Numerische Verfahren

Numerische Auswertung symbolischer Ergebnisse

Numerische Auswertung von Ausdrücken

```
ClearAll["Global`*"]
```

```
N[Sin[π/4]]
```

```
0.707107
```

```
N[Sin[π/4], 60]
```

```
0.707106781186547524400844362104849039284835937688474036588340
```

Ist das Argument numerischer Natur und trägt der verarbeitende Funktionsbezeichner das Attribut **NumericFunction**, so wird beim Auswerten eine numerische Näherungsroutine aufgerufen. Da die Auswertung der **FullForm** *Sin[Times[Pi,Power[4.,−1]]]* von innen nach außen stets einen numerischen Wert als Argument für den nächsten Funktionsaufruf liefert, wird letztlich ein numerischr Wert zurückgegeben.

```
Sin[π/4.]
```

```
0.707107
```

Hier die Bestätigung, dass alle beteiligten Funktionen das Attribut **NumericFunction** tragen.

```
MemberQ[Attributes[#], NumericFunction] & /@ {Sin, Times, Power}
```

```
{True, True, True}
```

```
N[√2] N[√8] - 4
```

```
8.88178 × 10^-16
```

```
% // Chop
```

```
0
```

```
√2 √8 - 4
```

```
0
```

Rechengenauigkeit

```
ClearAll["Global`*"]
```

```
$MachinePrecision
```

```
15.9546
```

Rechnungen werden standardmäßig mit Maschinengenauigkeit ausgeführt. **Precision** gibt in diesem Fall den symbolischem Wert **MachinePrecision** zurück, der numerisch (mit **N**) zu einem Zahlenwert ausgewertet wird, der auch in der Systemkonstanten **$MachinePrecision** gespeichert ist.
In früheren Versionen war dieser Wert ganzzahlig.

```
√2.
```

```
1.41421
```

```
% // Precision
```

```
MachinePrecision
```

```
% // N
```

```
15.9546
```

```
√2.`30
```

```
1.41421356237309504880168872421
```

```
% // Precision
```

```
30.301
```

$$a_1 = 0.3;\ b = \frac{1}{10};$$

$$c_1 = N\left[a_1^b,\ 40\right]$$

```
0.886568
```

```
Precision /@ {a1, b, c1}
```

```
{MachinePrecision, ∞, MachinePrecision}
```

```
0.3`25
```

```
0.3000000000000000000000000
```

$$a_2 = 0.3000000000000000000000000;$$
$$c_2 = N\left[a_2^b,\ 40\right]$$

```
0.8865681505652133345154271
```

```
u = Precision /@ {a2, b, c2}
```

```
{24.4771, ∞, 25.4771}
```

```
MachineNumberQ /@ {a1, a2, c1, c2}
```

```
{True, False, True, False}
```

```
a3 = N[π^2, 30]
b3 = N[π, 35]^2
```

```
9.869604401089358618834490999876151
```

```
a3 // InputForm
```

```
9.869604401089358618834490999876151`30.
```

```
Precision[12345.1234567890123456789]
Accuracy[12345.1234567890123456789]
```

```
23.0915
```

```
19.
```

```
c3 = a3 - b3
```

```
0. × 10^-30
```

```
Precision /@ {a3, b3, c3}
```

```
{30., 34.699, 0.}
```

Ungenaue Eingaben an wenig signifikanter Stelle haben auf die Ergebnisgenauigkeit geringen Einfluss, so dass sich in einem solchen Fall die Genauigkeit auch vergrößern kann.

```
10^5 + c2
```

```
100000.88656815056521333451542 71
```

```
% // Precision
```

```
30.5294
```

▪ Näherungswerte in exakte Werte verwandeln

```
ClearAll["Global`*"]
```

```
u = 5./17
Rationalize[u]
```

```
0.294118
```

$$\frac{5}{17}$$

Diese beiden Rechnungen sehen zwar gleich aus, sind es aber nicht!

```
u1 = 3.14159
Rationalize[u1]
```

```
3.14159
```

$$\frac{314\,159}{100\,000}$$

```
u2 = N[π]
Rationalize[u2]
```

```
3.14159
```

```
3.14159
```

```
InputForm /@ {u1, u2}
```

```
{3.14159, 3.141592653589793}
```

Die Umwandlung kann man erzwingen, indem als Präzision 0 angegeben wird.

```
Rationalize[u2, 0.007]
% - N[π, 20]
```

$$\frac{22}{7}$$

```
0.0012644892673496187
```

```
Rationalize[u2, 0]
% - N[π, 25]
```

$$\frac{245\,850\,922}{78\,256\,779}$$

$$-7.8179366 \times 10^{-17}$$

```
Rationalize[N[π, 40], 0]
% - N[π, 50]
```

$$\frac{262\,452\,630\,335\,382\,199\,398}{83\,541\,266\,890\,691\,994\,833}$$

$$1.011863227 \times 10^{-40}$$

■ Gleichungen numerisch lösen

■ Solve und NSolve

```
ClearAll["Global`*"]
```

```
Solve[x^2 + 2 x + 3 == 0, x]
```

$$\left\{\left\{x \to -1 - i\sqrt{2}\right\}, \left\{x \to -1 + i\sqrt{2}\right\}\right\}$$

```
% // N
```

```
{{x → -1. - 1.41421 i}, {x → -1. + 1.41421 i}}
```

```
Solve[x^2 + 2 x + 3. == 0, x]
```

```
{{x → -1. - 1.41421 i}, {x → -1. + 1.41421 i}}
```

```
NSolve[x^2 + 2 x + 3 == 0, x]
```

```
{{x → -1. - 1.41421 i}, {x → -1. + 1.41421 i}}
```

```
NSolve[x^2 + 2 x + 3 == 0, x, 30]
```

```
{{x → -1.00000000000000000000000000000 - 1.41421356237309504880168872421 i},
 {x → -1.00000000000000000000000000000 + 1.41421356237309504880168872421 i}}
```

```
sys = {x^4 + y^3 == 1, y^4 + x^3 == 1};
(sol = (esol = Solve[sys, {x, y}]) // N;) // Timing
```

```
{16.39, Null}
```

```
sol // Length
sol // Union // Length
```

```
16
```

```
12
```

```
esol // ByteCount
```

```
291 208
```

```
red = Reduce[sys, {x, y}, Backsubstitution → True]
```

```
(x == 0 && y == 1) || (x == 1 && y == 0) ||
 (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 1], {0, 1, 0, 0}] &&
   y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 1], {0, 1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 2], {0, 1, 0, 0}] &&
   y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 2], {0, 1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 3], {0, 1, 0, 0}] &&
   y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 3], {0, 1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 4], {0, 1, 0, 0}] &&
   y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 4], {0, 1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 1],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 1], {-2, -2, -1, -1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 2],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 2], {-2, -2, -1, -1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 3],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 3], {-2, -2, -1, -1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 4],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 4], {-2, -2, -1, -1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 5],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 5], {-2, -2, -1, -1, 0, 0}]) ||
 (x == AlgebraicNumber[Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 6],
     {0, 1, 0, 0, 0, 0}] && y == AlgebraicNumber[
     Root[3 + 5 #1 + 6 #1^2 + 6 #1^3 + 4 #1^4 + 2 #1^5 + #1^6 &, 6], {-2, -2, -1, -1, 0, 0}])
```

```
sol = {red // N // ToRules}
```

```
{{x → 0., y → 1.}, {x → 1., y → 0.}, {x → -1.38028, y → -1.38028},
 {x → 0.819173, y → 0.819173}, {x → -0.219447 - 0.914474 i, y → -0.219447 - 0.914474 i},
 {x → -0.219447 + 0.914474 i, y → -0.219447 + 0.914474 i},
 {x → -1.04414 - 0.390425 i, y → -0.188646 + 1.18298 i},
 {x → -1.04414 + 0.390425 i, y → -0.188646 - 1.18298 i},
 {x → -0.188646 - 1.18298 i, y → -1.04414 + 0.390425 i},
 {x → -0.188646 + 1.18298 i, y → -1.04414 - 0.390425 i},
 {x → 0.232786 - 1.27599 i, y → 0.232786 + 1.27599 i},
 {x → 0.232786 + 1.27599 i, y → 0.232786 - 1.27599 i}}
```

```
Eliminate[sys, y]
```

