Equations and Systems of Equations

Linear and nonlinear equations

There are two groups of equations and systems of equations: linear and nonlinear. This division holds for both equations for unknown quantities (numbers) that are considered in this chapter and equations for unknown functions such as differential equations that will be considered later. Linear equations contain only zero and first combined powers of unknowns, such as the equation

```
ax == b
```

and the system of equations

```
ax + by == c
dx + ey == f.
```

Here *x* and *y* are unknowns and other symbols are parameters or numbers. Equations containing higher combined powers of unknowns such as x^2, xy, or functions of unknowns such as Sin[x], are nonlinear.

Linear equations and systems of equations are analytically solvable and the solution, if it exists, is unique. Nonlinear equations can have several solutions and analytical solutions are available in a limited number of cases such as quadratic, cubic, and quartic equations

$$ax^2 + bx + c == 0$$
$$ax^3 + bx^2 + cx + d == 0$$
$$ax^4 + bx^3 + cx^2 + dx + e == 0,$$

as well as some irrational equations

$$\sqrt{x - a} + \sqrt{x - b} == c,$$

some trigonometric equations

```
Sin[x] == Cos[2 x]
```

and other equations. All these equations are algebraic because they can be simplified to polynomial equations

If equations contain different functions of unknowns at the same time, such as

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

they are called transcedental equations because they are non-algebraic and never can be solved analytically. In many cases solution of nonlinear equations must be done numerically and special care should be taken in the case of many solutions.

Solution of equations with *Mathematica*

Linear equations

Both analytical and numerical solutions of equations are output by *Mathematica* in the form of rules

```
In[25]:= Solve[a x + b == 0, x]
```

Out[25]= $\left\{\left\{x \to -\frac{b}{a}\right\}\right\}$

that with the help of replacement can be further used to find the resulting expression for *x*

```
x /. Solve[a x + b == 0, x]
```

$$\left\{-\frac{b}{a}\right\}$$

or better

```
x /. Solve[a x + b == 0, x][[1]]
```

$$-\frac{b}{a}$$

or to define *x* as a function of the parameters

```
xab[a_, b_] = x /. Solve[a x + b == 0, x][[1]];
xab[a, b]
```

$$-\frac{b}{a}$$

■ Nonlinear algebraic equations

Note that the solution of an equation is a list of rules corresponding to all its roots. Quadratic equation has two roots,

$$\texttt{sol = Solve}\left[a\,x^2 + b\,x + c == 0,\ x\right]$$

$$\left\{\left\{x \to \frac{-b-\sqrt{b^2-4\,a\,c}}{2\,a}\right\},\ \left\{x \to \frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}\right\}\right\}$$

so that sol is a list of two elements. The roots of this equations form the list

```
x /. sol
```

$$\left\{\frac{-b-\sqrt{b^2-4\,a\,c}}{2\,a},\ \frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}\right\}$$

and the individual roots can be addressed as

```
x /. sol[[1]]
x /. sol[[2]]
```

$$\frac{-b-\sqrt{b^2-4\,a\,c}}{2\,a}$$

$$\frac{-b+\sqrt{b^2-4\,a\,c}}{2\,a}$$

The following irrational equation can be reduced to a quadratic equation by squaring it two times. The resulting quadratic equation has two roots, only one of which is the solution of the initial equation, why the other solution is parasite. This is why this equation is still algebraic.

$$\texttt{sol = Solve}\left[\sqrt{x-a} + \sqrt{x-b} == c,\ x\right]$$

$$\left\{\left\{x \to \frac{a^2-2\,a\,b+b^2+2\,a\,c^2+2\,b\,c^2+c^4}{4\,c^2}\right\}\right\}$$

The simplified expression for this solution is

$$\mathbf{sol = \left\{\left\{x \to \frac{Simplify\left[a^2 - 2\,a\,b + b^2\right] + Simplify\left[2\,a\,c^2 + 2\,b\,c^2 + c^4\right]}{4\,c^2}\right\}\right\}}$$

$$\left\{\left\{x \to \frac{(a-b)^2 + c^2\left(2\,a + 2\,b + c^2\right)}{4\,c^2}\right\}\right\}$$

