {y, -1, 1}, ColorFunction → (Hue[#/(maxf - minf - .82)] &),
   PlotPoints → 28 , Contours → 9, FrameLabel -> {"x", "y"}];

a2 = ShowLegend[c, {Hue[#/(maxf - minf - .82)] &, 10,
    ToString[maxf], ToString[minf], LegendPosition → {1.1, -.4}},
   FrameLabel -> {"x", "y"}, RotateLabel → True];

Show[GraphicsRow[{a1, a2}], ImageSize → 500]
```



```
GraphicsRow[{Plot3D[Sin[x y], {x, 0, π}, {y, 0, π}, ColorFunction → "Rainbow"],
  Graphics[Legend[ColorData["Rainbow"][1 - #1] &, 10, " 1", "-1"]]}]
```



6.8 Animated Drawings

```
?? Animate
```

Animate[*expr*, {*u*, u_{min}, u_{max}}] generates an animation of *expr* in which *u* varies continuously from u_{min} to u_{max}.
Animate[*expr*, {*u*, u_{min}, u_{max}, *du*}] takes *u* to vary in steps *du*.
Animate[*expr*, {*u*, {u_1, u_2, …}}] makes *u* take on discrete values u_1, u_2, ….
Animate[*expr*, {*u*, …}, {*v*, …}, …] varies all the variables *u*, *v*, …. »

```
Attributes[Animate] = {HoldAll, Protected, ReadProtected}

Options[Animate] =
 {Alignment → Automatic, AnimationDirection → Forward, AnimationRate → Automatic,
  AnimationRepetitions → ∞, AnimationRunning → True, AnimationRunTime → 0,
  AnimationTimeIndex → Automatic, AppearanceElements → Automatic, AutoAction → False,
  AutorunSequencing → Automatic, BaselinePosition → Automatic, BaseStyle → {},
  Bookmarks → {}, ContentSize → Automatic, ContinuousAction → Automatic,
  ControlAlignment → Automatic, ControllerLinking → Automatic,
  ControllerMethod → Automatic, ControllerPath → Automatic,
  ControlPlacement → Automatic, ControlType → Automatic, DefaultBaseStyle → Animate,
  DefaultDuration → 5., DefaultLabelStyle → AnimateLabel, Deinitialization :→ None,
  Deployed → False, DisplayAllSteps → False, Evaluator → Automatic, Exclusions → {},
  Frame → False, FrameLabel → None, FrameMargins → Automatic, ImageMargins → 0,
  InterpolationOrder → Automatic, Initialization :→ None, LabelStyle → {},
  LocalizeVariables → True, Method → {}, Paneled → True, PausedTime → Automatic,
  PreserveImageOptions → True, RefreshRate → Automatic, RotateLabel → False,
  SaveDefinitions → False, ShrinkingDelay → Automatic, SynchronousInitialization → True,
  SynchronousUpdating → True, TouchscreenAutoZoom → False,
  TouchscreenControlPlacement → Automatic, TrackedSymbols → Full,
  UndoTrackedVariables :→ None, UnsavedVariables :→ None, UntrackedVariables :→ None}
```

6.8.1 Animations in Two Dimensions

Phase space diagram of the following system of differential equations

$x(t)^3 - x(t) + p'(t) + \epsilon\, p(t) = \gamma\, \cos(\omega\, t)$
$x'(t) = p(t)$

```
Clear[x, y, p, t, ω, ε, γ]
```

```
deqn = {-x[t] + x[t]^3 + Derivative[1][p][t] == γ Cos[ω t] - ε p[t],
     Derivative[1][x][t] == p[t]};
```

```
param  =  {ε -> 0.25,  γ  -> 0.3, ω -> 1.0};
initial   = {x[0] == 0,  p[0] == -0.8};
```

```
traj = NDSolve[Join[deqn,  initial] /. param,   {x[t],  p[t]}, {t,  0,  22 π}];
```

The animation below may not appear in the pdf-file.

