13. Elementary Mathematical Functions

2016-07-08

```
$Version
```

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

13.1 Mathematical Constants (A Selection)

E	**N[E]**	2.718281828459045
I		Sqrt[-1]
π	**N[Pi]**	3.14159....
Infinity	∞	
Degree = π/180 factor	**N[Degree]**	0.0174532925199433 degrees to radians conversion
EulerGamma (= γ)	**N[EulerGamma]**	0.5772156649015329

Mathematica can calulate an arbitrary number of decimal places of these constants. How this is done for π is described in subsection 7.1.3. *f*

13.2 Numerical Functions

N[*expr*]	gives the numerical value of ***expr***
expr//N	postfix form of above command
N[*expr,n*]	attempts to give a result with n-digit precision
SetPrecision[*expr,n*]	yields a version of ***expr*** in which a number has been set to have a precision of ***n*** digits. (See examples below !)
Chop[*expr*]	replaces approximate real small numbers by **0**
Chop[*expr,delta*]	replaces numbers smaller in absolute magnitude than ***delta*** by **0**

```
nus = 1 + 1 / 10
```

$$\frac{11}{10}$$

```
N[11 / 10]
```

```
1.1
```

```
N[%, 50]
```

```
1.1
```

```
SetPrecision[%, 50]
```

```
1.10000000000000008881784197001252323389053344726 56
```

```
SetPrecision[nus, 50]
```

```
1.10000000000000000000000000000000000000000000000000
```

Note the difference of the last two results ! The very last input is an exact number, which is really rendered with
50-digit precision. In the last but one evaluation the input is a number with 17 digits precision. SetPrecision adds

Chop[] uses a default tolerance of about 10^-10 (depends on machine and version). It works on **Real** and **Complex** numbers. It is very useful to chop spurious small contributions to otherwise large numbers introduced by the inaccuracy of numerical calculations.

Input	Output
`Chop[1.0000000000000001]`	1.000000000000000
`N[π]`	3.14159

```
SetPrecision[π, 55]
```

```
3.1415926535897932384626433832795028841971693993751058 21
```

Besselfunction of 1-st kind of order 5.3 with argument 3.4445.

`BesselJ[5.3, 3.4445]`	0.0542925

```
SetPrecision[BesselJ[2.5, 3.4445], 55]
```

```
0.4531989890765290018848077124857809394598007202148437500
```

```
N[BesselJ[5 / 2, 3.4445237609867234123465982342365], 55]
```

```
0.4532000387041516172707361 78857
```

```
fx = BesselJ[5 / 2, x] // Simplify // Together
```

$$-\frac{\sqrt{\frac{2}{\pi}}\ \left(3\,x\,\text{Cos}[x]-3\,\text{Sin}[x]+x^2\,\text{Sin}[x]\right)}{x^{5/2}}$$

```
N[fx /. x -> 3, 55]
```

```
0.4127100322097159934374967959418627149872611162453162254
```

```
N[fx /. x -> 3.4445237609867234123465982342365, 55]
```

```
0.4532000387041516172707361 78857
```

Here it is shown how **Chop[]** is used to remove spurious small numbers entering numerical calculations.

```
p = x^5 + 17. x + 23.
```

$$23.+17.\,x+x^5$$

```
so = NSolve[ p == 0]
```

```
{{x → -1.20407}, {x → -1.0907 - 1.60808 i},
 {x → -1.0907 + 1.60808 i}, {x → 1.69274 - 1.4812 i}, {x → 1.69274 + 1.4812 i}}
```

```
p /. so
```

$$\{0.,\ -1.06581\times10^{-14}+0.\,i,\ -1.06581\times10^{-14}+0.\,i,\ 1.42109\times10^{-14}+5.68434\times10^{-14}\,i,\ 1.42109\times10^{-14}-5.68434\times10^{-14}\,i\}$$

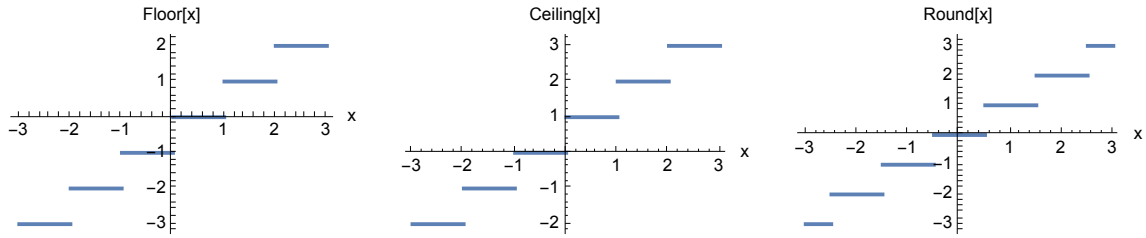
```
Chop[p /. so]
```

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

Round[x]	integer closest to **x**
Floor[x]	greatest integer not larger than **x** , Gauss' bracket **[x]**
Ceiling[x]	least integer not smaller than **x**

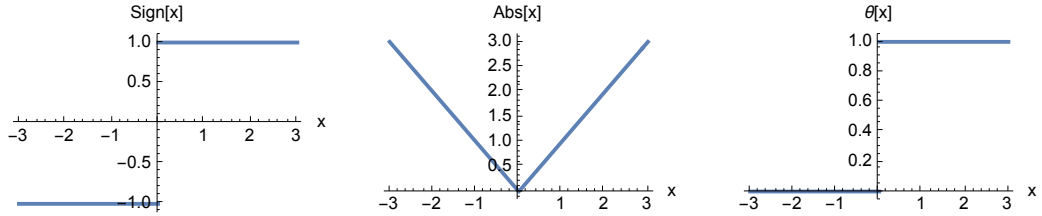
Input	Output
Round[π]	3
Round[π]	3
Floor[π]	3
Ceiling[π]	4
Round[-π]	-3
Floor[-π]	-4
Ceiling[-π]	-3
Round[.5]	0
Round[.50000000001]	1

```
SetOptions[Plot, PlotStyle → Thick];
p1 = Plot[Floor[x], {x, -3, 3}, AxesLabel → {"x", "Floor[x]"}];
p2 = Plot[Ceiling[x], {x, -3, 3}, AxesLabel → {"x", "Ceiling[x]"}];
p3 = Plot[Round[x], {x, -3, 3}, AxesLabel → {"x", "Round[x]"}];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 600]
```



Sign[x]	=	**+1**	for	**x > 0**
		0	for	**x = 0**
		-1	for	**x < 0**
Abs[x]	absolute value	**|x|**		
UnitStep[x]	= **θ(x)**	= **1**	for	**x ≥ 0,** Heaviside step function ;
		= **0**	for	**x < 0**

```
p1 = Plot[Sign[x], {x, -3, 3}, AxesLabel → {"x", "Sign[x]"}, PlotStyle → Thick];
p2 = Plot[Abs[x], {x, -3, 3}, AxesLabel → {"x", "Abs[x]"}, PlotStyle → Thick];
p3 = Plot[UnitStep[x], {x, -3, 3}, AxesLabel → {"x", "θ[x]"}, PlotStyle → Thick];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 550]
```



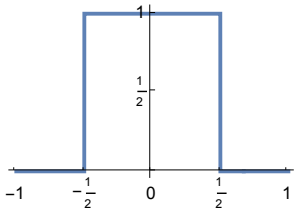
UnitStep[0]	**1**
UnitStep[-0.999]	**0**
Limit[UnitStep[x], x -> 0, Direction -> 1]	**0**

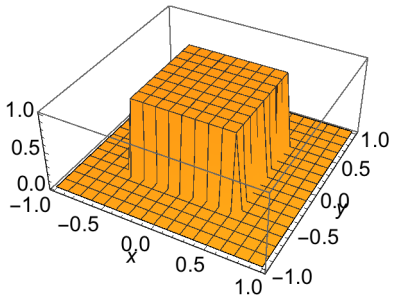
```
? UnitBox
```

UnitBox[x] represents the unitbox function, equal to 1 for $|x| \le \frac{1}{2}$ and 0 otherwise

UnitBox[x_1, x_2, ...] represents the multidimensional unitbox function, equal to 1 if $|x_i| \le \frac{1}{2}$ and 0 otherwise ≫

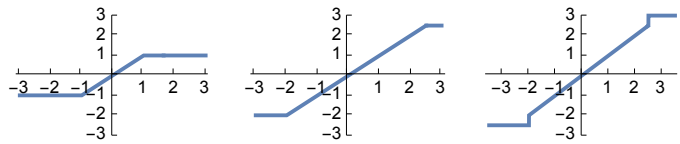
```
pu1 = Plot[UnitBox[x], {x, -1, 1},
  Ticks → {Range[-2, 2, 1 / 2], {0, 1 / 2, 1}}, ImageSize → 150]
pu2 = Plot3D[UnitBox[x, y], {x, -1, 1}, {y, -1, 1},
  AxesLabel → {("\ x"), "\ y"}, BaseStyle → FontSize → 11, ImageSize → 200]
```





`Clip[x]`	gives x clipped to be between −1 and +1.
`Clip[x,{min,max}]`	gives x for $min \le x \le max$, min for $x < min$ and max for $x > max$.
`Clip[x,{min,max},{`v_{min}, v_{max}`}]`	gives v_{min} for $x < min$ and v_{max} for $x > max$.

```
pl1 = Plot[Clip[x], {x, -3, 3},
    PlotRange → 3 {-1, 1}, Ticks → {Range[-3, 3], Range[-3, 3]}];
pl2 = Plot[Clip[x, {-2, 2.5}], {x, -3, 3}, PlotRange → 3 {-1, 1},
    Ticks → {Range[-3, 3], Range[-3, 3]}];
pl3 = Plot[Clip[x, {-2, 2.5}, {-2.5, 3}], {x, -3.5, 3.5},
    PlotRange → 3 {-1, 1}, Ticks → {Range[-3, 3], Range[-3, 3]}];
GraphicsRow[{pl1, pl2, pl3}]
```



`Max[x1, x2, ... ]`	the maximum of `x1, x2, ...`
`Max[{...}, {...} ]`	the maximum of all values contained in all the lists
`Min[x1, x2, ... ]`	the minimum of `x1, x2, ...`
`Min[{...}, {...} ]`	the minimum of all values contained in all the lists

```
Max[{-1,2}, {N[π], N[E]}]
```
3.14159

```
Min[{-1,2},{N[π], N[E]}]
```
−1

```
?? FindMinimum
```

FindMinimum[f, x] searches for a local minimum in f, starting from an automatically selected point.
FindMinimum[f, {x, x0}] searches for a local minimum in f, starting from the point x = x0.
FindMinimum[f, {{x, x0}, {y, y0}, ...}] searches for a local minimum in a function of several variables.
FindMinimum[{f, cons}, {{x, x0}, {y, y0}, ...}] searches for a local minimum subject to the constraints cons.
FindMinimum[{f, cons}, {x, y, ...}] starts from a point within the region defined by the constraints. >>

```
Attributes[FindMinimum] = {HoldAll, Protected}
```

```
Options[FindMinimum] =
 {AccuracyGoal → Automatic, Compiled → Automatic, EvaluationMonitor → None,
  Gradient → Automatic, MaxIterations → Automatic, Method → Automatic,
  PrecisionGoal → Automatic, StepMonitor → None, WorkingPrecision → MachinePrecision}
```

```
f = Sin[x]
```

```
Sin[x]
```

```
FindMinimum[f, x]
```

```
{-1., {x → -1.5708}}
```

```
FindMinimum[f, {x, 3}]
```

```
{-1., {x → 4.71239}}
```

```
FindMinimum[Abs[f], {x, 3}]
```