Below is a particular case of the above equation. Since x is the only symbol here, it is the unknown and it does not have to be indicated in the Solve command

$$\mathbf{Solve\left[\sqrt{2\,x} + \sqrt{x+1} == \sqrt{2}\right]}$$

```
N[%]
```

$$\left\{\left\{x \to 7 - 4\sqrt{3}\right\}\right\}$$

```
{{x → 0.0717968}}
```

Some equations can be solved analytically in terms of complex numbers

$$\mathbf{Solve\left[\sqrt{x-1} + \sqrt{x} + \sqrt{x+1} == 3\right]}$$

$$\left\{\left\{x \to 7 - \frac{1}{2}\sqrt{\frac{160}{9} + \frac{224 \times 2^{2/3}}{9\left(41 + 9\,i\sqrt{47}\right)^{1/3}} + \frac{16}{9}\left(2\left(41 + 9\,i\sqrt{47}\right)\right)^{1/3}} - \frac{1}{2}\sqrt{\left(\frac{320}{9} - \frac{224 \times 2^{2/3}}{9\left(41 + 9\,i\sqrt{47}\right)^{1/3}} - \frac{16}{9}\left(2\left(41 + 9\,i\sqrt{47}\right)\right)^{1/3} + \frac{256}{3\sqrt{\frac{160}{9} + \frac{224 \times 2^{2/3}}{9\left(41+9\,i\sqrt{47}\right)^{1/3}} + \frac{16}{9}\left(2\left(41 + 9\,i\sqrt{47}\right)\right)^{1/3}}}\right)}\right\}\right\}$$

This solution is real but *Mathematica* cannot simplify this expression to the explicitly real form. Conversion to the numerical form shows that the solution is real indeed, up to the imaginary numerical noise term

```
N[%]
```

$$\left\{\left\{x \to 1.1866 - 2.46145 \times 10^{-17}\,i\right\}\right\}$$

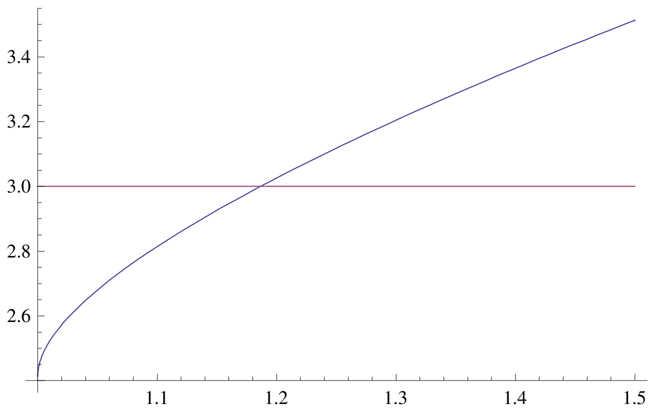
The latter can be eliminated with the help of the Chop command

```
Chop[%]
```

```
{{x → 1.1866}}
```

In such cases it is advisable to illustrate the equation graphically

```
Plot[{√(x - 1) + √x + √(x + 1), 3}, {x, 1, 1.5}, PlotRange → All]
```



Indeed, the solution above is OK. More practical is to force *Mathematica* to solve this equation numericaly from the beginning

```
Solve[√(x - 1) + √x + √(x + 1) == 3.]

{{x → 1.1866}}
```

For more complicated equations there can be an output saying that *Mathematica* has reduced the equation to a polynomial equation

```
Solve[√(x - 1) + √x + √(x + 1) + √(x + 2) == 5]

{{x → Root[-82 534 969 521 + 124 036 315 200 #1 -
      62 668 165 600 #1^2 + 14 394 809 600 #1^3 - 1 552 800 000 #1^4 + 64 000 000 #1^5 &, 1]}}
```

that can be solved only numerically

```
N[%]

{{x → 1.28929}}
```

Again, numerical solution can be done from the beginning

```
Solve[√(x - 1) + √x + √(x + 1) + √(x + 2) == 5.]

{{x → 1.28929}}
```

In this case one can use the NSolve command that works on polynomial equations, according to *Mathematica*'s help

```
NSolve[√(x - 1) + √x + √(x + 1) + √(x + 2) == 5]

{{x → 1.28929}}
```

Although this equation is not polynomial, *Mathematica* can reduce it to a polynomial equation, this is why NSolve works here.