```
?? ParametricPlot
```

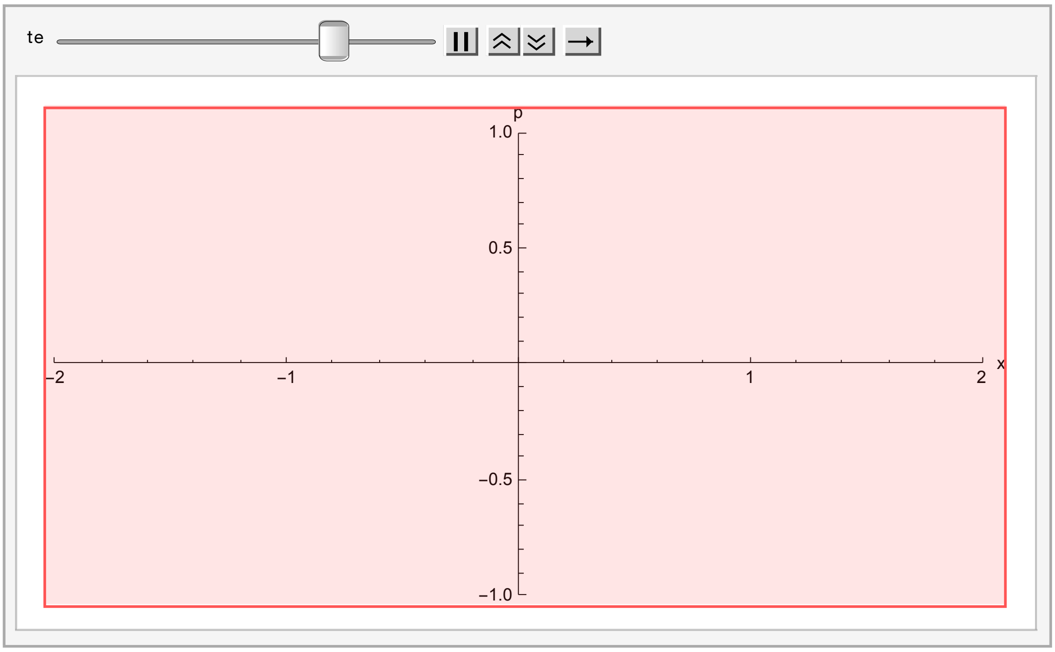
ParametricPlot[{f_x, f_y}, {u, u_{min}, u_{max}}] generates
  a parametric plot of a curve with x and y coordinates f_x and f_y as a function of u.
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, …}, {u, u_{min}, u_{max}}] plots several parametric curves
ParametricPlot[{f_x, f_y}, {u, u_{min}, u_{max}}, {v, v_{min}, v_{max}}] plots a parametric region
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, …}, {u, u_{min}, u_{max}}, {v, v_{min}, v_{max}}] plots several parametric regions
ParametricPlot[…, {u, v} ∈ reg] takes parameters {u, v} to be in the geometric region reg. >>

```
Attributes[ParametricPlot] = {HoldAll, Protected, ReadProtected}

Options[ParametricPlot] = {AlignmentPoint → Center, AspectRatio → Automatic, Axes → True,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, BoundaryStyle → Automatic,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
  EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
  FormatType :→ TraditionalForm, Frame → Automatic, FrameLabel → None,
  FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {}, GridLines → None,
  GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All, ImageSize → Automatic,
  ImageSizeRaw → Automatic, LabelStyle → {}, MaxRecursion → Automatic, Mesh → Automatic,
  MeshFunctions → Automatic, MeshShading → None, MeshStyle → Automatic,
  Method → Automatic, PerformanceGoal :→ $PerformanceGoal, PlotLabel → None,
  PlotLegends → None, PlotPoints → Automatic, PlotRange → Automatic,
  PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic,
  PlotStyle → Automatic, PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic,
  Prolog → {}, RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
  TextureCoordinateFunction → Automatic, TextureCoordinateScaling → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}