FindMinimum::lstol:
The line search decreased the step size to within the tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances. >>

```
{1.66627 × 10^-8, {x → 3.14159}}
```

FindMinimum::lstol:
The line search decreased the step size to within the tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances. >>

```
FindMinimum[{x Cos[x], 1 ≤ x ≤ 15}, {x, 7}]
```

```
{-9.47729, {x → 9.52933}}
```

```
FindMinimum[{x Cos[x], 15 ≤ x ≤ 30}, {x, 20}]
```

```
{-22.0138, {x → 22.0365}}
```

```
FindMinimum[Sin[x] + Cos[y], {x, 2}, {y, 4}]
```

FindMinimum::lstol:
The line search decreased the step size to within the tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances. >>

```
{-2., {x → 4.71239, y → 3.14159}}
```

FindMinimum::lstol:
The line search decreased the step size to within the tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the function. You may need more than MachinePrecision digits of working precision to meet these tolerances. >>

```
f = Sin[x y]
```

```
Sin[x y]
```

```
FindMinimum[f, {x, π / 2}, {y, 0}]
```

```
{-1., {x → 1.67007, y → -0.940555}}
```

```
FindMinimum[f, {x, π / 2}, {y, π}]
```

```
{-1., {x → 1.51383, y → 3.11289}}
```

```
FindMinimum[f, {x, 2.4, 2, 3}, {y, 2.4, 2, 3}]
```

```
{-1., {x → 2.1708, y → 2.1708}}
```

```
FindMinimum[Abs[f], {x, 2.4, 2, 3}, {y, 2.4, 2, 3}]
```

FindMinimum::regex: The point {3., 3.} is at the edge of the search region {2., 3.} in coordinate 1 and the computed search direction points outside the region. >>

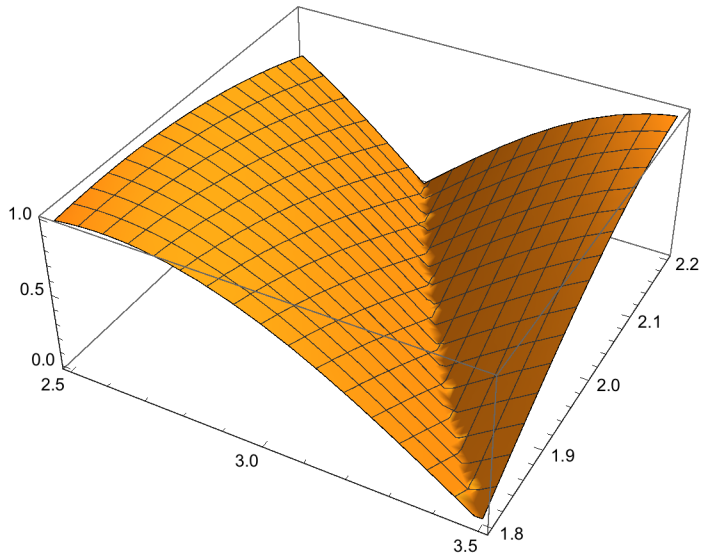
```
{0.412118, {x → 3., y → 3.}}
```

FindMinimum::regex: The point {3., 3.} is at the edge of the search region {2., 3.} in coordinate 1 and the computed search direction points outside the region. >>

```
FindMinimum[Abs[f], {x, 2.4, 3}, {y, 1.8, 2.2}]
```

$$\left\{1.22465 \times 10^{-16}, \{x \to 1.74533, y \to 1.8\}\right\}$$

```
Plot3D[Abs[f], {x, 2.5`, 3.5`}, {y, 1.8`, 2.2`}]
```




```
?? FindMaximum
```

FindMaximum[f, x] searches for a local maximum in f, starting from an automatically selected point.
FindMaximum[f, {x, x0}] searches for a local maximum in f, starting from the point x = x0.
FindMaximum[f, {{x, x0}, {y, y0}, ...}] searches for a local maximum in a function of several variables.
FindMaximum[{f, cons}, {{x, x0}, {y, y0}, ...}] searches for a local maximum subject to the constraints cons.
FindMaximum[{f, cons}, {x, y, ...}] starts from a point within the region defined by the constraints. >>

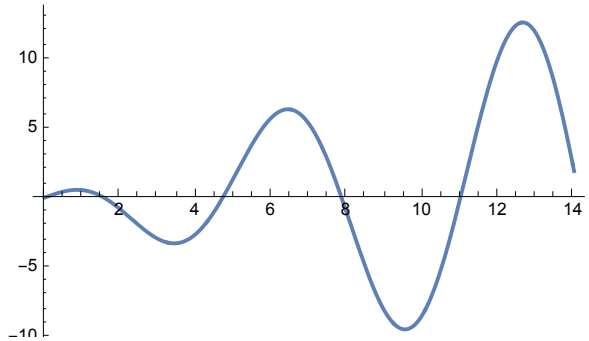
```
Attributes[FindMaximum] = {HoldAll, Protected}
```

```
Options[FindMaximum] =
 {AccuracyGoal → Automatic, Compiled → Automatic, EvaluationMonitor → None,
  Gradient → Automatic, MaxIterations → Automatic, Method → Automatic,
  PrecisionGoal → Automatic, StepMonitor → None, WorkingPrecision → MachinePrecision}
```

```
FindMaximum[x Cos[x], {x, 2}]
```

```
{0.561096, {x → 0.860334}}
```

```
Plot[x Cos[x], {x, 0, 14}, ImageSize → 300]
```



```
FindMaximum[{x Cos[x], 1 ≤ x ≤ 15}, {x, 7}]
```

```
{6.361, {x → 6.4373}}
```

```
FindMaximum[{x Cos[x], 1 ≤ x ≤ 15}, {x, 12}]
```

```
{12.6059, {x → 12.6453}}
```

Find the maximum of a linear function, subject to linear and integer constraints:

```
FindMaximum[{-x - y, x + 2 y ≥ 3 && x ≥ 0 && y ≥ 0 && y ∈ Integers}, {x, y}]
```

```
{-2., {x → 0., y → 2}}
```

Local maximum constrained within a disk:

```
FindMaximum[{Sin[x] Sin[2 y], x^2 + y^2 < 3}, {{x, 2}, {y, 2}}]
```

```
{0.999656, {x → 1.54538, y → 0.782181}}
```

`Or` constraints can be specified:

```
FindMaximum[{x + y, x^2 + y^2 ≤ 1 || (x + 2)^2 + (y + 2)^2 ≤ 1}, {x, y}]
```

```
{1.41421, {x → 0.707106, y → 0.707106}}
```

13.3 Complex Numbers and Complex Expressions

z = x + I y	the complex number **z**
Re[z]	the real part of **z**
Im[z]	the imaginary part of **z**
Conjugate[z]	the complex conjugate **z*** of **z**
Abs[z]	the absolute value **|z|** of **z**
Arg[z]	the argument ϕ such that **z** = **|z|** $e^{i\phi}$, with $-\pi < \phi \leq \pi$

In general, these functions do not work on every symbolic input ! In most cases one can get results on expressions by using **ComplexExpand[]**, s. §4.6.1.

```
Abs[x + I y]
```

```
Abs[x + i y]
```

```
z = 3 + I 4
```

```
3 + 4 i
```

```
Abs[z]
```

```
5
```

```
Re[z]
```

```
3
```

```
Im[z]
```

```
4
```

```
Conjugate[z]
```

```
3 - 4 i
```

```
Arg[z]
```

$$\text{ArcTan}\left[\frac{4}{3}\right]$$

```
N[%]
```

```
0.927295
```

```
N[{Arg[ -5 + 0.1 I], Arg[ -5 -  0.1 I] }]
```

```
{3.1216, -3.1216}
```

```
z = Exp[I π]
```

```
-1
```

```
Arg[z]
```

π

```
Limit[Arg[z + I ϵ], ϵ → 0, Direction → +1]
```

$-\pi$

```
Limit[Arg[z + I ϵ], ϵ → 0, Direction → -1]
```

π

```
z = Exp[-I π]
```

```
-1
```

```
Arg[z]
```

π

```
Limit[Arg[z + I ϵ], ϵ → 0, Direction → +1]
```

$-\pi$

```
Limit[Arg[z + I ϵ], ϵ → 0, Direction → -1]
```

π

Note, that both, $e^{i\pi}$ and $e^{-i\pi}$, inserted into the operator **Arg[]** give the **same** value -1. But the directional limits give different values !

13.3.1 Functions of Complex Expressions

```
z = x + I y
```

```
x + i y
```

```
Re[z]
```

```
-Im[y] + Re[x]
```

```
Im[z]
```

```
Im[x] + Re[y]
```

```
ComplexExpand[Re[z]]
```

```
x
```

```
ComplexExpand[Im[z]]
```

```
y
```

```
Abs[z]
```

```
Abs[x + i y]
```

```
ComplexExpand[Abs[z]]
```

$$\sqrt{x^2 + y^2}$$

```
ComplexExpand[Abs[z],TargetFunctions → {Re,Im}]
```

$$\sqrt{x^2 + y^2}$$

```
ComplexExpand[Abs[Exp[I v]]]
```

```
1
```

```
ComplexExpand[ Re[ Sin[z] ]]
```

```
Cosh[y] Sin[x]
```

```
ComplexExpand[ Im[ Tan[z] ]]
```

$$\frac{\text{Sinh}[2\,y]}{\text{Cos}[2\,x] + \text{Cosh}[2\,y]}$$

```
E^(I z)
```

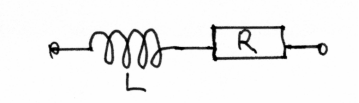
$$e^{i\,(x+i\,y)}$$

```
ComplexExpand[%]
```

$$e^{-y}\,\text{Cos}[x] + i\,e^{-y}\,\text{Sin}[x]$$

13.3.2 The Use of Complex Quantities for Describing Alternating Currents

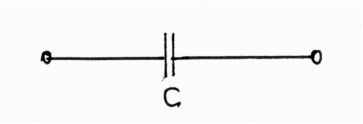
Impedance of a coil:




```
Z1 = I ω L + R
```

$$R + i\,L\,\omega$$

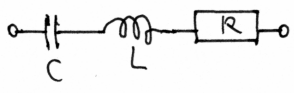
Impedance of a condensor:




```
Z2 = (I ω CC)^-1
```

$$-\frac{i}{CC\,\omega}$$

Series impedance of circuit comprising coil and condensor:




```
Z = Z1 + Z2
```

$$R - \frac{i}{CC\,\omega} + i\,L\,\omega$$

Frequency of series resonance:

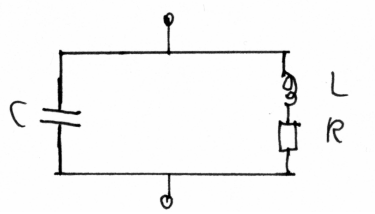
```
omr = Solve[ComplexExpand[Im[Z]] == 0, ω]
```

$$\left\{\left\{\omega \to -\frac{1}{\sqrt{CC}\ \sqrt{L}}\right\}, \left\{\omega \to \frac{1}{\sqrt{CC}\ \sqrt{L}}\right\}\right\}$$

```
Z /. omr[[2]]
```

$$R$$

Parallel resonance of a circuit comprising a coil and a condensor:




```
Y = Z1^-1 + Z2^-1
```

$$i\ CC\ \omega + \frac{1}{R + i\ L\ \omega}$$

```
Z = 1/Y
```

$$\frac{1}{i\ CC\ \omega + \frac{1}{R+i\ L\ \omega}}$$

```
Z = ComplexExpand[Z, TargetFunctions → {Re,Im}]
```

$$\frac{R}{\left(R^2 + L^2\ \omega^2\right)\ \left(\frac{R^2}{(R^2+L^2\ \omega^2)^2} + \left(CC\ \omega - \frac{L\ \omega}{R^2+L^2\ \omega^2}\right)^2\right)} +$$

$$i\left(-\frac{CC\ \omega}{\frac{R^2}{(R^2+L^2\ \omega^2)^2} + \left(CC\ \omega - \frac{L\ \omega}{R^2+L^2\ \omega^2}\right)^2} + \frac{L\ \omega}{\left(R^2 + L^2\ \omega^2\right)\ \left(\frac{R^2}{(R^2+L^2\ \omega^2)^2} + \left(CC\ \omega - \frac{L\ \omega}{R^2+L^2\ \omega^2}\right)^2\right)}\right)$$

```
Y = 1 / Z  // FullSimplify
```

$$i\left(CC\ \omega + \frac{1}{i\ R - L\ \omega}\right)$$

```
Y = ComplexExpand[Y, TargetFunctions → {Re, Im}]
```

$$\frac{R}{R^2 + L^2\ \omega^2} + i\left(CC\ \omega - \frac{L\ \omega}{R^2 + L^2\ \omega^2}\right)$$

Frequency of parallel resonance:

```
omr = Solve[ComplexExpand[Im[Y]] == 0, ω]
```

$$\left\{\{\omega \to 0\}, \left\{\omega \to -\frac{\sqrt{L - CC\ R^2}}{\sqrt{CC}\ L}\right\}, \left\{\omega \to \frac{\sqrt{L - CC\ R^2}}{\sqrt{CC}\ L}\right\}\right\}$$

```
Yr = Y /. omr[[3]]
```

$$\frac{R}{R^2 + \frac{L-CC\ R^2}{CC}} + i\left(\frac{\sqrt{CC}\ \sqrt{L - CC\ R^2}}{L} - \frac{\sqrt{L - CC\ R^2}}{\sqrt{CC}\ \left(R^2 + \frac{L-CC\ R^2}{CC}\right)}\right)$$

```
Yr = ExpandAll[Yr]
```

$$\frac{CC\ R}{L}$$

The resistance of the parallel circuit at resonance is:

```
Zr = 1/Yr
```

$$\frac{L}{CC\,R}$$

13.4 Pseudorandom Numbers

```
?? Random
```

Random[] gives a uniformly distributed pseudorandom Real in the range 0 to 1.
Random[*type*, *range*] gives a pseudorandom number of the specified type, lying in the specified range. Possible types are: Integer, Real and Complex. The default range is 0 to 1. You can give the range {*min*, *max*} explicitly; a range specification of *max* is equivalent to {0, *max*}. >>

```
Attributes[Random] = {Protected}
```

```
Random[]
```

```
0.123287
```

```
RandomReal[{0, 1}, 3]
```

```
{0.113634, 0.804404, 0.39501}
```

```
RandomReal[5, 6]
```

```
{3.21914, 1.62661, 3.25073, 3.2565, 3.93598, 1.50383}
```

```
RandomComplex[]
```

```
0.966065 + 0.188387 i
```

```
RandomComplex[{0, 4 + 3 i}, WorkingPrecision → 20]
```

```
2.3146640493492661801 + 2.1280515765461398320 i
```

```
RandomInteger[{0, 1}, 10]
```

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

```
RandomInteger[{100, 1000}, 10]
```

```
{111, 502, 255, 190, 823, 835, 597, 668, 317, 156}
```

The sequences generated by **Random[]** are not rigorously random. They are generated by a deterministic algorithm, which always gives the same result if starting from the same integer seed.

SeedRandom[s]	reseed the pseudorandom generator with the integer **s**.

```
SeedRandom[143]; RandomReal[{0, 1}, 5]
```

```
{0.110762, 0.364563, 0.163681, 0.753386, 0.977218}
```

```
SeedRandom[143]; RandomReal[{0, 1}, 5]
```

```
{0.110762, 0.364563, 0.163681, 0.753386, 0.977218}
```

13.4.1 Testing hypotheses with the help of random numbers

Example 1

```
test = Sin[Cos[x]] == Cos[Sin[x]]
```

```
Sin[Cos[x]] == Cos[Sin[x]]
```

```
test /.x → RandomReal[]
```

```
False
```

```
test /. x -> 3
```

```
False
```

```
test /. x -> 1 / Sqrt[2]
```

```
False
```

```
N[Sin[Cos[x]] /. x -> π / 4]
N[Cos[Sin[x]] /.  x -> π / 4]
```

```
0.649637
```

```
0.760245
```

Example 2

```
Clear[x];
 Product[ Sin[x + 2 π k / 5], {k, 0, 4} ]
```

$$\text{Cos}\left[\frac{\pi}{10}-x\right]\text{Cos}\left[\frac{\pi}{10}+x\right]\text{Sin}\left[\frac{\pi}{5}-x\right]\text{Sin}[x]\,\text{Sin}\left[\frac{\pi}{5}+x\right]$$

```
Simplify[%]
```

$$\frac{1}{16}\text{Sin}[5\,x]$$

```
test = Product[ N[Sin[x + 2 π k / 5]], {k, 0, 4} ] == Sin[5 x] / 16
```

$$1.\,\text{Cos}[0.314159-1.\,x]\,\text{Cos}[0.314159+x]\,\text{Sin}[0.628319-1.\,x]\,\text{Sin}[x]\,\text{Sin}[0.628319+x]==\frac{1}{16}\text{Sin}[5\,x]$$

```
test /.x → RandomReal[]
```

```
True
```

13.5 Integer and Number-Theoretical Functions

Mod[k,n]	k modulo n (remainder from deviding k by n)
Quotient[m, k]	the quotient of m and k = the integer part of m/k
GCD[n1, n2, ...]	the greatest common divisor of n1, n2,..
LCM[n1, n2, ...]	the least common multiple of n1, n2, ...
IntegerPart[x]	the integer part of x
FractionalPart[x]	the fractional part of x
IntegerDigits[n, b]	the digits of n in base b
RealDigits[x]	a list of the digits in the approximate real number x, together with the number of digits to the left of the decimal point
RealDigits[x,b]	the digits of x in base b

```
Mod[7,3]
```
```
1
```

```
Quotient[7, 3]
```
```
2
```

```
GCD[105, 35, 25]
```
```
5
```

```
LCM[105, 35, 25]
```
```
525
```

IntegerPart[π]	3
FractionalPart[π]	−3 + π
FractionalPart[π//N]	0.141592
IntegerDigits[17, 2]	{1, 0, 0, 0, 1}
IntegerDigits[17, 8]	{2, 1}
IntegerDigits[17, 10]	{1, 7}
IntegerDigits[17, 3]	{1, 2, 2}

```
N[100 π, 18]
```

```
314.159265358979324
```

```
RealDigits[%]
```

```
{{3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3, 2, 4}, 3}
```

```
N[EulerGamma, 19]
```

```
0.5772156649015328606
```

```
RealDigits[%]
```

```
{{5, 7, 7, 2, 1, 5, 6, 6, 4, 9, 0, 1, 5, 3, 2, 8, 6, 0, 6}, 0}
```

FactorInteger[n]	a list of the prime factors of **n**, and their exponents
Divisors[n]	a list of the integers that divide **n**
Prime[n]	the **n**-th prime number
PrimePi[x]	the number of primes less than **x**, π(x)
PrimeQ[n]	False : **n** is not a prime; True : **n** is a prime.

```
29!
```

```
8 841 761 993 739 701 954 543 616 000 000
```

```
FactorInteger[%]
```

```
{{2, 25}, {3, 13}, {5, 6}, {7, 4}, {11, 2}, {13, 2}, {17, 1}, {19, 1}, {23, 1}, {29, 1}}
```

```
PrimeQ[%%]
```

```
False
```

```
Divisors[8979]
```

```
{1, 3, 41, 73, 123, 219, 2993, 8979}
```

```
FactorInteger[8979]
```

```
{{3, 1}, {41, 1}, {73, 1}}
```

Fermat numbers:

```
Fe[n_] = 2^(2^n) + 1
```

$$1 + 2^{2^n}$$

```
Table[{k, Fe[k]}, {k, 0, 7}]
```

```
{{0, 3}, {1, 5}, {2, 17}, {3, 257}, {4, 65 537}, {5, 4 294 967 297},
 {6, 18 446 744 073 709 551 617}, {7, 340 282 366 920 938 463 463 374 607 431 768 211 457}}
```

```
Table[{k, PrimeQ[Fe[k]]}, {k, 0, 9}]
```

```
{{0, True}, {1, True}, {2, True}, {3, True}, {4, True},
 {5, False}, {6, False}, {7, False}, {8, False}, {9, False}}
```

Fermat found that the first 5 Fermat numbers are prime. He conjectured that all Fermat numbers are primes.
However, no Fermat number beyound the first five has turned out to be prime.

```
Table[{k, FactorInteger[Fe[k]]}, {k, 5, 6}]
```

```
{{5, {{641, 1}, {6 700 417, 1}}}, {6, {{274 177, 1}, {67 280 421 310 721, 1}}}}
```

```
FactorInteger[Fe[7]] // Timing  (* iMac9,1  *)
```

```
{0.124011, {{59 649 589 127 497 217, 1}, {5 704 689 200 685 129 054 721, 1}}}
```

```
FactorInteger[Fe[8]] // Timing  (* iMac9,1  *)
```

```
{0.611474, {{1 238 926 361 552 897, 1},
   {93 461 639 715 357 977 769 163 558 199 606 896 584 051 237 541 638 188 580 280 321, 1}}}
```

FactorInteger[Fe[9]] // Timing (* iMac9,1 *)

$Aborted

(Morrison & Brillhart 1970; factorizability proved by F.Klein in 1895.)
F_8 = 1 238 926 361 552 897 x 93 461 639 715 357 977 769 163 558 199 606 896 584 051 237 541 638\
188 580 280 321 (Brent & Pollard 1980; factorizability proved by Morehead and Western in 1909.)
F9 = 2424833 x 7455602825647884208337395736200454918783366342657 x 74164006262753080\
152478714190193747405994078109751902390582131614441575950470500809281871169394073 7
F_{10} - F_{21} are known to factorize. Not any prime Fermat number > F_4 has been found up to now.
F_{23471} consists of 10^{7000} decimal places, 5 x 2^{23473} + 1 is a prime factor comprising 7067 decimal places.
(Bild der Wissenschaft, Nov.1990).

Mersenne numbers

Mersenne numbers, $2^p - 1$, where p is a prime, were also conjectured to be primes. A counter example is the following one:

```
me = 2^67 - 1
```

```
147 573 952 589 676 412 927
```

```
FactorInteger[me]  (* Cole,  1903 *)
```

```
{{193 707 721, 1}, {761 838 257 287, 1}}
```

At the moment (since 2016-01-07), the largest known Mersenne prime is $2^{74\,207\,281}$-1, the 49-th (?) known
Mersenne prime, a number comprising 22 338 618 digits. It was found by the Great Internet Mersenne Prime Search
(GIMPS). This and related information may be found on the web sites:
http://www.mersenne.org/primes/ or https://de.wikipedia.org/wiki/Mersenne-Primzahl
Mathematica is capable to evaluate $2^{74\,207\,281}$-1. The output covers nearly 4000 pages.