$$3 x^3 + 2 x^4 + x^5 - 3 x^6 - 7 x^7 - 5 x^8 - 2 x^9 + x^{10} + 4 x^{11} + 3 x^{12} + 2 x^{13} + x^{14} == 0$$

```
f = FactorList[First[%]]
```

$$\{\{1, 1\}, \{-1 + x, 1\}, \{x, 3\}, \{-1 + x^3 + x^4, 1\}, \{3 + 5 x + 6 x^2 + 6 x^3 + 4 x^4 + 2 x^5 + x^6, 1\}\}$$

```
GroebnerBasis[Prepend[sys, f[[5, 1]] == 0 ], {y, x}]
```

$$\{3 + 5 x + 6 x^2 + 6 x^3 + 4 x^4 + 2 x^5 + x^6, 2 + 2 x + x^2 + x^3 + y\}$$

```
sys /. sol
```

```
{{True, True}, {True, True}, {True, True}, {True, True},
 {False, False}, {False, False}, {False, False}, {True, False}, {True, False},
 {False, False}, {False, False}, {False, False}, {False, False}}
```

```
esys = First /@ sys
```

$$\{x^4 + y^3, x^3 + y^4\}$$

```
esys /. sol
```

$$\{\{1., 1.\}, \{1., 1.\}, \{1., 1.\}, \{1., 1.\}, \{1. - 2.22045 \times 10^{-16} i, 1. - 2.22045 \times 10^{-16} i\},$$
$$\{1. + 2.22045 \times 10^{-16} i, 1. + 2.22045 \times 10^{-16} i\}, \{1. + 0. i, 1. + 4.44089 \times 10^{-16} i\},$$
$$\{1. + 0. i, 1. - 4.44089 \times 10^{-16} i\}, \{1. - 2.22045 \times 10^{-16} i, 1. + 2.22045 \times 10^{-16} i\},$$
$$\{1. + 2.22045 \times 10^{-16} i, 1. - 2.22045 \times 10^{-16} i\},$$
$$\{1. - 2.22045 \times 10^{-16} i, 1. + 1.11022 \times 10^{-15} i\}, \{1. + 2.22045 \times 10^{-16} i, 1. - 1.11022 \times 10^{-15} i\}\}$$

NSolve liefert dasselbe Ergebnis wie **Solve[sys,vars]//N**, jedoch deutlich schneller und in einer anderen Reihenfolge.

```
(nsol = NSolve[sys, {x, y}];) // Timing
```

```
{0.09, Null}
```

```
nsol // Length
nsol // Union // Length
```

```
16
```

```
12
```

```
sys /. nsol
```

```
{{False, False}, {False, False}, {False, False}, {False, False}, {False, False},
 {False, False}, {True, True}, {True, True}, {True, True}, {True, True}, {True, True},
 {True, True}, {False, False}, {False, False}, {False, False}, {True, True}}
```

```
esys /. nsol
```

$$\{\{1. - 3.55271\times10^{-15}\,i,\ 1. + 5.55112\times10^{-15}\,i\},$$
$$\{1. + 3.55271\times10^{-15}\,i,\ 1. - 5.55112\times10^{-15}\,i\},\ \{1. + 5.77316\times10^{-15}\,i,\ 1. - 4.21885\times10^{-15}\,i\},$$
$$\{1. - 5.77316\times10^{-15}\,i,\ 1. + 4.21885\times10^{-15}\,i\},\ \{1. - 1.39888\times10^{-14}\,i,\ 1. + 1.9984\times10^{-15}\,i\},$$
$$\{1. + 1.39888\times10^{-14}\,i,\ 1. - 1.9984\times10^{-15}\,i\},\ \{1., 1.\},\ \{1., 1.\},\ \{1., 1.\},\ \{1., 1.\},$$
$$\{1., 1.\},\ \{1., 1.\},\ \{1., 1.\},\ \{1. + 5.77316\times10^{-15}\,i,\ 1. + 6.88338\times10^{-15}\,i\},$$
$$\{1. - 5.77316\times10^{-15}\,i,\ 1. - 6.88338\times10^{-15}\,i\},\ \{1., 1.\}\}$$

```
% // Chop
```

```
{{1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.},
 {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}, {1., 1.}}
```

```
red = Reduce[sys, {x, y}, Reals]
```

```
(x == 0 && y == 1) || (x == 1 && y == 0) ||
  (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 1], {0, 1, 0, 0}] &&
    y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 1], {0, 1, 0, 0}]) ||
  (x == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 2], {0, 1, 0, 0}] &&
    y == AlgebraicNumber[Root[-1 + #1^3 + #1^4 &, 2], {0, 1, 0, 0}])
```

```
{red // N // ToRules}
```

```
{{x → 0., y → 1.}, {x → 1., y → 0.},
 {x → -1.38028, y → -1.38028}, {x → 0.819173, y → 0.819173}}
```

```
Select[nsol // Union, (x /. #) ∈ Reals &]
```

```
{{x → -1.38028, y → -1.38028}, {x → 0., y → 1.},
 {x → 0.819173, y → 0.819173}, {x → 1., y → 0.}}
```

■ Nullstellenbestimmung mit FindRoot

Essentiell transzendente Gleichungen lassen sich mit den Kommandos **Solve**, **NSolve** oder **Reduce** nicht bearbeiten.

```
g1 = Log[1 + x] + Log[3 + x] + x;
```

Solve[g1 == 5, x]

```
Solve::tdep :
 The equations appear to involve the variables to be solved
   for in an essentially non-algebraic way.
```

```
Solve[x + Log[1 + x] + Log[3 + x] == 5, x]
```

Reduce[g1 == 5, x]

```
Reduce::nsmet :
 This system cannot be solved with the methods available to Reduce.
```

```
Reduce[x + Log[1 + x] + Log[3 + x] == 5, x]
```

NSolve[g1 == 5, x]

```
Solve::tdep :
 The equations appear to involve the variables to be solved
   for in an essentially non-algebraic way.
```

```
NSolve[x + Log[1 + x] + Log[3 + x] == 5, x]
```

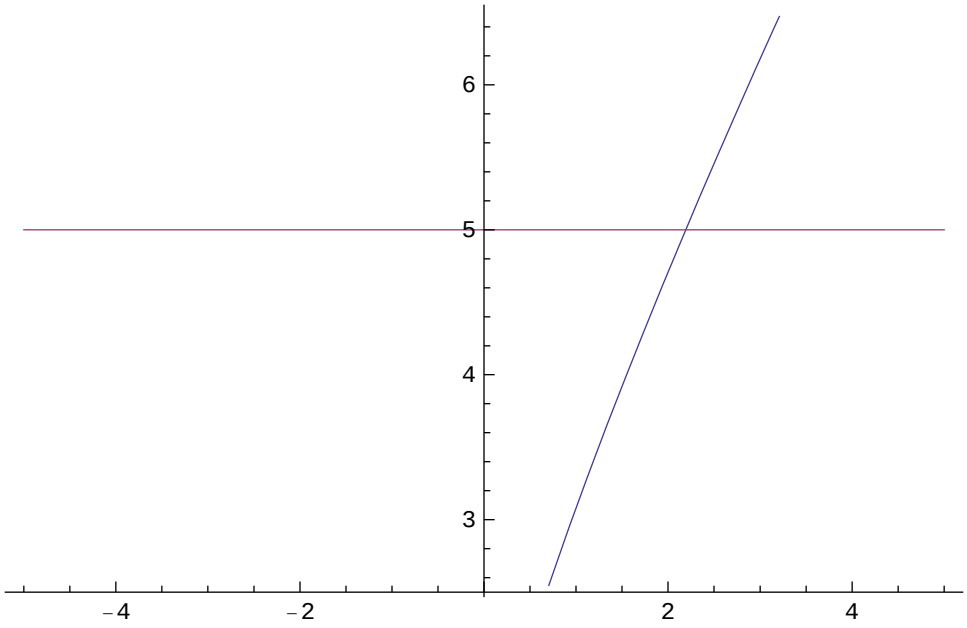
FindInstance[g1 == 5, x]

```
FindInstance::nsmet :
 The methods available to FindInstance are insufficient to
   find the requested instances or prove they do not exist.
```

```
FindInstance[x + Log[1 + x] + Log[3 + x] == 5, x]
```

Der Plot zeigt, dass eine relle Lösung von g1== 5in der Nähe von x== 2liegt.