SyntaxInformation[ParametricPlot] =
 {ArgumentsPattern → {___, OptionsPattern[]}, LocalVariables → {Plot, {2, 3}},
  OptionNames → {AlignmentPoint, AspectRatio, Axes, AxesLabel, AxesOrigin, AxesStyle,
    Background, BaselinePosition, BaseStyle, BoundaryStyle, ColorFunction,
    ColorFunctionScaling, ColorOutput, ContentSelectable, CoordinatesToolOptions,
    DisplayFunction, Epilog, Evaluated, EvaluationMonitor, Exclusions, ExclusionsStyle,
    FormatType, Frame, FrameLabel, FrameStyle, FrameTicks, FrameTicksStyle,
    GridLines, GridLinesStyle, ImageMargins, ImagePadding, ImageSize, ImageSizeRaw,
    LabelStyle, LegendBackground, LegendBorder, LegendBorderSpace, LegendLabel,
    LegendLabelSpace, LegendOrientation, LegendPosition, LegendShadow, LegendSize,
    LegendSpacing, LegendTextDirection, LegendTextOffset, LegendTextSpace,
    MaxRecursion, Mesh, MeshFunctions, MeshShading, MeshStyle, Method, PerformanceGoal,
    PlotLabel, PlotLegend, PlotLegends, PlotPoints, PlotRange, PlotRangeClipping,
    PlotRangePadding, PlotRegion, PlotStyle, PlotTheme, PreserveImageOptions,
    Prolog, RegionFunction, RotateLabel, ShadowBackground, ShadowBorder,
    ShadowForeground, ShadowOffset, TargetUnits, TextureCoordinateFunction,
    TextureCoordinateScaling, Ticks, TicksStyle, WorkingPrecision}}
```

```
Animate[
ParametricPlot[{x[t], p[t]} /. traj, {t, 0, te},
 PlotRange -> {{-2, 2}, {-1, 1}}, PlotPoints -> 100, AxesLabel → {"x", "p"},
   FormatType → {FontSize → 18 }, ImageSize →  500],
{te, 0, 15 π}]
```



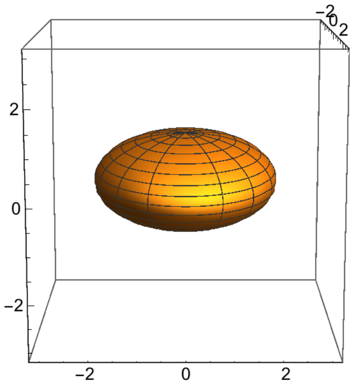
6.8.2 Animations in Three Dimensions

Rotation if a triaxial ellipsoid

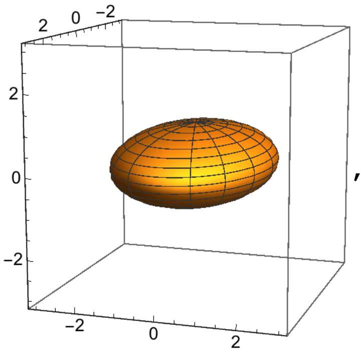
```
Clear[a, b, c, lr, th, ph]
a = 3; b = 2; c = 1;
lr = { a Sin[th] Cos[ph], b Sin[th] Sin[ph], c Cos[th]};
```