An **option** may be used in **FactorInteger[]** so that one gets small factors rather fast. The number given
below comprises 215 bits, i.e. 65 decimal places. This is uninteresting for cryptologists (see below).

```
?? FactorInteger
```

FactorInteger[*m*] gives a list of the prime factors of the integer *m*, together with their exponents
FactorInteger[*m*, *k*] does partial factorization, pulling out at most *k* distinct factors >>

```
Attributes[FactorInteger] = {Listable, Protected}
```

```
Options[FactorInteger] = {GaussianIntegers → False}
```

```
fi = Timing[FactorInteger[
    18 402 786 717 172 645 644 535 779 054 968 269 097 752 223 096 614 652 509 534 106 463,
    Automatic]]
```

```
{2.327536, {{3, 2}, {7, 2}, {1 406 956 463 719, 1},
```

Incomplete factorization, but fast. As shown below, the last factor is not a prime. Finding its prime factors
takes not very long:

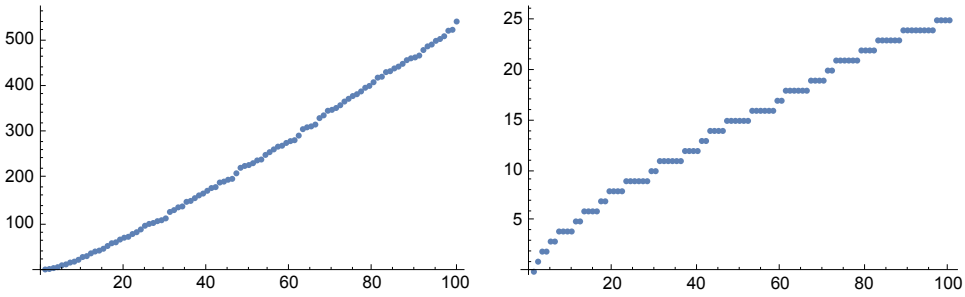
```
PrimeQ[fi[[2, 4, 1]]] // Timing
```

```
{0.000114, False}
```

```
Timing[FactorInteger[fi[[2, 4, 1]]]]
```

```
{4.201682,
 {{109 522 253 082 122 323 037 981, 1}, {270 808 302 454 534 186 160 412 037, 1}}}
```

```
p1 = ListPlot[Table[Prime[n], {n, 100}]];
p2 = ListPlot[Table[PrimePi[n], {n, 100}]];
Show[GraphicsRow[{p1, p2}], ImageSize → 500]
```



The program for the function **PrimePi[x],** giving the number of primes smaller than x**,** is implemeted for $x < 10^{14}$, only.
But there are asymptotic formulae and estimates:

$$\pi(x) \simeq \frac{x}{\text{Log}[x]} \simeq \text{li}(x), \quad \pi(x) < \frac{3\,x+2\,x\,\text{Log}[x]}{2\,\text{Log}[x]^2}$$

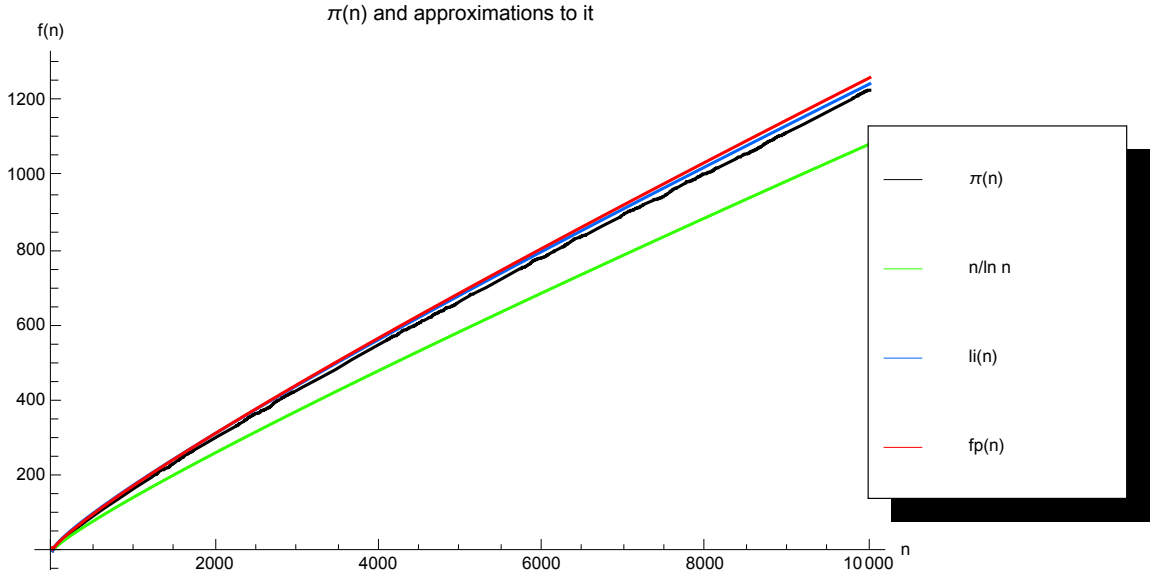
where the logarithmic integral $\text{li}(x) = \int_0^x \frac{1}{\log(x)}\,dx$ gives a better aproximation than x/Log[x].

```
Needs["PlotLegends`"]
```

```
Clear[fg, fi, fp]
```

```
fg[x_] = x / Log[x]; fi[x_] = LogIntegral[x]; fp[x_] = (3 x + 2 x Log[x]) / (2 Log[x]^2);
```

```
Plot[{PrimePi[x], fg[x], fi[x], fp[x]}, {x, 2, 10^4}, AxesLabel → {"n", "f(n)"},
 PlotStyle → {GrayLevel[0], Hue[0.3`], Hue[0.6`], Hue[0]}, ImageSize → 620,
 BaseStyle → {FontSize → 9}, LegendPosition → {0.85`, -0.45`},
 PlotLabel → "π(n) and approximations to it",
 PlotLegend → {"π(n)", "n/ln n", "li(n)", "fp(n)"}, ImageSize → 600]
```



```
{nx = 10^10 // N, fg[nx] // N, fi[nx // N] // N, fp[nx] // N, PrimePi[nx]}
```

$$\{1.\times 10^{10},\ 4.34294\times 10^{8},\ 4.55056\times 10^{8},\ 4.62586\times 10^{8},\ 455\,052\,511\}$$

```
IntegerPart[{nx = 10^14 // N, fg[nx], fi[nx], fp[nx], PrimePi[nx]}]
```

```
{100 000 000 000 000, 3 102 103 442 166,
 3 204 942 065 691, 3 246 449 128 654, 3 204 941 750 802}
```

```
IntegerPart[{nx = 10^15 // N, fg[nx], fi[nx], fp[nx], PrimePi[nx]}]
```

PrimePi::largp: Argument 1.`*^15 in PrimePi[1.×10^15] is too large for this implementation. >>

$$\{1\,000\,000\,000\,000\,000,\ 28\,952\,965\,460\,216,\ 29\,844\,571\,475\,287,\ 30\,210\,376\,773\,627,\ \text{IntegerPart}\left[\text{PrimePi}\left[1.\times 10^{15}\right]\right]\}$$

PrimePi::largp :
Argument $1.\times 10^{15}$ in PrimePi$\left[1.\times 10^{15}\right]$ is too large for this implementation. >>

13.5.1 Primes and Cryptology

The factoring of large numbers into primes has found technical applications in cryptology.The RSA method
invented by R.L.**R**ivest,A.**S**hamir and L.**A**dleman in 1978 uses a large number, the encryption key, which
is the product of two large primes.Only the product is needed for the encryption; this number and the algorithm
or program for encryption are public, so everybody can get and use them. The decryption program is also public.
But only a person knowing the two factors of the encryption key has the means to decrypt the encrypted text to
get the orignal message. Anybody wanting to find out the message illegally must succeed in the factorization.
The encryption is the shurer the larger the factors.The minimum number of bits of a product deemed shure is 700;
the standard is 1000, corresponding to 300 decimal places.The best algorithms available at present need several CPU
years to factor a number of 120 places.
A less involved scheme is the 56-bit DES encryption standard. It was cracked by the Electronic Frontier
Foundation's "Deep Crack" computer in January 1999 in a record time of 22 hours and 15 minutes with
a worldwide network of nearly 100 000 PC's.

With the help of **PrimeQ[]** one may find a number with the given number of figures which one expects to be a prime.
However the tests employed by **PrimeQ[]** are not 100% secure. The answer False is always correct, but True may be
in error with a small probability. There are more elaborate (and more time consuming) tests, whose results is certain.
The package **PrimalityProving`** contains a much slower algorithm which has been proved correct for all n.
This package is not meant to replace the built-in primality tester `PrimeQ` but rather to allow you to be completely secure
that a number is truly prime. The package should be used only to certify results after all the number theoretic work has been done.

```
<< PrimalityProving`
```

```
PrimeQ[1 000 000 000 000 037]
```

```
True
```

```
ProvablePrimeQ[1 000 000 000 000 037]
```

```
True
```

```
PrimeQ[1 000 000 000 000 038]
```

```
False
```

```
FactorInteger[1 000 000 000 000 038]
```

```
{{2, 1}, {3, 1}, {11 593, 1}, {34 679, 1}, {414 559, 1}}
```

Up to date **information on primes** may be found at the website: http://www.utm.edu/research/primes/ .
An elementary description of the **RSA method:**
http://www.math.uni-oldenburg.de/personen/schmale/vom_Tildebereich/dateien/Oeffentlich/Leitdatei-RSA.html
A corresponding *Mathematica* notebook RSA.nb may be obtained from BS.
Quantum Computers and Factoring: D.A. Mermin: What has quantum mechanics to do with factoring?
Physics Today, April 2007, pp. 8 - 9.

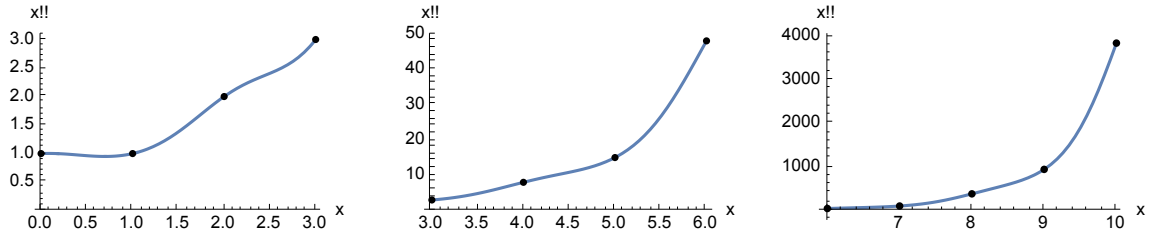
13.6 Combinatorial Functions

n!	**factorial**	=	**n.(n-1).(n-2) ... 2.1 = Gamma[n + 1]**
n!!	**double factorial**	=	**n·(n-2)·(n-4)· ... ·4·2 = n·(n-2)·(n-4)· ... ·3·1**
	= $(2/\pi)^{\frac{1}{4}(1-\text{Cos}[n\pi])}\ 2^{n/2}\ \text{Gamma}\left[1+\frac{n}{2}\right]$		Interpolation for non-integer arguments
Binomial[n, k]	**binomial coefficient**	= $\binom{n}{k}$ =	$\frac{n!}{k!\,(n-k)!}$
Multinomial[n1, n2, ..., nn]	**multinomial coefficient**	=	$\frac{(n1+n2+\ldots+nn)!}{n1!\,n2!\ldots nn!}$
Signature[{n1, n2, ...}]	**the signature of the permutation**		**n1,n2,...**

The interpolation of the factorial and of the double factorial for arbitrary (non-integer) n is expressed by the Gamma function.