Plot[{g1, 5}, {x, -5, 5}]



Diese kann mit **FindRoot** bestimmt werden.

```
FindRoot[g1 == 5, {x, 2}]
```

```
{x → 2.19215}
```

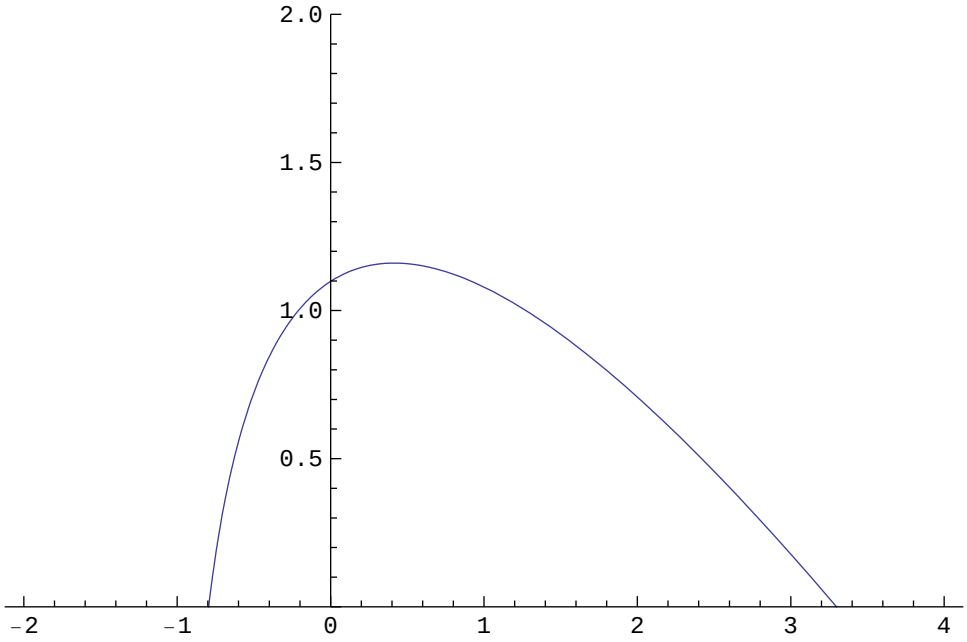
```
g1 /. %
```

```
5.
```

```
g2 = Log[1 + x] + Log[3 + x] - x
```

```
-x + Log[1 + x] + Log[3 + x]
```

```
Plot[g2, {x, -2, 4}, PlotRange -> {0, 2}]
```




```
FindRoot[g2 == 2, {x, -2}]
```

```
{x → -0.209427 + 1.66589 i}
```

```
FindRoot[g2 == 1, {x, -2}]
```

```
{x → 1.27428 - 2.32469 × 10^-21 i}
```

```
FindRoot[g2 == 1, {x, 0}]
```

```
{x → -0.210511}
```

```
FindRoot[g2 == 2, {x, 0}]
```

```
FindRoot::nlnum :
 The function value {Indeterminate} is not a list of numbers
   with dimensions {1} at {x} = {-1.}.
```

```
{x → 0.419252}
```

```
g2 /. %
```

```
1.1603
```

```
FindRoot[g2 == 2, {x, 0 + 0.001 I}]
```

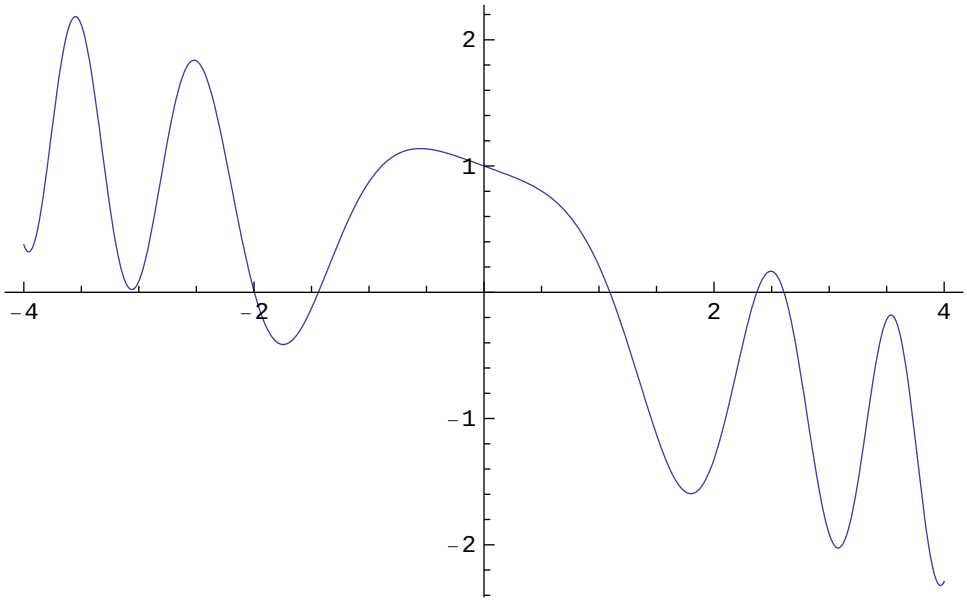
```
{x → -0.209427 + 1.66589 i}
```

```
sys = {x^2 + y^2 == 10, x^y == 2};
FindRoot[sys , {x, 1}, {y, 1}]
```

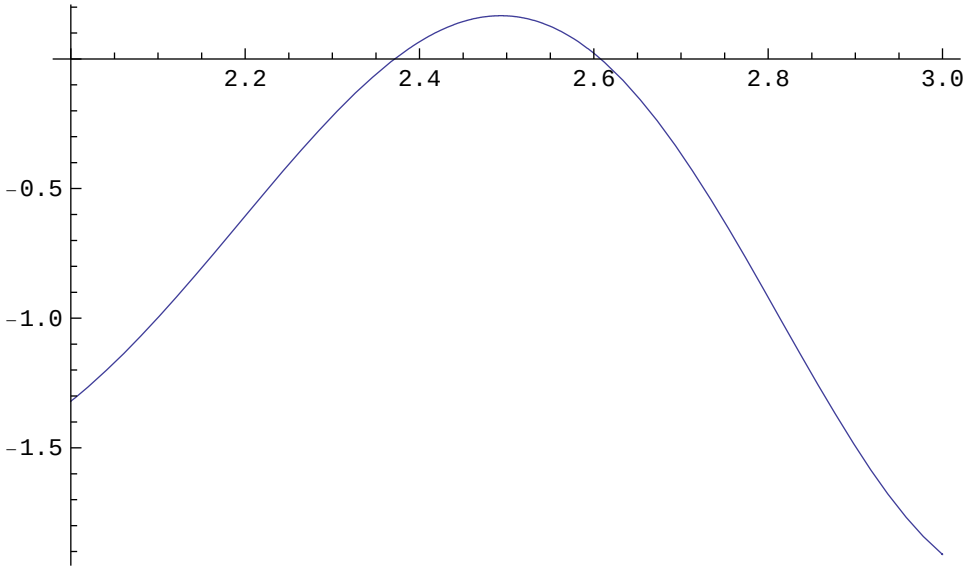
```
{x → 1.27043, y → 2.89586}
```

$$f := \text{Cos}\left[x^2\right] - \frac{x}{3};$$

```
Plot[f, {x, -4, 4}]
```



```
Plot[f, {x, 2, 3}]
```



Nullstellenbestimmung mit dem Newtonverfahren.

```
FindRoot[f == 0, {x, #}] & /@ {2.4, 2.6}
```

```
{{x → 2.3715}, {x → 2.60774}}
```

Nullstellenbestimmung mit dem Sekantenverfahren. Hierfür muss *f* nicht symbolisch differenzierbar sein.

```
FindRoot[f, {x, 2.4, 3}]
```

```
{x → 2.60774}
```