```
lieps = Table[ParametricPlot3D[lr, {th, 0, π}, {ph, 0, 2 π},
   PlotRange → 3.2 {{-1, 1}, {-1, 1}, {-1, 1}, SphericalRegion → True},
   ViewPoint  → 5 { Cos[α], Sin[α], 0.2}], {α, 0, π, π / 10}]
```

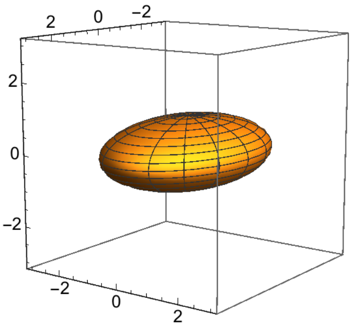
{



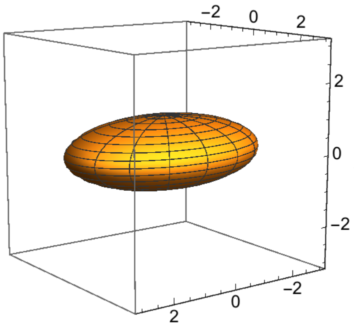
,



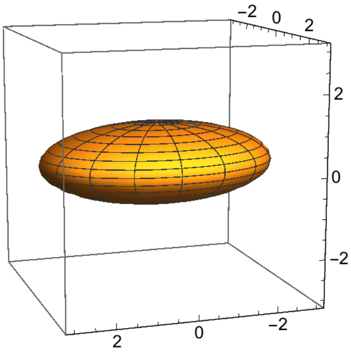
,



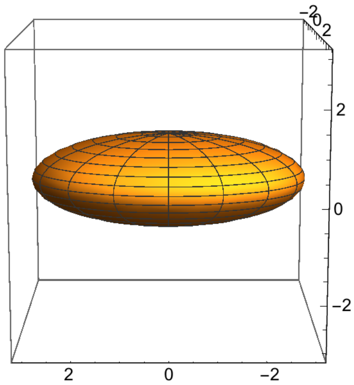
,



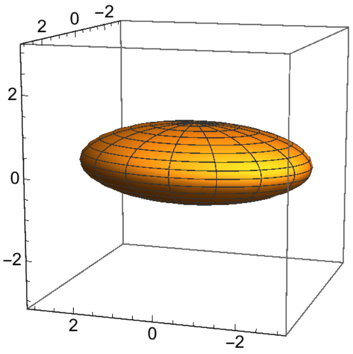
,



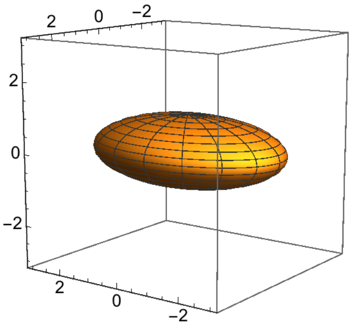
,



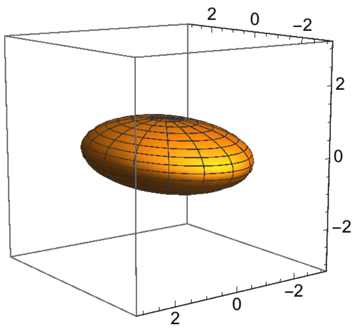
,



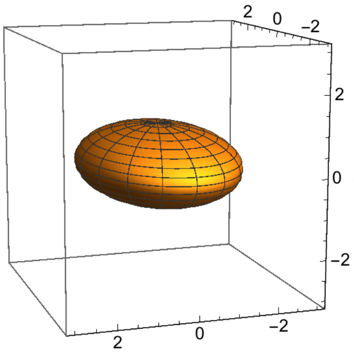
,



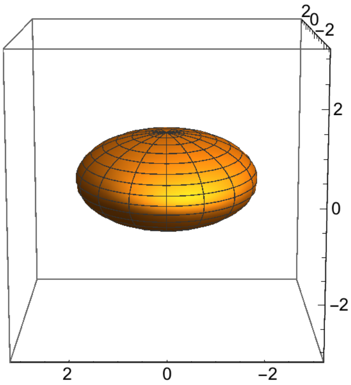
,



,



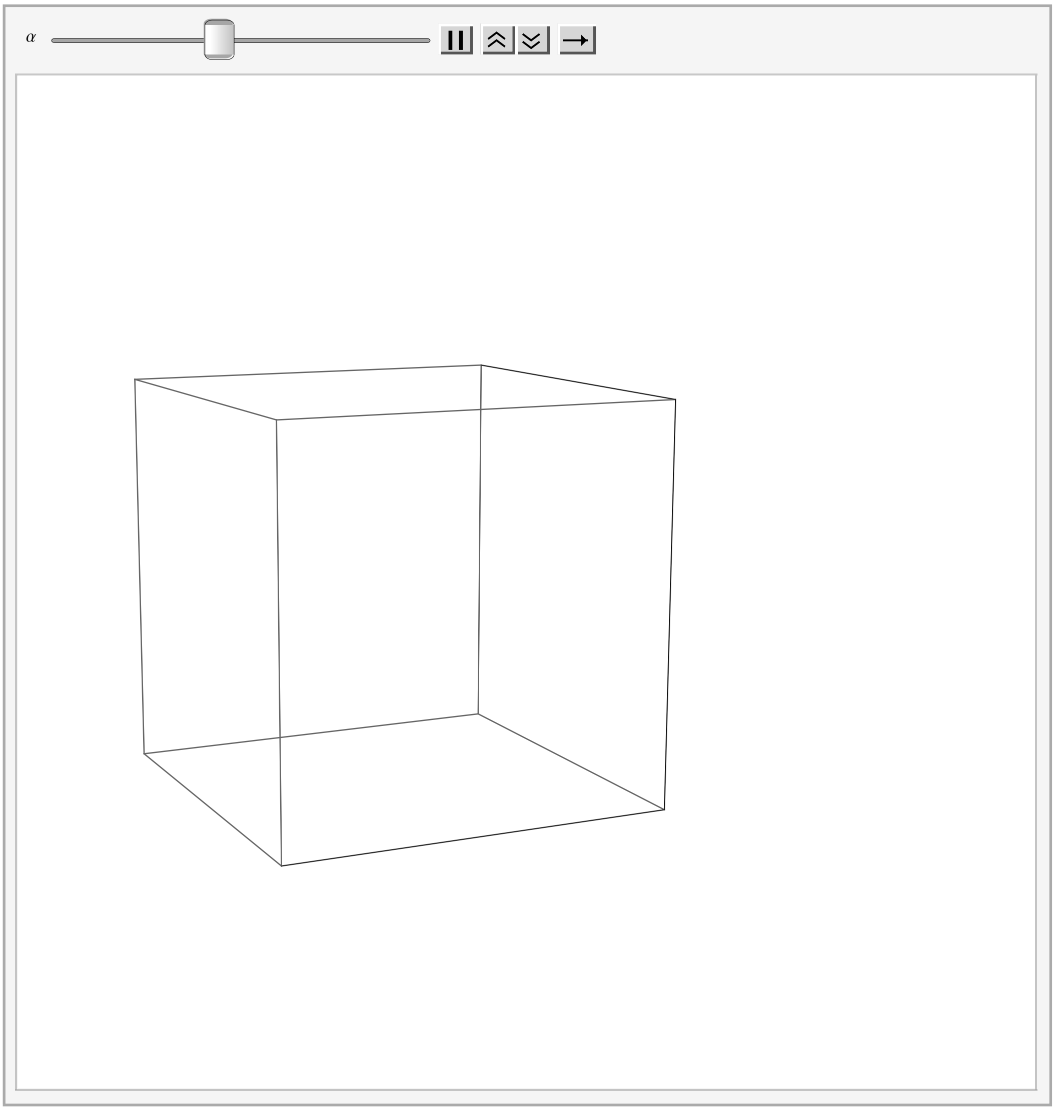
,



}

The animation below will not appear in the pdf-file.