Signature[{1,2,3}]	1
Signature[{3,2,1}]	-1
n!	n!
5!	120
5.3!	201.813
Gamma[6.3]	201.813
n!!	n!!
5!!	15 = 5.3.1
6!!	48 = 6.4.2
5.3!!	20.6494800168865788

```
Clear[x, ff]
intv = Table[Point[{n, n!!}], {n, 0, 10}];
pts = 0.025`; ff[x_] = (2/π)^(1/4 (1-Cos[x π])) 2^(x/2) Gamma[1 + x/2];
p1 = Plot[ff[x], {x, 0, 3}, AxesLabel → {"x", "x!!"},
    PlotRange → {0, 3.1}, Epilog → {PointSize[pts], intv[[Range[1, 4]]]}];
p2 = Plot[ff[x], {x, 3, 6}, AxesLabel → {"x", "x!!"},
    PlotRange → {0, 50}, Epilog → {PointSize[pts], intv[[Range[4, 7]]]}];
p3 = Plot[ff[x], {x, 6, 10}, AxesLabel → {"x", "x!!"},
    PlotRange → All, Epilog → {PointSize[pts], intv[[Range[7, 11]]]}];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 600]
```



Binomial[n, 2]	$\frac{1}{2} n (n - 1)$
Binomial[5, 2]	10
Binomial[5, 6]	0
Binomial[n/2, 2]	$\frac{1}{4} \left(-1 + \frac{n}{2}\right) n$
Binomial[3/2, 2]	$-\frac{3}{256}$

```
Clear[x, y, z]
p = Expand[(x + y + z)^5]
```

$$x^5 + 5\,x^4\,y + 10\,x^3\,y^2 + 10\,x^2\,y^3 + 5\,x\,y^4 + y^5 + 5\,x^4\,z + 20\,x^3\,y\,z + 30\,x^2\,y^2\,z + 20\,x\,y^3\,z + 5\,y^4\,z + 10\,x^3\,z^2 + 30\,x^2\,y\,z^2 + 30\,x\,y^2\,z^2 + 10\,y^3\,z^2 + 10\,x^2\,z^3 + 20\,x\,y\,z^3 + 10\,y^2\,z^3 + 5\,x\,z^4 + 5\,y\,z^4 + z^5$$

```
Coefficient[p, x^2 y^2 z]
```

30

```
Multinomial[2,2,1]
```

30

13.6.1 Vector-Coupling Coefficients Used in Quantum Mechanics

Vector-coupling coefficients (= Wigner coefficients, Clebsch-Gordan coeffients) are needed for coupling the eigenfunctions ot two angular momenta to give such a linear combination of these functions that the resulting functions are eigenfunctions of the operator of total angular momentum. Wigner 3j-symbols differ from the
vector-coupling coefficients by a factor; they are more symmetrical. 6j-symbols are needed for recoupling 3 angular momenta.
The eigenfunctions of the operators of each single angular momentum are:

$$j_i^2\,|j_i,m_i\rangle = j_i\,(j_i + 1)\,|j_i,m_i\rangle, \quad i = 1,2;$$

$$j_{iz}\,|j_i,m_i\rangle = m_i\,|j_i,m_i\rangle, \quad -j_i \le m_i \le j_i.$$

The operator of total angular momentum is

$$\mathbf{J} = \mathbf{j}_1 + \mathbf{j}_2$$

with

$$\mathbf{J}_z = \mathbf{j}_{1z} + \mathbf{j}_{2z}$$

and

$$\mathbf{J}^2 = \mathbf{j}_1^2 + \mathbf{j}_2^2 + 2\,(\mathbf{j}_1.\mathbf{j}_2);$$

the simultaneous eigenfunctions $|J,M\rangle$ of the last two operators give:

$$\mathbf{J}^2\,|J,M\rangle = J\,(J+1)\,|J,M\rangle\,;$$

$$J_z\,|J,M\rangle = M\,|J,M\rangle\,, \qquad -J \le M \le J;$$

$$j_1 + j_2 \ge J \ge |j_1 - j_2|.$$

These eigenfunctions are linear combinations of products of the eigenfunctions of the single operators; the coefficients are the vector-coupling coefficients.

$$|J,M\rangle = \sum_{m_1,m_2} |j_1,m_1\rangle\,|j_2,m_2\rangle\,(j_1\ m_1\ j_2\ m_2|\ j_1\ j_2\ J\ M)$$

In many cases j_3 and m_3 are used in place of J and M.

The Wigner 3j-symbols are related to the vector-coupling coefficients by:

$$\begin{pmatrix} j1 & j2 & j3 \\ m1 & m2 & m3 \end{pmatrix} = (-1)^{j_1-j_2-m_3}\,(2\,j_3+1)^{-1/2}\,(j_1 m_1 j_2 m_2\ |\ j_1 j_2 j_3 - m_3)$$

A.R. Edmonds: Angular Momentum in Quantum Mechanics.
Princeton Univ.Press 1957. (Dt.Ausg. BI TB 53/53a)

A. Lindner: Drehimpuls in der Quantenmechanik. Teubner Studienb"ucher, 1984.

M.Rotenberg, R. Bivins, N. Metropolis, J.K.Wooten: The 3-j and 6-j-Symbols.
The Technology Press. Cambridge, Mass. 1959

ClebschGordan[{j1,m1}, {j2,m2}, {j3,m3}]

Clebsch Gordan Coefficient, Wigner Coefficient, Vector Coupling Coefficient

ThreeJSymbol[{j1,m1}, {j2,m2}, {j3,m3}]

$\begin{pmatrix} j1 & j2 & j3 \\ m1 & m2 & m3 \end{pmatrix}$, Wigner 3j-Symbol

SixJSymbol[{j1,j2,j3}, {j4,j5,j6}]

$\begin{Bmatrix} j1 & j2 & j3 \\ j4 & j5 & j6 \end{Bmatrix}$, Wigner 6j-Symbols

```
ClebschGordan[{l, m}, {1 / 2, 1 / 2}, {l + 1 / 2, m + 1 / 2}]
```

$$\begin{cases} (-1)^{2\,l+2\,m}\,\sqrt{1+l}\,\sqrt{\frac{1+l+m}{1+3\,l+2\,l^2}} & l \ge 0 \\ 0 & \text{True} \end{cases}$$

```
ThreeJSymbol[{l, m}, {1 / 2, 1 / 2}, {l + 1 / 2, -(m + 1 / 2)}]
```

$$\begin{cases} \frac{(-1)^{1+m}\,\sqrt{\frac{1+l+m}{(1+l)\,(1+2\,l)}}}{\sqrt{2}} & l \ge 0 \\ 0 & \text{True} \end{cases}$$

The 3j-Symbol has value zero if the sum of the three magnetic quantum numbers $\mathbf{m_1, m_2, m_3}$ (given in the lower row of the symbol, or as the second element of each argument list of the command above) is not zero. In this case *Mathematica* programs are troublesome. Some computers give an expression with Gamma functions, which runs into trouble on evaluation and nay give wrong results.

```
ThreeJSymbol[{l, m}, {1 / 2, 1 / 2}, {l + 1 / 2, (m + 1 / 2)}]
```

$$\begin{cases} \frac{(-1)^{1+m}\,\sqrt{\frac{(1+m)!\,(1+l+m)!}{(1+l)\,(1+2\,l)\,\left((-1+l-m)!\right)^2\,((1+2\,m)!)^2}}}{\sqrt{2}} & m == -\frac{1}{2}\ \&\&\ l \ge 0 \\ 0 & \text{True} \end{cases}$$

```
% /. m -> 0
```

0

```
ThreeJSymbol[{l, m}, {1 / 2, 1 / 2}, {l + 1 / 2, -(m + 1 / 2)}]
```

$$\begin{cases} \frac{(-1)^{1+m}\sqrt{\frac{1+l+m}{(1+l)\,(1+2\,l)}}}{\sqrt{2}} & l \geq 0 \\ 0 & \text{True} \end{cases}$$

```
ThreeJSymbol[{5, 0}, {2, 1}, {2, -1}]
```

ClebschGordan::tri: ThreeJSymbol[{5, 0}, {2, 1}, {2, −1}] is not triangular. >>

0

ClebschGordan::tri :
ThreeJSymbol[{5, 0}, {2, 1}, {2, -1}] is not triangular. >>

The Wigner 3J coefficient is zero if the sum of the three magnetic quantum numbers m1, m2, m3 is not zero.
There is the additional condition that the lengthes j1, j2, j3 must be the sides of a (true or degenerate triangle).

```
ThreeJSymbol[{4, 0}, {2, 1}, {2, -1}] // Together
```

$$\frac{2\sqrt{\frac{2}{35}}}{3}$$

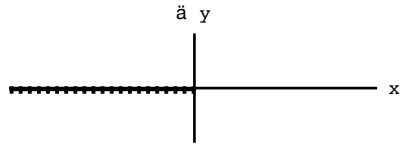
```
ThreeJSymbol[{4, -1}, {2, 2}, {2, -1}] // Together
```

$$-\frac{1}{3\sqrt{14}}$$

13.7 Roots and Fractional Powers

$$\text{Sqrt}[z] = |z|^{1/2}\, e^{i\,\arg[z]/2}, \qquad -\pi < \arg(z) \leq \pi.$$

is evaluated for complex values z as indicated. So for real positive arguments the positive branch is chosen:
The branch cut is fixed along the negative real axis: (- ∞, 0), as shown below.




```
Sqrt[(-2)^2]
```

2

```
Sqrt[-5]
```

$i\sqrt{5}$

13.7.1 Evaluations of the square root for arguments near the branch cut require some care

```
ϵ = 10^-10;
```

```
Sqrt[-5 + I ϵ]
```

$$\frac{\sqrt{-50\,000\,000\,000 + i}}{100\,000}$$

```
Sqrt[-5 - I ϵ]
```

$$\frac{\sqrt{-50\,000\,000\,000 - i}}{100\,000}$$

```
N[Sqrt[-5 + I ϵ], 5] // Chop
```

```
2.2361 i
```

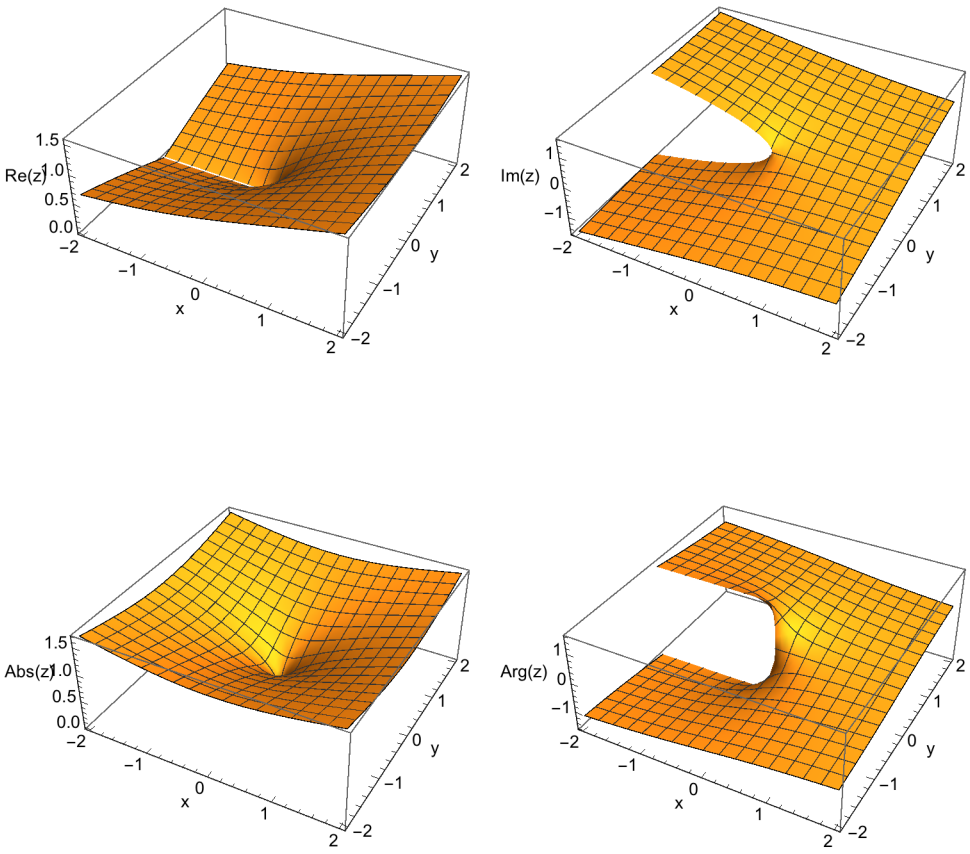
```
N[Sqrt[-5 - I ϵ], 5] // Chop
```

```
-2.2361 i
```