Problem: Die Nullstelle kann in einem Extremwert liegen.

```
FindRoot[f^2 == 0, {x, 3}]
```

```
{x → 2.60774}
```

Analyse in der Nähe von x=3.5: Suche mit Startwert und Intervallbegrenzung

```
FindRoot[f, {x, 3.5, 3.3, 3.7}]
```

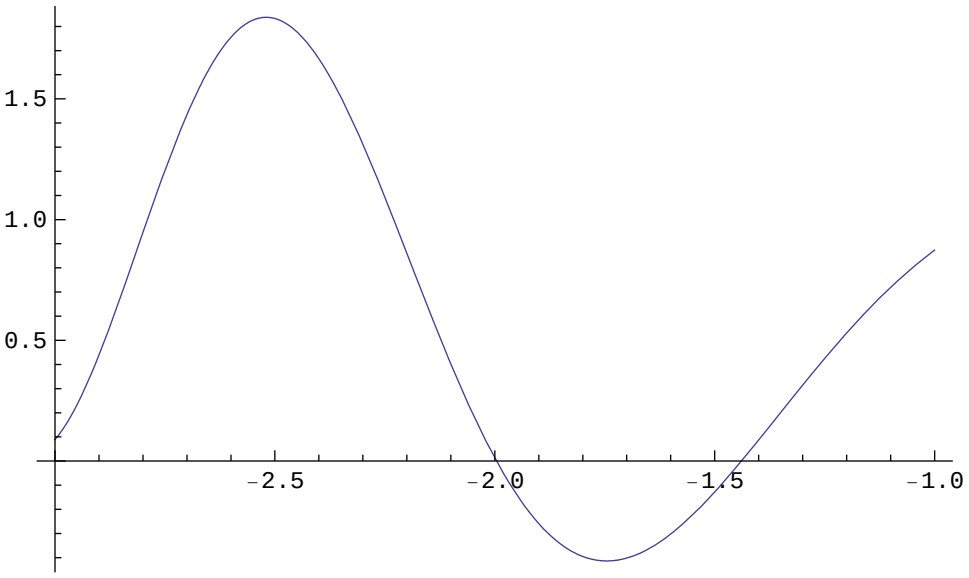
```
FindRoot::lstol :
 The line search decreased the step size to within tolerance
   specified by AccuracyGoal and PrecisionGoal but was
   unable to find a sufficient decrease in the merit
   function.  You may need more than MachinePrecision
   digits of working precision to meet these tolerances.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

```
{x → 3.53826}
```

```
FindMaximum[f, {x, 3.5}]
```

```
{-0.180528, {x → 3.53826}}
```

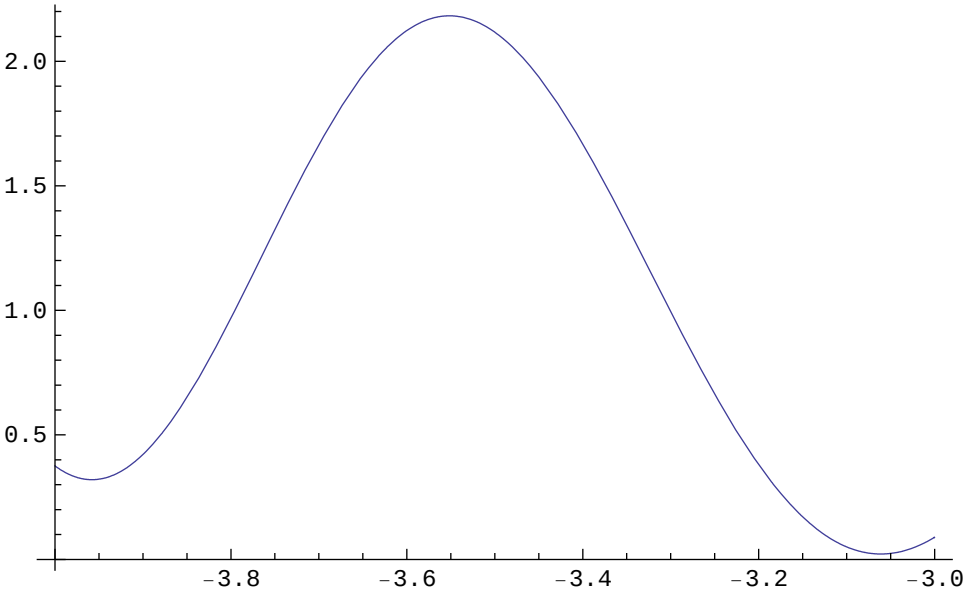
```
Plot[f, {x, -3, -1}]
```



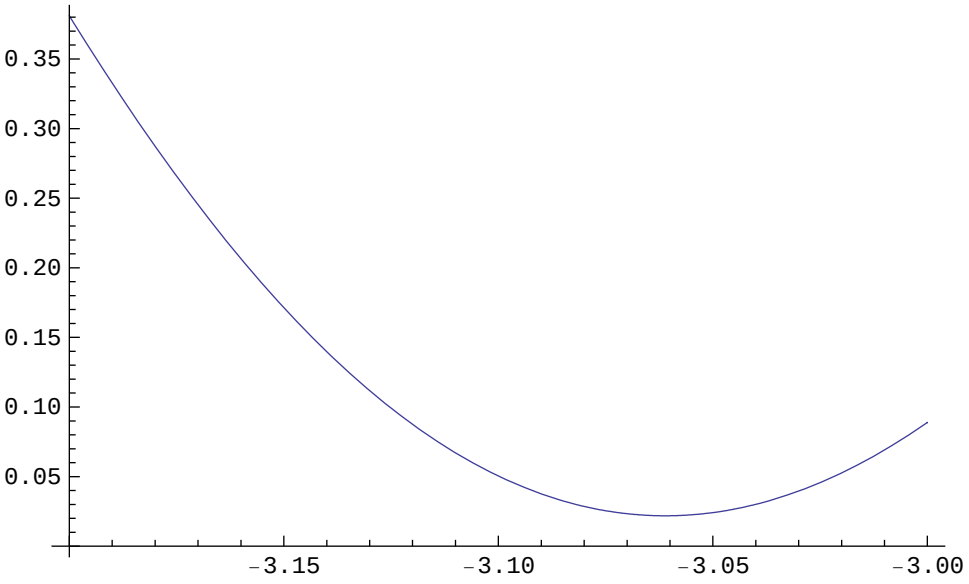

```
FindRoot[f == 0, {x, #}] & /@ {-2, -1.5}
```

```
{{x → -1.9961}, {x → -1.43914}}
```

```
Plot[f, {x, -4, -3}]
```




```
Plot[f, {x, -3.2, -3}]
```




```
FindRoot[f, {x, -3.0}]
```

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

```
{x → -3.0611}
```

```
FindRoot[f, {x, -3.1, -3.0}]
```

```
FindRoot::cvmit : Failed to converge to the
   requested accuracy or precision within 100 iterations.
```

```
{x → -3.10075}
```

```
FindMinimum[f, {x, -3.0}]
```

```
{0.0218484, {x → -3.0611}}
```

■ Numerische Integration

```
ClearAll["Global‘*"]
```

Numerische Bestimmung definiter Integrale ist eine Möglichkeit, wenn keine Stammfunktion in Termini anderer Funktionen berechnet werden kann.