```
Animate[ParametricPlot3D[lr, {th, 0, π}, {ph, 0, 2 π},
  PlotRange → 3.2 {{-1, 1}, {-1, 1}, {-1, 1}}, SphericalRegion → True,
  ViewPoint  → 5 { Cos[α], Sin[α], 0.2}, Ticks → None], {α, 0, 2 π}]
```



6.9 Exercises for Animations

6.31 Prepare an animation showing the phase space diagram of the following system of differential equations

(consult also Chap.11):

$x(t)^3 - x(t) + p'(t) + \epsilon\ (p(t)) = \gamma\ \cos(\omega\, t)$,

$x'(t) = p(t)$; $t = 0: \ x = 0.3, \ y = -0.6$;

$\varepsilon = 0.25, \ \gamma = 0.3, \ \omega = 1.1$; $0 \le t \le 15\,\pi$.

What are the values of x and p at t = 15π?

6.32 Prepare an animation showing a triaxial ellipsoid, which is no longer in the principal but in some oblique

position, rotating around the vertical axis. For this, one may multiply the list lr above by an orthogonal

matrix as, for example, by

$$\begin{pmatrix} \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & \frac{1}{2} & 0 \\ \frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{pmatrix}.$$

6.10 Sound

Mathematica provides a lot of commands for producing and treating soud signals. You can get information on this by choosing in the help menu “Virtual book” and then inserting “Sound”
in the window frame of the appearing menue.
Here only a few commands are given as examples. At the end we show how one can insert a beep as a warning signal in a *Mathematica* programme.

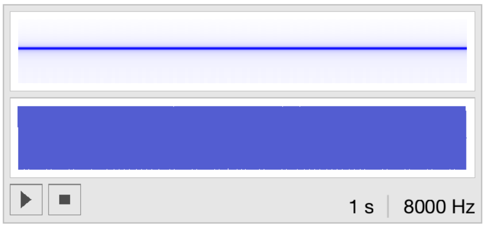
10.1 Some simple Sound commands

For example, just as you can use `Plot[`f`, {`x`, `x_{min}`, `x_{max}`}]` to plot a function, so also you can use **Play**`[`f`, {`t`, 0, `t_{max}`}]` to “play” a function. `Play` takes the function to define the waveform for a sound: the values of the function give the amplitude of the sound as a function of time.