13.7.2 Surfaces of the various parts of the square root function and the Rieman surface

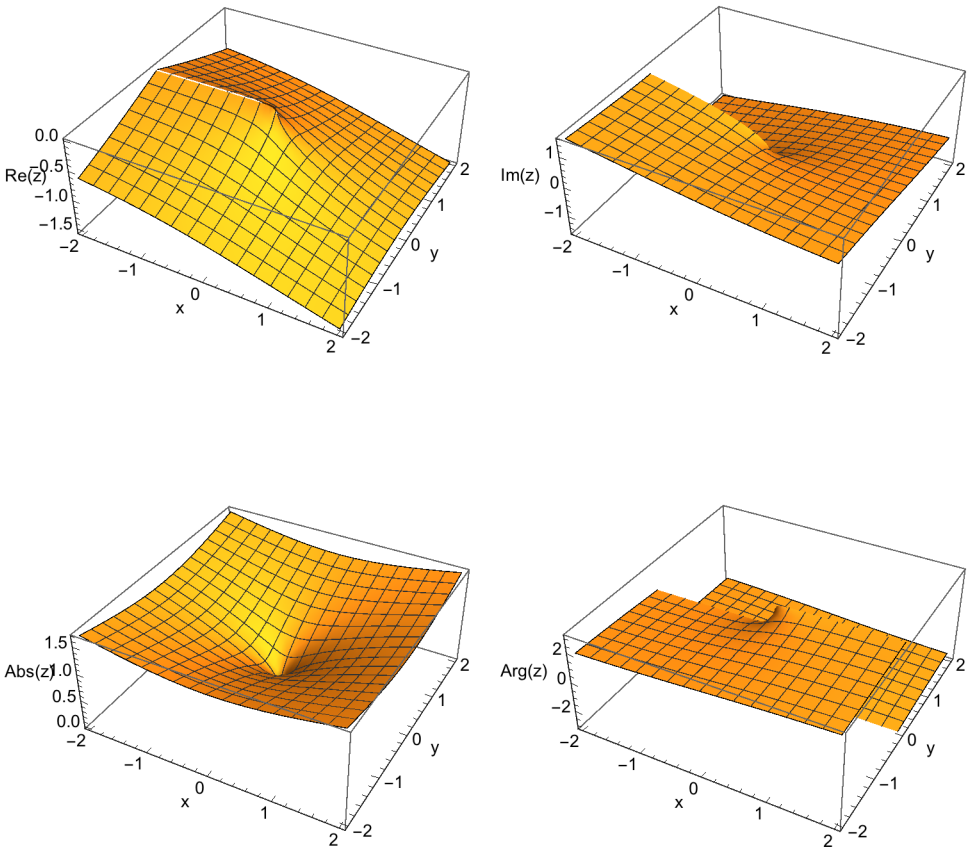
The positive branch of the square root:

```
cro = Plot3D[Re[√(x + i y)], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "Re(z)"}];
cio = Plot3D[Im[√(x + i y)], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "Im(z)"}];
sa = Plot3D[Abs[√(x + i y)], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "Abs(z)"}];
aro =
  Plot3D[Arg[√(x + i y)], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "Arg(z)"}];
Show[GraphicsGrid[{{cro, cio}, {sa, aro}}], ImageSize → 500]
```



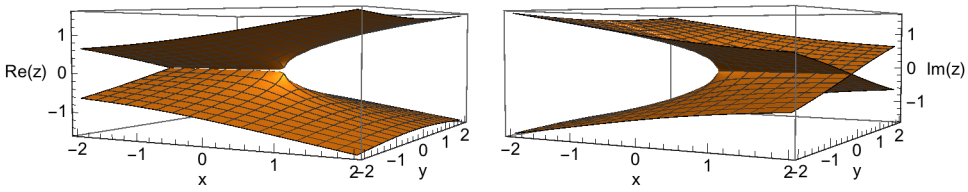
The negative branch of the square root:

```
cru =
  Plot3D[Re[-√(x + i y)], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "Re(z)"}];
ciu = Plot3D[Im[-√(x + i y)], {x, -2, 2}, {y, -2, 2},
   AxesLabel → {"x", "y", "Im(z)"}];
sa = Plot3D[Abs[√(x + i y)], {x, -2, 2}, {y, -2, 2},
   AxesLabel → {"x", "y", "Abs(z)"}];
aru = Plot3D[Arg[-√(x + i y)], {x, -2, 2}, {y, -2, 2},
   AxesLabel → {"x", "y", "Arg(z)"}];
Show[GraphicsGrid[{{cru, ciu}, {sa, aru}}], ImageSize → 500]
```



A genuine representation of Riemann surface of the square root would require a four-dimnesional space (x, y, u, v}.
We can only show two three-dimensional drawings for the real and the imaginary part.

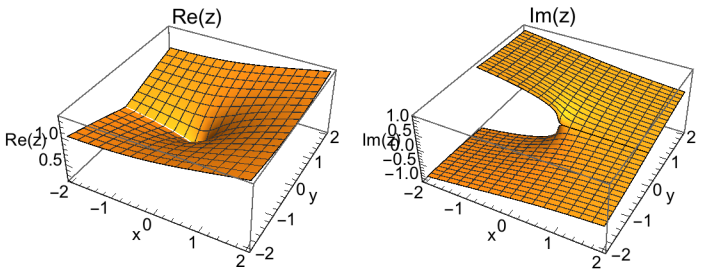
```
cr = Show[cro, cru, PlotRange → All, ViewPoint → {1.3, -2.4, .13},
    AxesEdge → {{-1, -1}, {1, -1}, {-1, -1}}];
ci = Show[cio, ciu, PlotRange → All,
    ViewPoint → {1.3, -2.4, .13}, AxesEdge → {{-1, -1}, {1, -1}, {1, 1}}];
GraphicsRow[{cr, ci}, ImageSize → 500]
```



$$z^s = |z|^s e^{i \arg[z]}, \quad -\pi < \arg(z) \le \pi.$$

The general power is multivalued if s is not an integer.
The branch cut is again the negative half of the real axis: (-∞, 0), s. first graph of this subsection.

```
sr = Plot3D[Re[(x + i y)^(1/3)], {x, -2, 2}, {y, -2, 2},
  AxesLabel → {"x", "y", "Re(z)"}, PlotLabel → "Re(z)"];
siu = Plot3D[Im[(x + i y)^(1/3)], {x, -2, 2}, {y, -2, -0.001`},
  AxesLabel → {"x", "y", "Im(z)"}, PlotLabel → "Im(z)"];
sil = Plot3D[Im[(x + i y)^(1/3)], {x, -2, 2}, {y, 0.001`, 2},
  AxesLabel → {"x", "y", "Im(z)"}, PlotLabel → "Im(z)"];
sl = Show[siu, sil, PlotRange → All]; Show[GraphicsRow[{sr, sl}]]
```



13.8 Exponential Functions and Logarithms

`Exp[z]`	exponential function	
`Log[z]`	natural logarithm	(basis e)
`Log[b, z]`	logarithm for basis	b

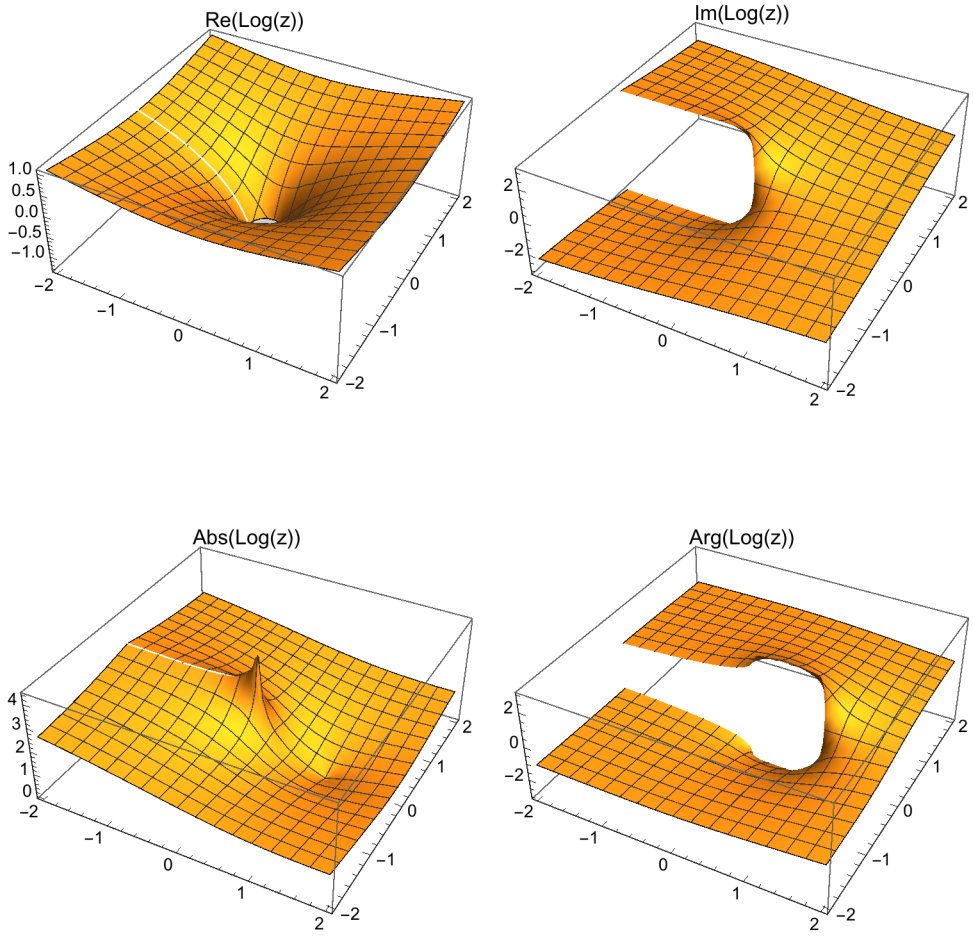
The branch cut of the logarithm is again the negative half of the real axis: (-∞, 0), s. first graph of preceeding subsection.

$$\text{Log}[z] = \text{Log}[|z|] + i\,\text{Arg}[z], \quad -\pi < \text{Arg}[z] \le \pi.$$

Input	Output
`Exp[2.5 + 1.3 I]`	`+3.25880 + 11.73854I`

```
Log[2.]                          0.693147
Log[-2.]                         0.693147 + 3.14159 I
Log[-2. + I 10^-5]               0.693147 + 3.14159 I
Log[-2. - I 10^-5]               0.693147 - 3.14159 I
Log[2, 1024]                          10
```

```
sr = Plot3D[Re[Log[x + i y]], {x, -2, 2}, {y, -2, 2}, PlotLabel → "Re(Log(z))"];
si = Plot3D[Im[Log[x + i y]], {x, -2, 2}, {y, -2, 2}, PlotLabel → "Im(Log(z))"];
sa = Plot3D[Abs[Log[x + i y]], {x, -2, 2}, {y, -2, 2}, PlotLabel → "Abs(Log(z))"];
sg = Plot3D[Arg[Log[x + i y]], {x, -2, 2},
   {y, -2, 2}, PlotRange → π {-1, 1}, PlotLabel → "Arg(Log(z))"];
Show[GraphicsRow[{sr, si}], ImageSize → 500]
Show[GraphicsRow[{sa, sg}], ImageSize → 500]
```



13.9 Trigonometric and Inverse Trigonometric Functions

`Cos[z]`	`ArcCos[z]`
`Sin[z]`	`ArcSin[z]` = $\int_0^z \frac{1}{\sqrt{1-t^2}}\,dt$
`Tan[z]`	`ArcTan[z]` = $\int_0^z \frac{i}{2}\left(\frac{1}{t+i}-\frac{1}{t-i}\right)dt$
`Cot[z] = 1/Tan[z]`	`ArcCot[z]`
`Csc[z] = 1/Sin[z]`	`ArcCsc[z]`
`Sec[z] = 1/Cos[z]`	`ArcSec[z]`
	`ArcTan[x,y]` is valid in the whole x, y-plane.