Many trigonometric equations can be solved by their reduction to polynomial equations

```
Solve[Sin[x] == Cos[2 x], x]
N[%]
```

Solve::ifun :
Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{x\to-\frac{\pi}{2}\right\},\ \left\{x\to\frac{\pi}{6}\right\},\ \left\{x\to\frac{5\pi}{6}\right\}\right\}$$

```
{{x → -1.5708}, {x → 0.523599}, {x → 2.61799}}
```

Indeed, this equation is equivalent to a quadratic equation for Sin[x] because

$$\mathrm{Cos}[2\,x] = \mathrm{Cos}[x]^2 - \mathrm{Sin}[x]^2 = 1 - 2\,\mathrm{Sin}[x]^2$$

This formula can be obtained with *Mathematica* by

```
TrigExpand[Cos[2 x]] /. Cos[x]^2 → 1 - Sin[x]^2
```

$1 - 2\,\mathrm{Sin}[x]^2$

The solution of the resulting quadratic equation with respect to Sin[x] has the form

```
Solve[Sin[x] == 1 - 2 Sin[x]^2, Sin[x]]
```

$$\left\{\{\mathrm{Sin}[x]\to-1\},\ \left\{\mathrm{Sin}[x]\to\frac{1}{2}\right\}\right\}$$

This output constitutes two simple triginometric equations for x that can be solved to give the result above, $x = -\pi/2$, $x = \pi/6$, and $x = 5\pi/6$. Since the original equation is algebraic, also NSolve (applicable to polynomials) does the job

```
NSolve[Sin[x] == Cos[2 x], x]
```

Solve::ifun :
Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

```
{{x → -1.5708}, {x → 0.523599}, {x → 2.61799}}
```

Both Solve and NSolve do not take into account the periodicity of the trigonometric function and thus lose some solutions. Complete solutions can be obtained by Reduce

```
Reduce[Sin[x] == Cos[2 x], x]
N[%]
```

$$C[1]\in\mathrm{Integers}\ \&\&\ \left(x==-\frac{\pi}{2}+2\,\pi\,C[1]\ ||\ x==\frac{\pi}{6}+2\,\pi\,C[1]\ ||\ x==\frac{5\,\pi}{6}+2\,\pi\,C[1]\right)$$

```
C[1] ∈ Integers &&
  (x == -1.5708 + 6.28319 C[1] || x == 0.523599 + 6.28319 C[1] || x == 2.61799 + 6.28319 C[1])
```

Another example of an algebraic equation is

```
Solve[e^x - e^(2 x) + 1 == 0]
```

Solve::ifun :
Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information. ≫

$$\left\{\left\{x\to i\,\pi+\mathrm{Log}\left[\frac{1}{2}\left(-1+\sqrt{5}\right)\right]\right\},\ \left\{x\to\mathrm{Log}\left[\frac{1}{2}\left(1+\sqrt{5}\right)\right]\right\}\right\}$$

One of the two solutions is real and the other is complex. This is easy to understand because this equation is a quadratic equation for e^x. An attempt to solve this equation for e^x

```
Solve[ⅇ^x - (ⅇ^x)^2 + 1 == 0, ⅇ^x]
```

General::ivar : e^x is not a valid variable. ≫

General::ivar : e^x is not a valid variable. ≫

```
Solve[1 + ⅇ^x - ⅇ^(2 x) == 0, ⅇ^x]
```

does not work, although an equation was solved for Sin[x] above. Well, with the substitution $y = e^x$ one obtains

```
Solve[y - y^2 + 1 == 0]
```

$$\left\{\left\{y \to \frac{1}{2}\left(1-\sqrt{5}\right)\right\}, \left\{y \to \frac{1}{2}\left(1+\sqrt{5}\right)\right\}\right\}$$

The first solution is negative, although exponential of a real argument is positive. This, looking for real solutions of our equations, one has to drop it.