```
Integrate[Sin[x^2 + Cos[x]], {x, 0, 2}]
```

$$\int_0^2 \text{Sin}\left[x^2 + \text{Cos}[x]\right] dx$$

```
% // N
```

```
1.48203
```

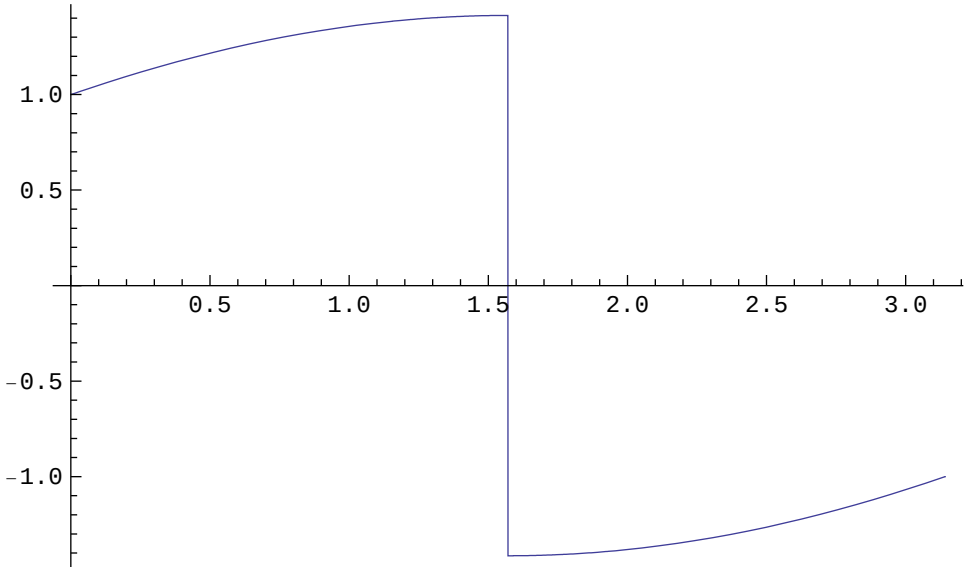
Das Ergebnis von **Integrate[f,{x,a,b}]//N** und **NIntegrate[f,{x,a,b}]** ist dasselbe.
NIntegrate geht in einem solchen Fall aber schneller, weil der Aufwand zur Feststellung, dass es keine Stammfunktion in geschlossner Form gibt, nicht anfällt.

```
NIntegrate[Sin[x^2 + Cos[x]], {x, 0, 2}]
```

```
1.48203
```

Integration einer Funktion mit Sprungstelle im Integrationsintervall.

```
f = Cos[x] / Sqrt[1 - Sin[x]];
Plot[f, {x, 0, π}]
```




```
sf[x_] = ∫f dx // Simplify
```

$$-2\sqrt{1-\sin[x]}$$

```
∫_0^(π/2) f dx
```

2

```
NIntegrate[f, {x, 0, π/2}]
```

2.

```
∫_0^(3π/4) f dx
```

$$2-\sqrt{4-2\sqrt{2}}$$

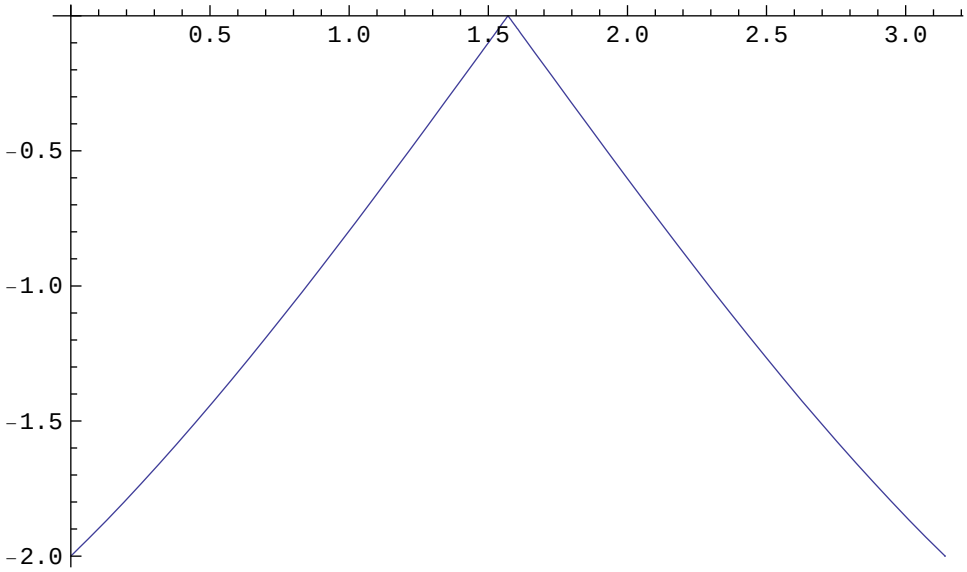
```
% // N
```

0.917608

```
sf[3π/4] - sf[0] // Simplify
```

$$2-\sqrt{4-2\sqrt{2}}$$

```
Plot[sf[x], {x, 0, π}]
```




```
NIntegrate[f, {x, 0, 3π/4}]
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.

NIntegrate::ncvb :
 NIntegrate failed to converge to prescribed accuracy
   after 9 recursive bisections in x near {x} =
   {1.5693}. NIntegrate obtained 0.9170805749970934‘ and
   0.0008453999243025678‘ for the integral and error estimates.

0.917081
```

```
NIntegrate[f, {x, 0, 3π/4}, MaxRecursion → 20]
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.

0.917607
```

```
NIntegrate[f, {x, 0, 3 π/4}, MaxRecursion → 40, WorkingPrecision → 20]
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.
```

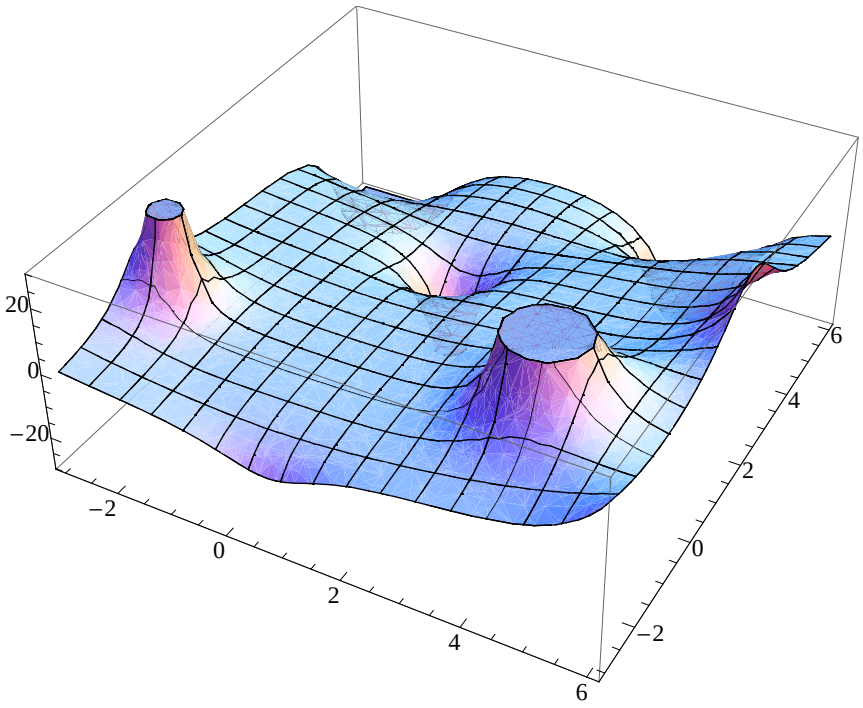
```
0.91760779972387813954
```

```
NIntegrate[f, {x, 0, π/2, 3 π/4}]
```

```
0.917608
```

Beispiel: Mehrdimensionales Integral

```
Clear[f]
f := (x^2 - y^2) / (Cos[x + y] + Sin[x - y] + 21/10);
Plot3D[f, {x, -3, 6}, {y, -3, 6}]
```



Integrate kommt hier in vernünftiger Zeit gar nicht durch. Das Kommando ist deshalb auskommentiert.