Arguments of trigonometric functions in radians; results of inverse trigonometric functions in radians. Arguments may also be complex numbers. Symbolic arguments give results e.g. in the followin cases:

```
Table[Sin[π / n], {n, 16}]
```

$$\left\{0,\ 1,\ \frac{\sqrt{3}}{2},\ \frac{1}{\sqrt{2}},\ \sqrt{\frac{5}{8}-\frac{\sqrt{5}}{8}},\ \frac{1}{2},\ \text{Sin}\left[\frac{\pi}{7}\right],\ \text{Sin}\left[\frac{\pi}{8}\right],\ \text{Sin}\left[\frac{\pi}{9}\right],\right.$$

$$\left.\frac{1}{4}\left(-1+\sqrt{5}\right),\ \text{Sin}\left[\frac{\pi}{11}\right],\ \frac{-1+\sqrt{3}}{2\sqrt{2}},\ \text{Sin}\left[\frac{\pi}{13}\right],\ \text{Sin}\left[\frac{\pi}{14}\right],\ \text{Sin}\left[\frac{\pi}{15}\right],\ \text{Sin}\left[\frac{\pi}{16}\right]\right\}$$

```
ArcTan[Sqrt[3]]
```

$$\frac{\pi}{3}$$

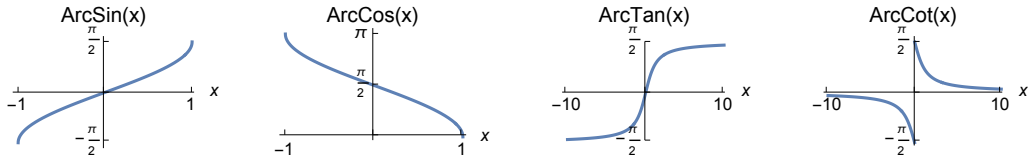
```
ArcSin[2 Sqrt[3] / 2]
```

$$\text{ArcSin}\left[\sqrt{3}\right]$$

In the second line below we indicate for which arguments the functions in the first line have symbolic values:

Sin, Cos, Tan, Cot:	ArcSin, ArcCos:	ArcTan, ArcCot:
0, π, $\pi/2$, $\pi/3$, $\pi/4$, $\pi/5$, $\pi/6$, $\pi/10$, $\pi/12$;	0, 1/2, $\frac{\sqrt{3}}{2}$, 1;	0, $\frac{1}{\sqrt{3}}$, 1, $\sqrt{3}$.

```
SetOptions[Plot, DisplayFunction → Identity];
s = Plot[ArcSin[x], {x, -1, 1}, AxesLabel → {x, None},
  PlotLabel → "ArcSin(x)", Ticks → {{-1, 0, 1}, {-π/2, 0, π/2}}];
c = Plot[ArcCos[x], {x, -1, 1}, AxesLabel → {x, None},
  PlotLabel → "ArcCos(x)", Ticks → {{-1, 0, 1}, {0, π/2, π}}];
t = Plot[ArcTan[x], {x, -10, 10}, AxesLabel → {x, None}, PlotLabel → "ArcTan(x)",
  PlotRange → {-π/2, π/2}, Ticks → {{-10, 0, 10}, {-π/2, 0, π/2}}];
o = Plot[ArcCot[x], {x, -10, 10}, AxesLabel → {x, None},
  PlotLabel → "ArcCot(x)", Ticks → {{-10, 0, 10}, {-π/2, 0, π/2}}];
Show[GraphicsGrid[{{s, c, t, o}}], ImageSize → 550]
```

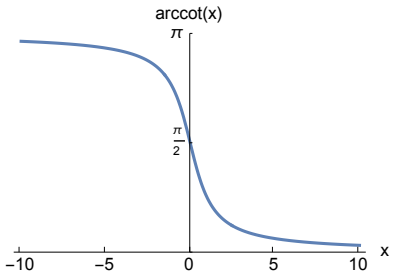


13.9.1 Principal branch of ArcCot[x]

Mathematica defines the principal branch of ArcCot[x] in a way (s. figure above) differing from the usual convention [AS] plotted below

```
Clear[x]
usualarccot[x_] := If[x < 0, π + ArcCot[x], ArcCot[x]]
```

```
Plot[usualarccot[x], {x, -10, 10}, Ticks → {Range[-10, 10, 5], π Range[0, 2] / 2},
 PlotRange → {0, π}, AxesLabel → {"x", "arccot(x)"}, ImageSize → 200]
```



13.9.2 Values of ArcTan[x, y] in the whole x,y-plane

Note: ArcTan[x,y] works correctly in all four quadrants of the x, y-plane; whereas ArcTan[y/x] is limited to the
1st and 4th quadrant (principal branch of ArcTan[], s. figure above) and example below

```
ArcTan[-2, 3]
```

$$\pi - \text{ArcTan}\left[\frac{3}{2}\right]$$

```
N[%]
```

```
2.1588
```

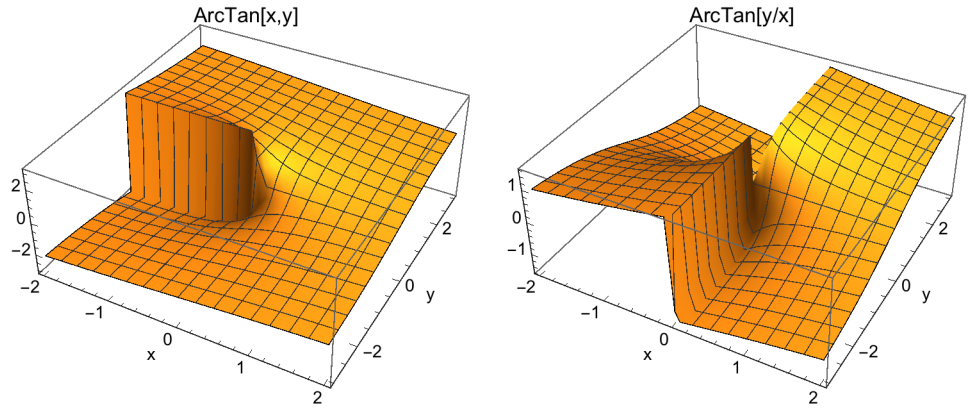
```
ArcTan[-3 / 2]
```

$$-\text{ArcTan}\left[\frac{3}{2}\right]$$

```
N[%]
```

```
-0.982794
```

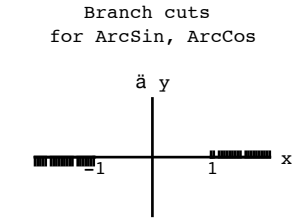
```
p1 = Plot3D[ArcTan[x, y], {x, -2, 2}, {y, -3, 3},
    AxesLabel → {"x", "y", ""}, PlotLabel → "ArcTan[x,y]"];

p2 = Plot3D[ArcTan[y/x], {x, -2, 2}, {y, -3, 3},
    AxesLabel → {"x", "y", ""}, PlotLabel → "ArcTan[y/x]"];

Show[GraphicsRow[{p1, p2}], ImageSize → 500]
```

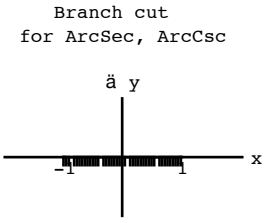


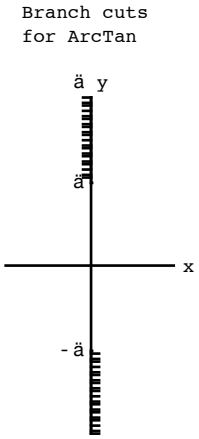
13.9.3 Branch points of the inverse trigonometric functions

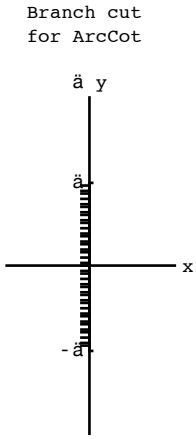
The Inverse trigonometric functions are multivalued functions. The branch points may be found from the integral

ArcTan[z],
ArcCot[z].*Mathematica* gives the values for the principal branches.The definition used for the ArcCot[x] does
not correspond to the usual definition. The branch cuts are shown in the figures:









Examples:

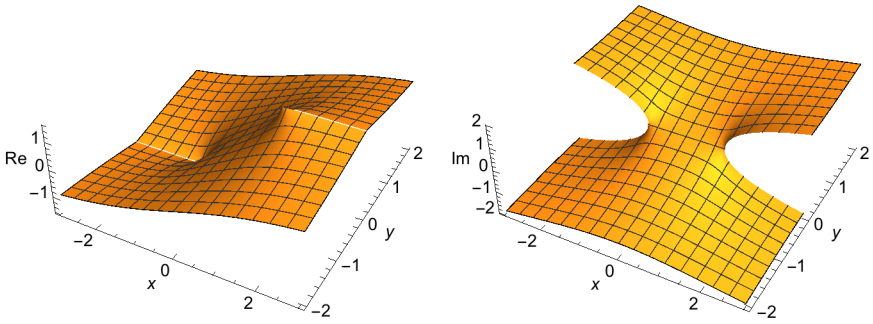
```
{ArcSin[2 + 0.1 I], ArcSin[2 - 0.1 I], ArcSin[2.] }
```

$\{1.51316 + 1.31888\,i,\ 1.51316 - 1.31888\,i,\ 1.5708 - 1.31696\,i\}$

Sin[Pi / 3]	$\frac{\sqrt{3}}{2}$	**ArcSin[3^(1 / 2) / 2]**	$\frac{\pi}{3}$
Tan[Pi / 6]	$\frac{1}{\sqrt{3}}$	**ArcTan[3^(-1 / 2)]**	$\frac{\pi}{6}$
ArcTan[Infinity]	$\frac{\pi}{2}$	**ArcTan[1]**	$\frac{\pi}{4}$
ArcSec[Infinity]	$\frac{\pi}{2}$	**ArcCos[2.]**	0. + 1.31696 I
Cos[2. + I 3]	-4.18963 - 9.10923 I		
ArcSin[Sin[4.5]]	-1.35840	**ArcSin[Sin[4.5 - 2π]]**	-1.35841