■ Transcedental (non-algebraic) equations

Transcedental equations cannot be solved with Solve, NSolve, and Reduce

```
Solve[Cos[x] == x]
```

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

```
Solve[Cos[x] == x]
```

```
NSolve[Cos[x] == x]
```

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

```
NSolve[Cos[x] == x]
```

```
Reduce[Cos[x] == x]
```

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

```
Reduce[Cos[x] == x]
```

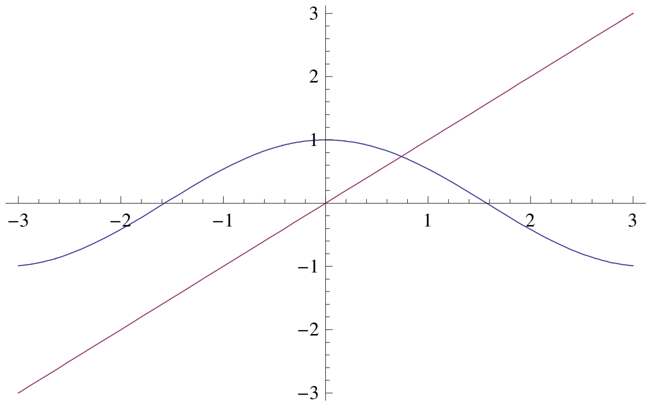
The only way to solve such equations is to use the essentially numerical procedure FindRoot

```
FindRoot[Cos[x] == x, {x, 0}]
```

```
{x → 0.739085}
```

0 in this command is the starting value of x. Since this equation has a single solution

```
Plot[{Cos[x], x}, {x, -3, 3}]
```

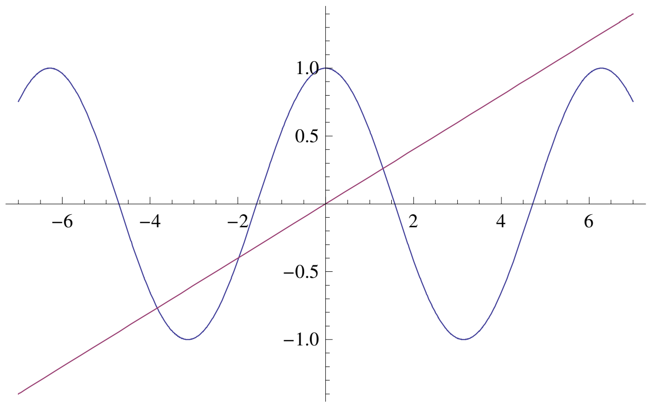


the choice of the starting point is irrelevant. If the equation has several solutions, usually (but not always!) FindRoot finds the solution closest to the starting point. For instance, the equation

```
Cos[x] == x / 5
```

has three solutions:

```
Plot[{Cos[x], x / 5}, {x, -7, 7}]
```



```
FindRoot[Cos[x] == x / 5, {x, 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 → 6.08183}
```

Here FindRoot got stuck in the vicinity of the near-solution $x \approx 6$.

```
FindRoot[Cos[x] == x / 5, {x, 2}]
```

```
{x → 1.30644}
```

This is the correct positive solution.

```
FindRoot[Cos[x] == x / 5, {x, -1}]
```

```
{x → -1.97738}
```

Also a correct solution.

```
FindRoot[Cos[x] == x / 5, {x, -5}]
```

```
{x → -3.83747}
```

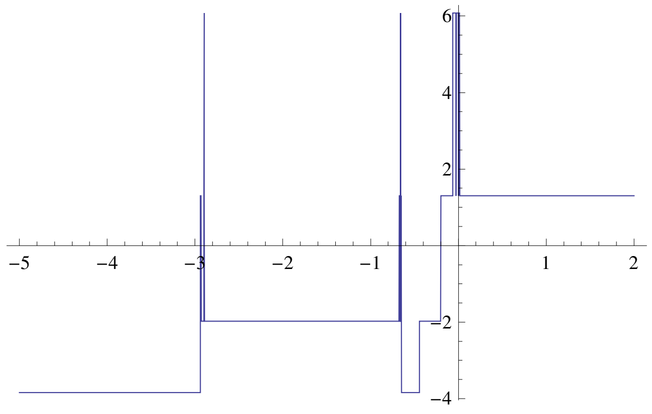
But

```
FindRoot[Cos[x] == x / 5, {x, -6}]
```

```
{x → -1.97738}
```