```
(* (i=Integrate[f,{x,-3,6},{y,-3,6}];)//Timing *)
```

```
NIntegrate[f, {x, -3, 6}, {y, -3, 6}] // Timing
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.
```

```
{1.38, -109.52}
```

```
NIntegrate[f, {x, -3, 6}, {y, -3, 6},
  WorkingPrecision → 15] // Timing
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.
```

```
{12.66, -109.519572644826}
```

```
%[[2]] // Precision
```

```
15.
```

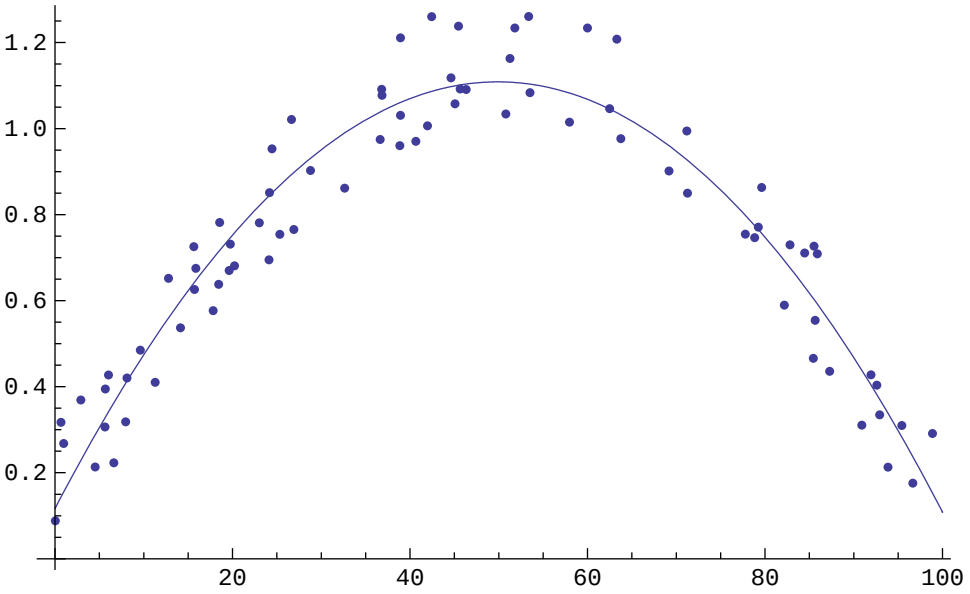
Datenanalyse

Fit mit vorgegebenen Basisfunktionen

```
ClearAll["Global`*"]
list1 = Table[Random[Real, 100], {80}];
data = {#, Sin[# / 100 Pi] + Random[Real, 0.3]} & /@ list1;
gr0 = ListPlot[data, PlotStyle -> PointSize[0.01]];
```

```
f1 = Fit[data, {1, x, x^2}, x]
gr1 = Plot[f1, {x, 0, 100}];
Show[{gr0, gr1}]
```

$0.116791 + 0.0397563\,x - 0.000398392\,x^2$



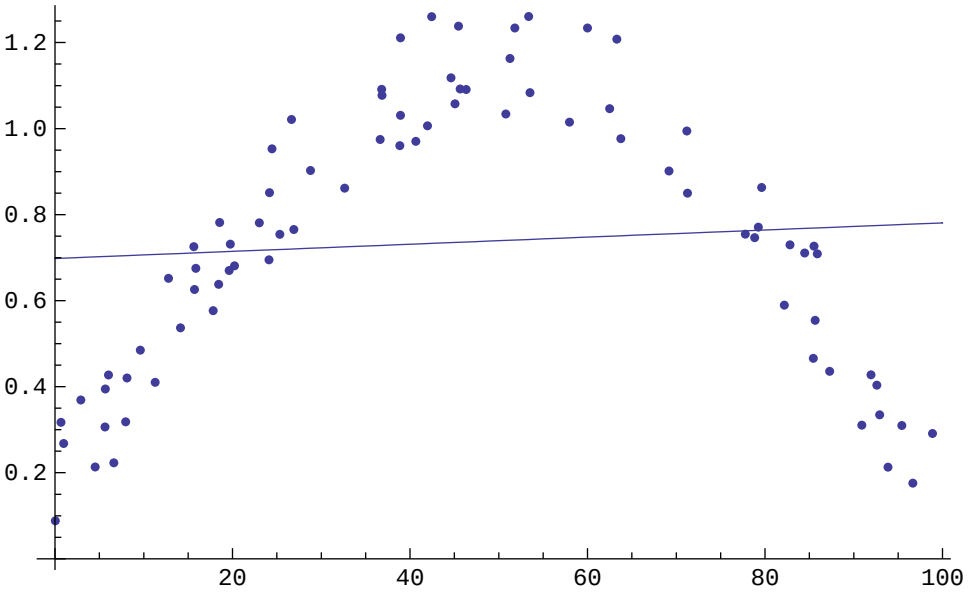
Dieselbe Aufgabe mit **FindFit**.

```
FindFit[data, f1a = a x^2 + b x + c, {a, b, c}, x];
f1a /. %
```

$0.116791 + 0.0397563\,x - 0.000398392\,x^2$

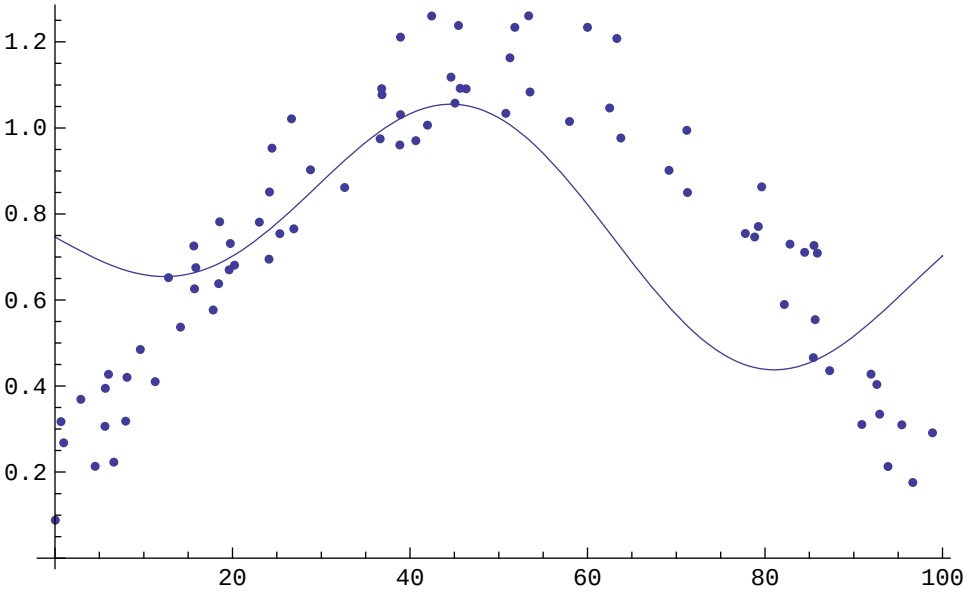
```
f2 = Fit[data, {1, x}, x]
gr2 = Plot[f2, {x, 0, 100}];
Show[{gr0, gr2}]
```

$$0.698034 + 0.000827811\,x$$




```
f3 = Fit[data, {1, Sin[x/10], Sin[x/20]}, x]
gr3 = Plot[f3, {x, 0, 100}];
Show[{gr0, gr3}]
```

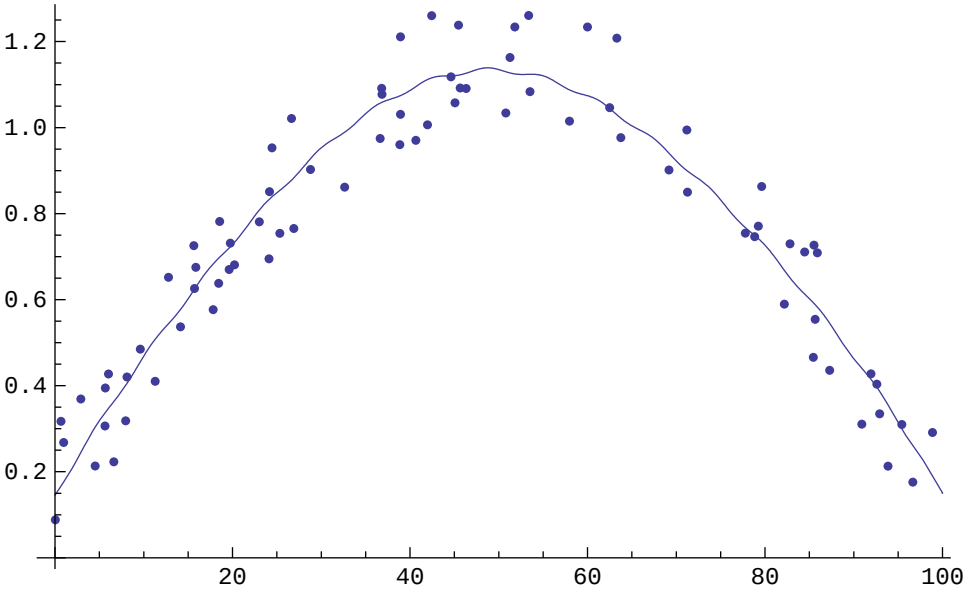
$$0.74644 + 0.154888\,\text{Sin}\left[\frac{x}{20}\right] - 0.192338\,\text{Sin}\left[\frac{x}{10}\right]$$



Regressionsanalyse (Regress)

```
f4 = Fit[data, {1, x, x^2, Sin[x], Sin[x/10]}, x]
gr4 = Plot[f4, {x, 0, 100}];
Show[{gr0, gr4}]
```

$$0.145863 + 0.0384732\,x - 0.000386209\,x^2 - 0.0307277\,\sin\left[\frac{x}{10}\right] - 0.00510776\,\sin[x]$$



Informationen über die Güte der Approximation liefert das Kommando **Regress** aus dem Paket **LinearRegression**. Standardmäßig wird die Parametertabelle ausgegeben, es gibt aber auch andere interessante Optionen.