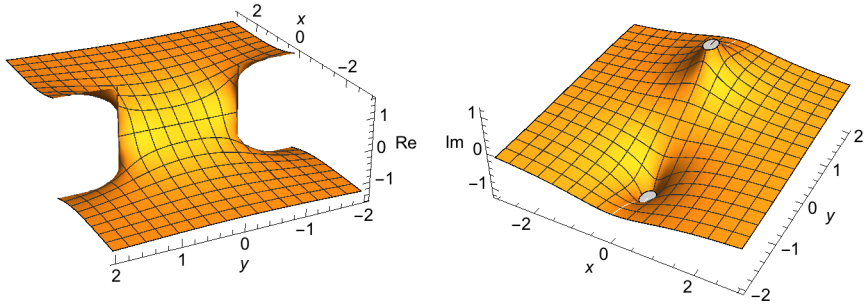
```
SetOptions[ Plot3D, Boxed -> False];
sr = Plot3D[ Re[ArcSin[x + I y]], {x, -3, 3}, {y, -2, 2},
        AxesLabel -> {x,y,Re}, PlotPoints -> 50];
si = Plot3D[ Im[ArcSin[x + I y]], {x, -3, 3}, {y, -2, 2},
        AxesLabel -> {x,y,Im}, PlotPoints -> 50 ];
Print["                              ArcSin"]
Show[GraphicsRow[{sr,si}], ImageSize ->  450]
```

```
                          ArcSin
```




```
sr = Plot3D[ Re[ArcTan[x + I y]], {x, -3, 3}, {y, -2, 2},
        AxesLabel -> {x,y,Re}, PlotPoints -> 50,
            ViewPoint -> {-4,2,2} ];
si = Plot3D[ Im[ArcTan[x + I y]], {x, -3, 3}, {y, -2, 2},
        AxesLabel -> {x,y,Im}, PlotPoints -> 50 ];
Print["                            ArcTan"]
Show[GraphicsRow[{sr,si}], ImageSize ->  450]
```

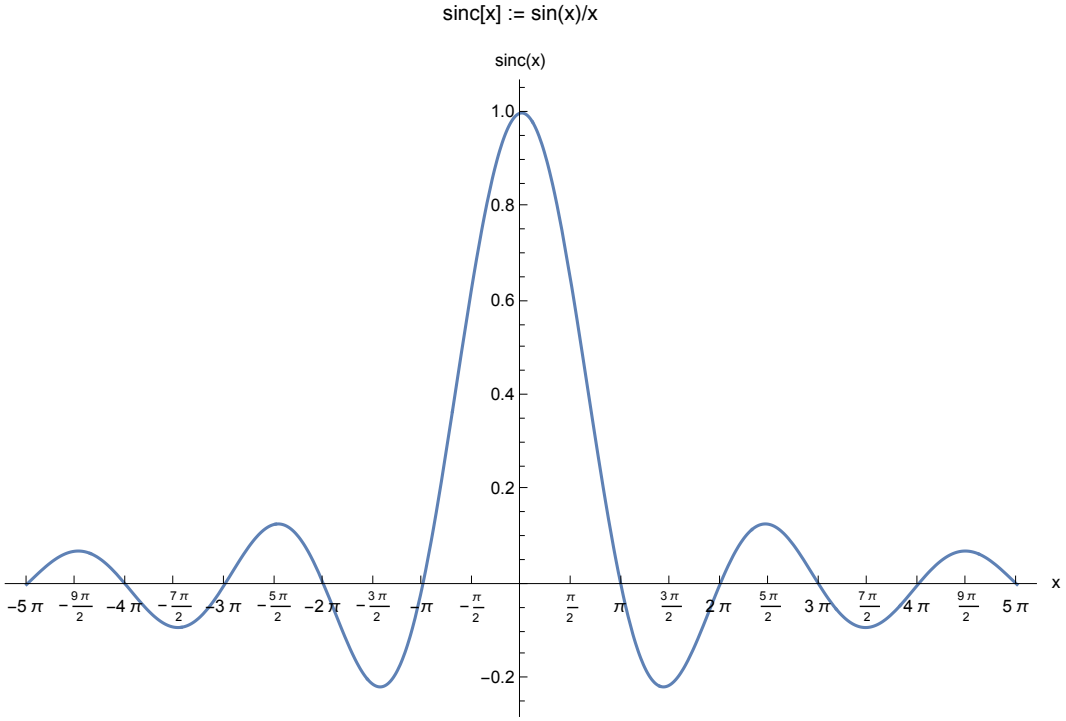
```
                        ArcTan
```



13.9.5 The function Sinc[x]

Sinc[x] gives Sinc[x] := sin (x)/x, x in radians .
Attributes [Sinc] = {Listable, NumericFunction, Listable}
The function sinc[x] = Sinc[x] describes single – slit diffraction pattern for a 4λ slit.

```
Plot[Sinc[x], {x, -5 π, 5 π},
 Ticks → {π Range[-5, 5, 1 / 2], Automatic}, AxesLabel → {"x", "sinc(x)"},
 PlotLabel → "sinc[x] := sin(x)/x\n", PlotRange → All, ImageSize → 550]
```



13.10 Hyperbolic And Inverse Hyperbolic Functions

```
Cosh[z]                    ArcCosh[z] # log[z + √(z2 - 1)]
Sinh[z]                    ArcSinh[z] # log[z + √(z2 + 1)]
Tanh[z]                    ArcTanh[z] # log[(1+z)/(1-z)]/2
Coth[z] = 1/Tanh[z]        ArcCoth[z] # log[(z+1)/(z-1)]/2
Csch[z] = 1/Sinh[z]        ArcCsch[z] # log[z-1 + √(z-2 + 1)]
Sech[z] = 1/Cosh[z]        ArcSech[z] # log[z-1 + √(z-2 - 1)]
```

Symbolic Arguments:

Cosh, Sinh, Tanh, Coth, Csch, Sech: 0, 1, ∞ .
ArcSinh[0] = 0, ArcCosh[1] = 0, ArcCosh[0] = $i\,\pi/2$;
ArcTanh[0] = 0, ArcTanh[Infinity] = $-i\,\pi/2$;
ArcCoth[0] = $i\,\pi/2$, ArcCoth[Infinity] = 0;
ArcCsch[0] = Complex Infinity, ArcCsch[∞] = 0;
ArcSech[0] = Complex Infinity, ArcSech[∞] = I $\pi/2$.

Inverse hyperbolic functions are replaced with logarithms. The symbol # above indicates that the equivalences given above are influenced by different branchings. The branch cuts for the inverse hyperbolic functions are given below. The branch points may be found from the logarithms above or from the integral representations given below.

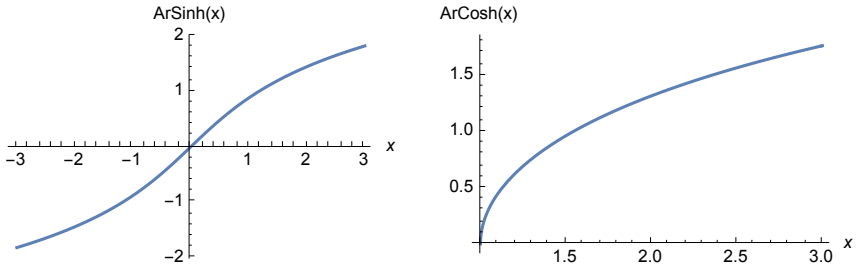
```
ArcSinh[z]          (- i ∞,  - i), (i, i ∞)
ArcCosh[z]          (-   ∞,    1)
ArcTanh[z]          (-   ∞,  - 1), (1,   ∞)
ArcCsch[z]          (- i, i )
ArcSech[z]          (-   ∞,    0), (1,   ∞)
ArcCoth[z]          (- 1,  1 )
```

$$\text{ArcSinh}(z) = \int_0^z \frac{1}{\sqrt{y^2+1}}\,dy; \qquad \text{ArcCosh}(z) = \int_0^x \frac{1}{\sqrt{y^2-1}}\,dy;$$

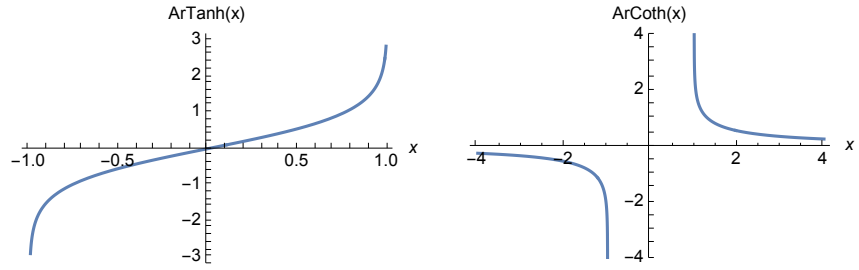
$$\text{ArcTanh}(z) = \int_0^x \frac{1}{1-y^2}\,dy = \frac{1}{2}\int_0^x \frac{1}{1+y}\,dy + \frac{1}{2}\int_0^x \frac{1}{1-y}\,dy;$$

$$\text{ArcCoth}(z) = -\int_0^x \frac{1}{y^2-1}\,dy = -\frac{1}{2}\int_0^x \frac{1}{1+y}\,dy - \frac{1}{2}\int_0^x \frac{1}{1-y}\,dy;$$

```
SetOptions[Plot, DisplayFunction -> Identity];
s = Plot[ArcSinh[x], {x, -3, 3}, AxesLabel -> {x, "ArSinh(x)"}];
c = Plot[ArcCosh[x], {x, 1, 3}, AxesLabel -> {x, "ArCosh(x)"}];
Show[GraphicsRow[{s, c}], ImageSize → 450]
```

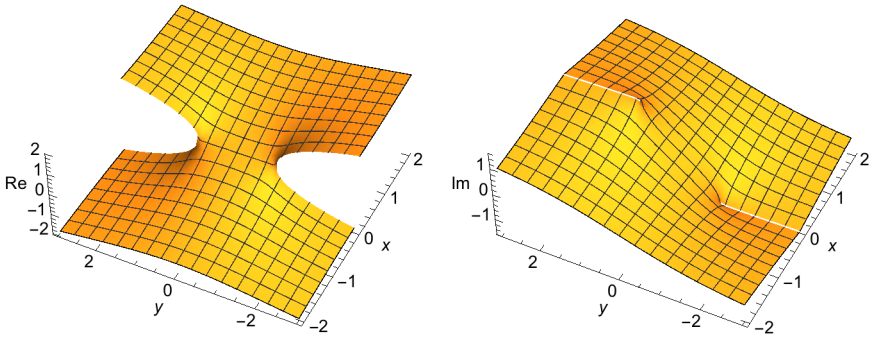


```
s = Plot[ArcTanh[x], {x, -1, 1}, AxesLabel -> {x, "ArTanh(x)"}];
c = Plot[ArcCoth[x], {x, -4, 4}, AxesLabel -> {x, "ArCoth(x)"},
    PlotRange -> {-4, 4}];
Show[GraphicsRow[{s, c}], ImageSize → 450]
```




```
SetOptions[Plot3D, PlotPoints -> 20, ViewPoint -> {-4, -2, 4},
     Boxed -> False];
r = Plot3D[ Re[ArcSinh[x + I y]], {x, -2, 2}, {y, -3, 3},
      AxesLabel -> {x, y, Re}];
i = Plot3D[ Im[ArcSinh[x + I y]], {x, -2, 2}, {y, -3, 3},
      AxesLabel -> {x, y, Im}];
Print["                                    ArSinh"]
Show[GraphicsRow[{r, i}], ImageSize → 450]
```

ArSinh




```
r = Plot3D[Re[ArcCosh[x + i y]], {x, -2, 2}, {y, -3, 3}, AxesLabel → {x, y, Re}];
i = Plot3D[Im[ArcCosh[x + i y]], {x, -2, 2}, {y, -3, 3}, AxesLabel → {x, y, Im}];
Print["                                    ArCosh"]
Show[GraphicsRow[{r, i}], ImageSize → 450]
```

ArCosh