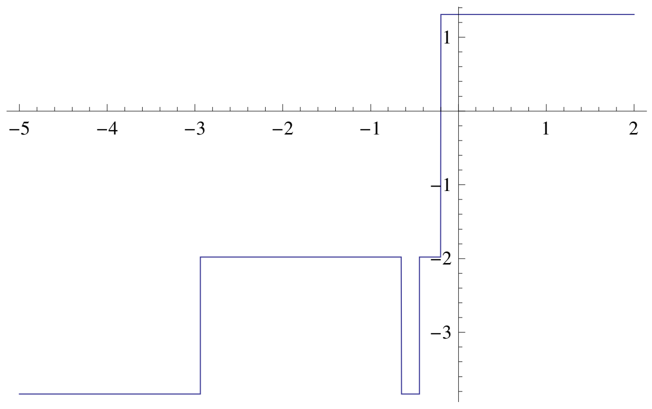
gives a solution that is not the closest to the starting point. One can plot the dependence of the solution on the starting point

```
xx0[x0_] := x /. FindRoot[Cos[x] == x / 5, {x, x0}]
Plot[xx0[x0], {x0, -5, 2}]
```



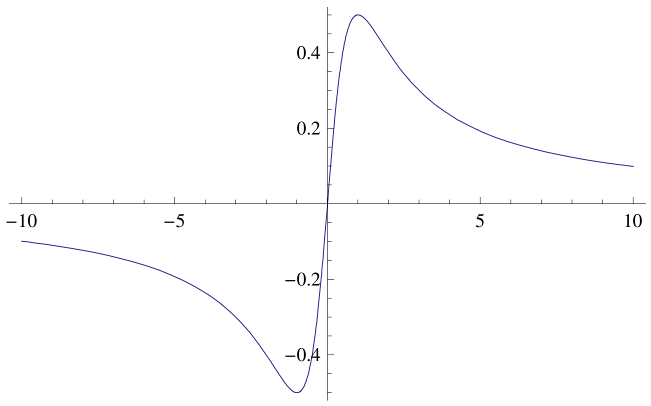
Here one can see all three roots plus the wrong root $x \approx 6$. One can eliminate the wrong root by bracketing the search interval to, say, {-5,2} in the FindRoot command

```
xx0[x0_] := x /. FindRoot[Cos[x] == x / 5, {x, x0, -5, 2}]
Plot[xx0[x0], {x0, -5, 2}]
```



There are functions that make problems in the standard version of FindRoot if the starting point is not sufficiently close to the root:

```
In[39]:= f[x_] = x/(1 + x^2);
         Plot[f[x], {x, -10, 10}]
Out[40]=
```



```
FindRoot[f[x] == 0, {x, 0.7}]
{x → 0.}
```

is the correct solution but already

```
FindRoot[f[x] == 0, {x, 0.8}]
FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations. >>
{x → -2.12308 × 10^30}
```

does not work, as well as

```
FindRoot[f[x] == 0, {x, 5}]
FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations. >>
{x → 6.68388 × 10^30}
```

In such situations one can use $\{x, x_1, x_2\}$ in the FindRoot statement. *Mathematica*'s help says that in this case FindRoot does not use analytical derivatives of $f[x]$ (see below on the role of derivatives) and computes derivatives numerically using two different values of x, the initial two values being x_1 and x_2. It also appears that if x_1 and x_2 are bracketing the root, the latter is found without problems:

```
FindRoot[f[x] == 0, {x, -100, 300}]
{x → -6.65758 × 10^-17}
```

That is a correct result up to numerical errors.