```
Needs["LinearRegression`"]
Regress[data, {1, x, x^2, Sin[x], Sin[x/10]}, x]
```

{ParameterTable →

	Estimate	SE	TStat	PValue
1	0.145863	0.0320593	4.54977	0.000020298
x	0.0384732	0.00159762	24.0816	0.
x^2	−0.000386209	0.0000158051	−24.4357	1.87863×10^{-37},
Sin[x]	−0.00510776	0.015009	−0.340313	0.734573
$\sin\left[\frac{x}{10}\right]$	−0.0307277	0.0174871	−1.75716	0.0829715

RSquared → 0.918915, AdjustedRSquared → 0.91459, EstimatedVariance → 0.00856012,

ANOVATable →

	DF	SumOfSq	MeanSq	FRatio	PValue
Model	4	7.27571	1.81893	212.488	0.
Error	75	0.642009	0.00856012		
Total	79	7.91772			

}

```
RegressionReportValues[Regress]
```

```
{AdjustedRSquared, ANOVATable, BestFit, BestFitParameters,
 BestFitParametersDelta, CatcherMatrix, CoefficientOfVariation,
 CookD, CorrelationMatrix, CovarianceMatrix, CovarianceMatrixDetRatio,
 DurbinWatsonD, EigenstructureTable, EstimatedVariance, FitResiduals,
 HatDiagonal, JackknifedVariance, MeanPredictionCITable, ParameterCITable,
 ParameterConfidenceRegion, ParameterTable, PartialSumOfSquares, PredictedResponse,
 PredictedResponseDelta, RSquared, SequentialSumOfSquares, SinglePredictionCITable,
 StandardizedResiduals, StudentizedResiduals, SummaryReport, VarianceInflation}
```

Nichtlineare Regression (FindFit)

```
ClearAll["Global`*"]
data = Table[{x, 3 Sin[5 x] + Random[]}, {x, 0, 2 π, 0.01 π}];
f = a Sin[b x];
```

Nichtlineare Fits sind sehr von den Anfangswerten für die gewählten Parameter abhängig. Defaultstartwert für alle Parameter ist 1, was hier weit von den wirklichen Werten entfernt ist.

```
f1 = f /. FindFit[data, f, {a, b}, x]
```

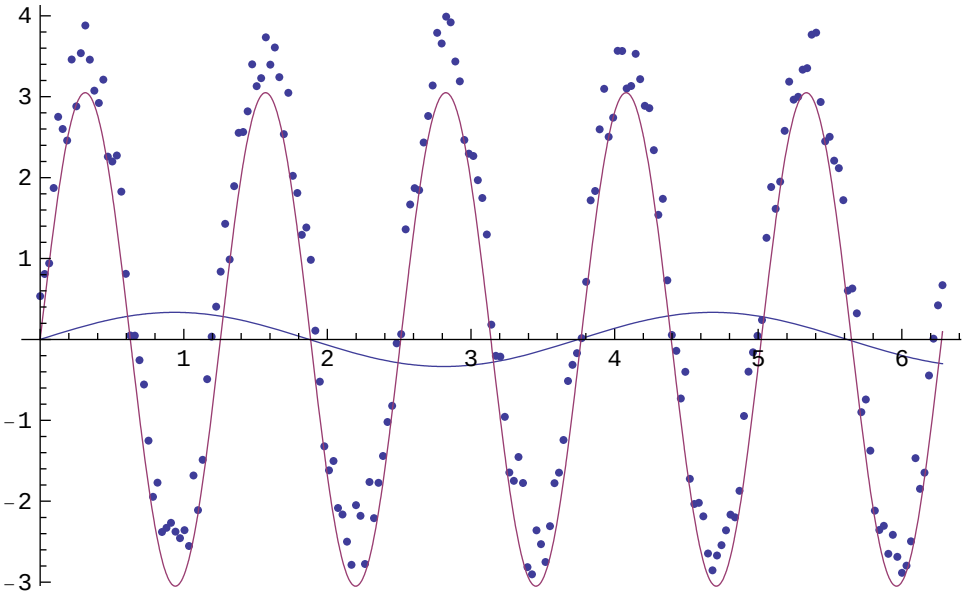
```
0.334611 Sin[1.67702 x]
```

Mit besseren Startwerten erhalten wir, trotz des hohen Rauschanteils in den Daten, einen sehr guten Fit.

```
f2 = f /. FindFit[data, f, {{a, 5}, {b, 5.2}}, x]
```

```
3.0494 Sin[5.00504 x]
```

```
Show[ ListPlot[data], Plot[{f1, f2}, {x, 0, 2 Pi}]]
```




```
data1 = Table[{x, 20 Sin[3 x] + 10 Exp[0.2 x] + 10 Random[]}, {x, 0, 2 π, 0.01 π}];
f = a Sin[b x] + c Exp[d x];
```

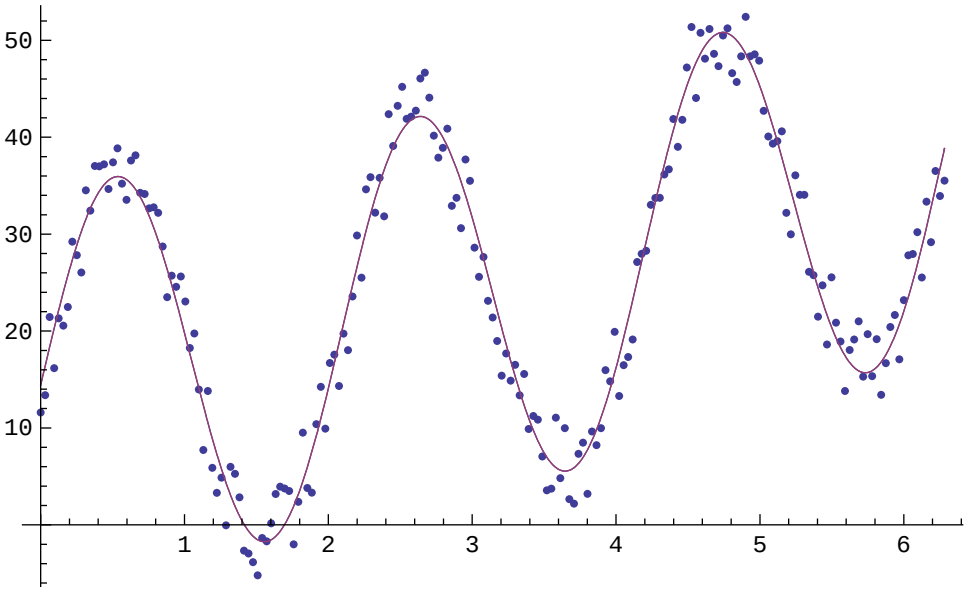
```
f1 = f /. FindFit[data1, f, {a, b, c, d}, x]
```

$14.434\, e^{0.15868\, x} + 20.2378 \sin[2.99776\, x]$

```
f2 = f /. FindFit[data1, f, {{a, 12.}, {b, 2.8}, {c, 7.}, {d, 0.3}}, x]
```

$14.434\, e^{0.15868\, x} + 20.2378 \sin[2.99776\, x]$

```
Show[ ListPlot[data1], Plot[{f1, f2}, {x, 0, 2 Pi}]]
```



■ Zur Genauigkeit numerischer Rechnungen

▪ PrecisionGoal, AccuracyGoal, WorkingPrecision

Normalerweise rechnet *Mathematica* mit Maschinengenauigkeit und verwendet dazu die schnelle interne Hardware−Arithmetik.