```
? Play
```

Play[f, {t, t_{min}, t_{max}}] creates an object that plays as a sound
whose amplitude is given by f as a function of time t in seconds between t_{min} and t_{max}. »

```
snd = Play[Sin[2 Pi 440 t], {t, 0, 1}]
```



Press the cursor on the button with a triangle (a square) to start (stop) the signal.

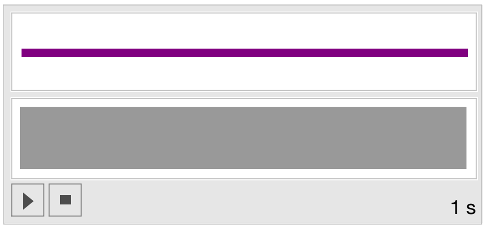
```
? EmitSound
```

EmitSound[snd] emits the sound snd when evaluated
EmitSound[{snd_1, snd_2, …}] emits each of the sounds snd_i in sequence »

```
EmitSound[snd]
```

Produce a middle C :

```
Sound[SoundNote["C"]]
```

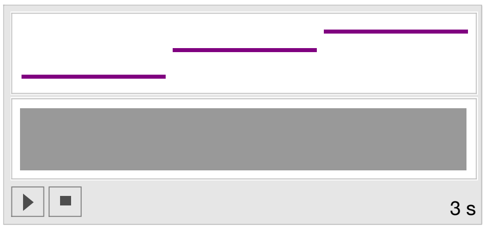


One can generate the sound of various musical instruments:

```
Sound[SoundNote[{"C", "G"}, 1, "Harpsichord"]]
```

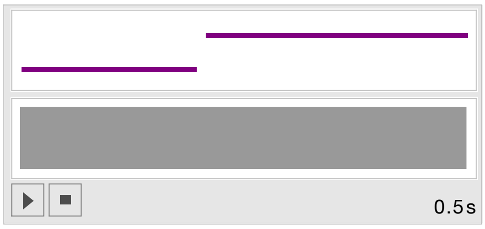
Produce a sequence of three notes :

```
Sound[{SoundNote["C"], SoundNote["G"], SoundNote["C5"]}]
```



C for 0.2 seconds, G for 0.3 seconds:

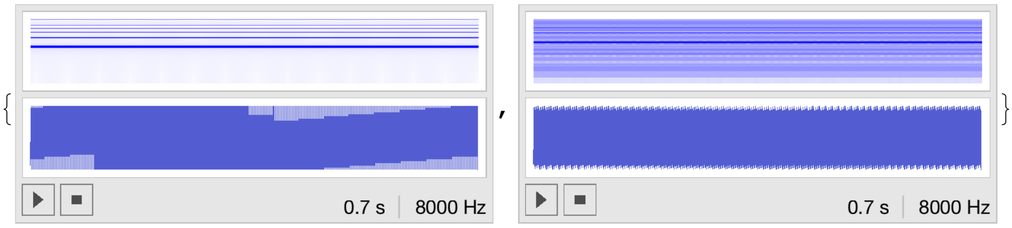
```
Sound[{SoundNote["C", 0.2], SoundNote["G", 0.3]}]
```



A beep as a warning signal

This is the sound signal:

```
sou = {Play[FractionalPart[500 t], {t, 0, 0.7}],
   Play[FractionalPart[700 t], {t, 0, 0.7}]}
```



This is used in a check whether an even function expression contains sines or not.

Expression does not contain a sine:

```
expr1 = Sin[x]^8 // TrigReduce // Expand
```

$$\frac{35}{128}-\frac{7}{16}\text{Cos}[2\,x]+\frac{7}{32}\text{Cos}[4\,x]-\frac{1}{16}\text{Cos}[6\,x]+\frac{1}{128}\text{Cos}[8\,x]$$

```
test = Position[expr1, Sin]
```

```
{}
```

```
If[test != {}, {EmitSound[sou], Print["Error"]}]
```

Expr1 does not contain a sine; therefor no signal !

Expression does contain a sine:

```
expr2 = Sin[x]^8 // TrigReduce // Simplify
```

$\text{Sin}[x]^8$

```
test = Position[expr2, Sin]
```

```
{{1, 0}}
```

```
If[test != {}, {EmitSound[sou], Print["Error"]}];
```

```
Error
```