FindRoot works iteratively. The number of steps and function evaluations needed to reach the root with the required precision can be output via StepMonitor and EvaluationMonitor:

```
s = 0; e = 0;
FindRoot[Cos[x] == x / 5, {x, 1}, StepMonitor :→ s++,
 EvaluationMonitor :→ {e++, Print[e, "    ", x]}]
Print["Steps = ", s, "    Evaluations = ", e]
```

```
1   1.
2   1.32675
3   1.30648
4   1.30644
5   1.30644
       {x → 1.30644}
Steps = 4    Evaluations = 5
```

or

```
In[41]:= s = 0; e = 0;
         FindRoot[f[x] == 0, {x, -2, 200}, StepMonitor :→ s++,
          EvaluationMonitor :→ {e++, Print[e, "    ", x]}]
         Print["Steps" → s, "     Evaluations" → e]
1   -2.
2   200.
3   197.506
4   97.7531
5   47.8766
6   22.9383
7   10.4691
8   4.23457
9   1.11728
10   -0.609815
11   0.253735
12   -0.0477139
13   0.0024644
14   -5.32008 × 10^-6
15   3.22404 × 10^-11
16   -9.12497 × 10^-22
Out[42]= {x → -9.12497 × 10^-22}
Steps → 14    Evaluations → 16
```

■ Numerical method of solving transcedental equations

The basic method for solving transcedental equations of the type $F[x] == 0$ is the Newton method, in which the function is linearized around the starting point x_0

$$F_{lin}[x] = F[x_0] + F'[x_0]\,(x - x_0)$$

and the root of the $F_{lin}[x]$ is found as

$$x_1 = x_0 - \frac{F[x_0]}{F'[x_0]}.$$

Taking x_1 as the next starting point, one repeats this procedure iteratively until $|x_1 - x_0| < \delta$. Below is the procedure code. The first argument of the Module statement is the list of local variables. As we need xList as an output, we keep it global. Also the maximal number of iterations kMax is kept global.

```
In[12]:= FindRootNewton[Func_, xStart_] := Module[{del, k, x0, x1, FDer},
            FDer[x_] = ∂x Func[x];
            x1 = xStart;  x0 = xStart + 1;   k = 1;     del = 10^-16;
            xList = {x1};
            While[Abs[x1 - x0] > del && k < kMax,
             k ++;
             x0 = x1;
                         Func[x0]
             x1 = x0 - ----------;
                         FDer[x0]
             xList = Append[xList, x1]
             ];
            x1];
```

FindRootNewton is programmed as a function, the output is the value $x1$, not a rule. Its application is the following:

```
In[133]:= F[y_] = y + y^2;   kMax = 30;
          FindRootNewton[F, 1.]

Out[134]= 0.
```

For a comparison,

```
In[23]:= FindRoot[F[x], {x, 1.}]

Out[23]= {x → 5.42101 × 10^-20}
```

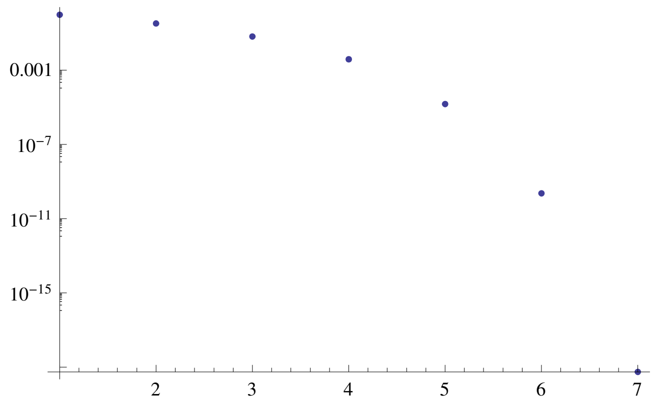
Number of iterations and iteration results in FindRootNewton:

```
In[60]:= Length[xList]
         xList

Out[60]= 8

Out[61]= {1., 0.333333, 0.0666667, 0.00392157, 0.000015259, 2.32831 × 10^-10, 5.42101 × 10^-20, 0.}
```