```
ClearAll["Global`*"]

sys = {x^2 + y^2 == 10, x^y == 2};
sol = FindRoot[sys , {x, 1}, {y, 1}]
sol // Precision
% // N
sol // Accuracy
```

$\{x \to 1.27043, y \to 2.89586\}$

```
MachinePrecision
```

15.9546

15.4928

```
sol = FindRoot[sys , {x, 1}, {y, 1}, PrecisionGoal → 30]
% // Precision
```

```
{x → 1.27043, y → 2.89586}
```

```
MachinePrecision
```

```
sol = FindRoot[sys , {x, 1}, {y, 1}, WorkingPrecision → 20]
sol // Precision
sol // Accuracy
```

```
{x → 1.2704333456136053792, y → 2.8958589596789795152}
```

```
20.
```

```
19.5382
```

```
sol = FindRoot[sys , {x, 1}, {y, 1},
    WorkingPrecision → 20, AccuracyGoal → 4]
sol // Precision
sol // Accuracy
```

```
{x → 1.2704333456143941074, y → 2.8958589596788429836}
```

```
20.
```

```
19.5382
```

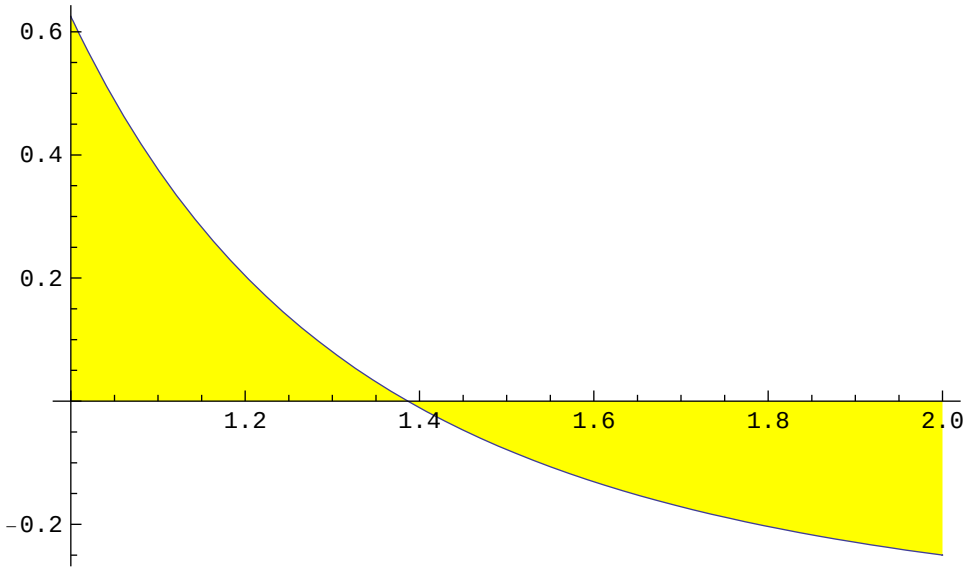
```
(First /@ sys) /. sol
% // Accuracy
```

```
{10.000000000001213301, 2.0000000000035303354}
```

```
18.699
```

Bei schlecht konditionierten numerischen Problemen muss oft eine andere als die Standardeinstellung zwischen den verschiedenen Präzisionsparametern verwendet werden.
Im folgenden Beispiel lautet das exakte Ergebnis Null, so dass ein Näherungsverfahren mit Stellenauslöschung zu "kämpfen" hat. Dies ist klarerweise ein Wechselspiel zwischen **AccuracyGoal** − wieviele Stellen nach dem Komma garantiert werden sollen − und **WorkingPrecision** − auf wieviele Stellen genau dafür zu rechnen ist. **PrecisionGoal** − wieviele gültige Ziffern zu produzieren sind − kann dagegen keine Role spielen, da das (erwartete) Ergebnis ja gleich Null ist und folglich überhaupt keine gültigen Ziffern hat.

```
f = 1/x^3 - 3/8;
Plot[f, {x, 1, 2}, Filling -> Axis, FillingStyle -> Yellow]
```




```
∫_1^2 f dx

0
```

In der Standardeinstellung kommt das richtige Ergebnis heraus, aber die nötige Konvergenzgüte wird nicht erreicht.

```
NIntegrate[f, {x, 1, 2}]
% // Accuracy
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.

NIntegrate::ncvb :
 NIntegrate failed to converge to prescribed accuracy
   after 9 recursive bisections in x near {x} =
   {1.08009}. NIntegrate obtained -1.30104×10^-17 and
   1.7550522667109103'*^-17 for the integral and error estimates.

0.×10^-17

16.7557
```

Wird **AccuracyGoal** heruntergesetzt, so verschwinden die Warnungen. Arbeits− und Ergebnisgenauigkeit sind im Einklang; das Ergebnis hat geringere **Accuracy**.

```
NIntegrate[f, {x, 1, 2}, AccuracyGoal → 4]
% // Accuracy
```

```
0. × 10^-7
```

```
6.12363
```

Dasselbe mit **PrecisionGoal** hat dagegen keinen Einfluss auf das Verhalten, selbst wenn die **WorkingPrecision** hochgesetzt wird.

```
NIntegrate[f, {x, 1, 2}, PrecisionGoal → 4]
% // Accuracy
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.
```

```
NIntegrate::ncvb :
 NIntegrate failed to converge to prescribed accuracy
   after 9 recursive bisections in x near {x} =
   {1.08009}. NIntegrate obtained -1.30104 × 10^-17 and
   1.7550522667109103'*^-17 for the integral and error estimates.
```

```
0. × 10^-17
```

```
16.7557
```

```
NIntegrate[f, {x, 1, 2}, PrecisionGoal → 4, WorkingPrecision → 80]
% // Accuracy
```

```
NIntegrate::slwcon :
 Numerical integration converging too slowly; suspect one of the
   following: singularity, value of the integration is 0,
   highly oscillatory integrand, or WorkingPrecision too small.
```

```
NIntegrate::ncvb :
 NIntegrate failed to converge to prescribed accuracy
   after 9 recursive bisections in x near {x}
   = { <<136>> }. NIntegrate obtained <<141>> and
   <<141>> for the integral and error estimates.
```

```
2.112447061418290903415270254105907086941091786966236834357606441968485204946483 1×
 10^-87
```

```
166.675
```

AccuracyGoal schon.

```
NIntegrate[f, {x, 1, 2}, AccuracyGoal → 30,  WorkingPrecision → 40]
% // Accuracy
```

```
0. × 10^-31
```

```
30.584
```

ExtraPrecision

Die von *Mathematica* verwendeten Algorithmen zur numerischen Auswertung
mathematischer Funktionen sind so ausgelegt, dass sie die Genauigkeit so weit
als möglich erhalten. Dazu wird intern an vielen Stellen mit höherer Präzision gerechnet.

```
N[Sin[10^10], 30]
% // Precision
```

```
-0.487506025087510691527794294348

30.
```

Diese intern vorgesehen Präzisionserhöhung ist durch den in der Systemvariablen **$MaxExtraPrecision** gespeicherten Wert begrenzt, was an manchen Stellen nicht ausreicht.

```
N[Sin[10^100], 30]
% // Precision
```

```
N::meprec : Internal precision limit
   $MaxExtraPrecision = 50.‘ reached while evaluating
   Sin[100000000000000000000000000000000000000000000000000000000000
     00000000000000000000000000000000000000000].

0.

0.
```

Verwenden Sie **Block**, um Systemvariablen intermediär zu verändern, ohne dass dies globale Auswirkungen hat.

```
Block[{$MaxExtraPrecision = 200}, N[Sin[10^100], 30]]
% // Precision
```

```
-0.372376123661276688262086695553

30.
```

Der vorherige globale Wert der Systemvariablen wird nach Verlassen des Block−Konstrukts automatisch wieder hergestellt.

```
$MaxExtraPrecision
```

```
50.
```