One can see that the convergence becomes very fast when x becomes close to the root.

```
In[62]:= ListLogPlot[xList]
```

Out[62]=



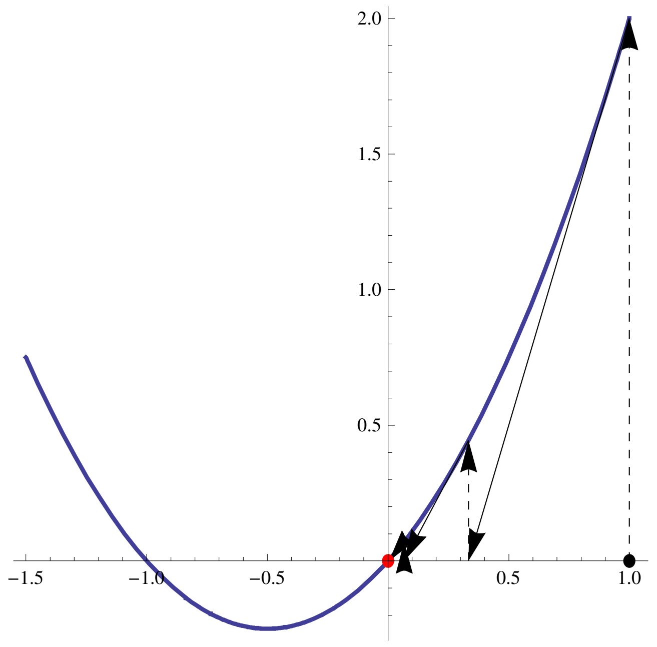
Here is the plotting procedure visualizing how the Newton's method works. *kShow* is the maximal number of iterations to show.

```
In[137]:= ShowHowNewtonWorks[kShow_, xLeft_, xRight_] := Show[
             Plot[F[x], {x, xLeft, xRight}, PlotRange → All, PlotStyle → {Thick}, AspectRatio → 1],
             Graphics[{Dashed, Arrowheads[Medium],
               Table[Arrow[{{xList[[k]], 0}, {xList[[k]], F[xList[[k]]]}}], {k, 1, kShow}]}],
             Graphics[{Arrowheads[Medium], Table[
                Arrow[{{xList[[k]], F[xList[[k]]]}, {xList[[k + 1]], 0}}], {k, 1, kShow}]}],
             Graphics[Disk[{First[xList], 0}, 0.01 (xRight - xLeft)]],
             Graphics[{Red, Disk[{Last[xList], 0}, 0.01 (xRight - xLeft)]}]
            ];
```

Now visualize

```
In[138]:= kShow = 3;
          ShowHowNewtonWorks[kShow, -1.5, 1]
```

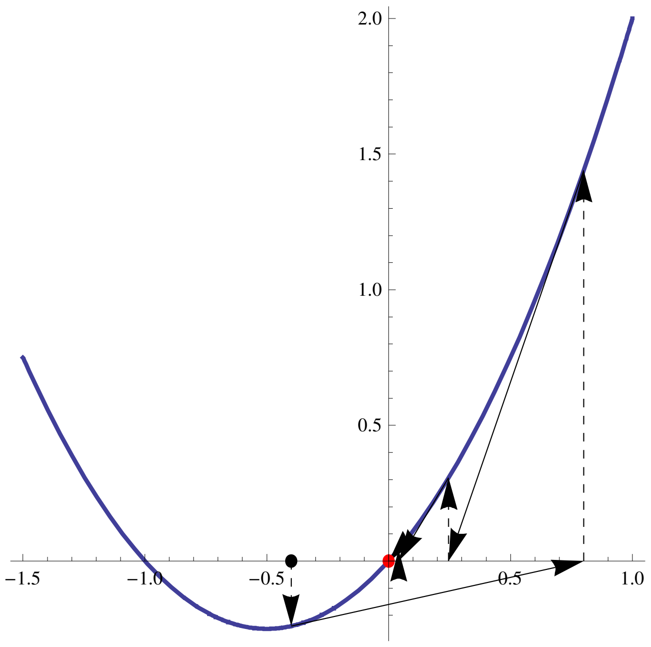
Out[139]=



If, for the same function, one takes the starting point on the other side of the root, the situation can become dangerous. For the starting value –0.4 the procedure still converges to the same root.

```
In[144]:= F[y_] = y + y^2;  kMax = 30;
          FindRootNewton[F, -0.4]
          kShow = 4;
          ShowHowNewtonWorks[kShow, -1.5, 1]

Out[145]= 0.
```

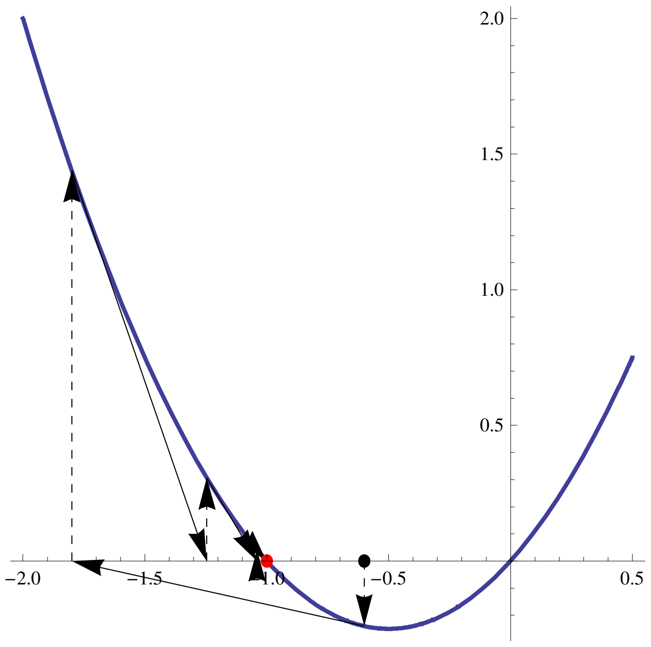
Out[147]=



But for the starting value −0.6 and less the procedure converges to another root.

```
In[148]:= F[y_] = y + y^2;  kMax = 30;
          FindRootNewton[F, -0.6]
          kShow = 4;
          ShowHowNewtonWorks[kShow, -2, 0.5]

Out[149]= -1.
```

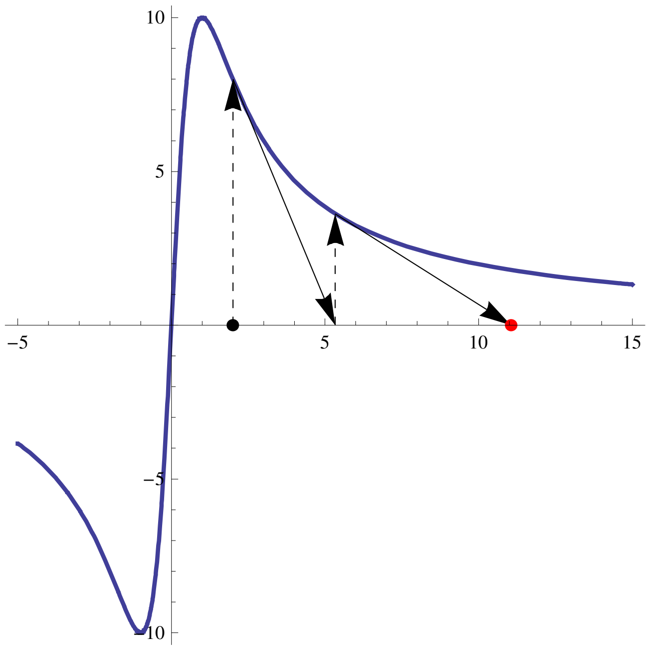
Out[151]=



For the function considered above, $x / (1 + x^2)$ and similar, the Newton method diverges for most of starting points

```
In[152]:= F[x_] = 20 x/(1 + x^2);   kMax = 3;
          FindRootNewton[F, 2.]
          kShow = 2;
          ShowHowNewtonWorks[kShow, -5, 15]

Out[153]= 11.0553
```

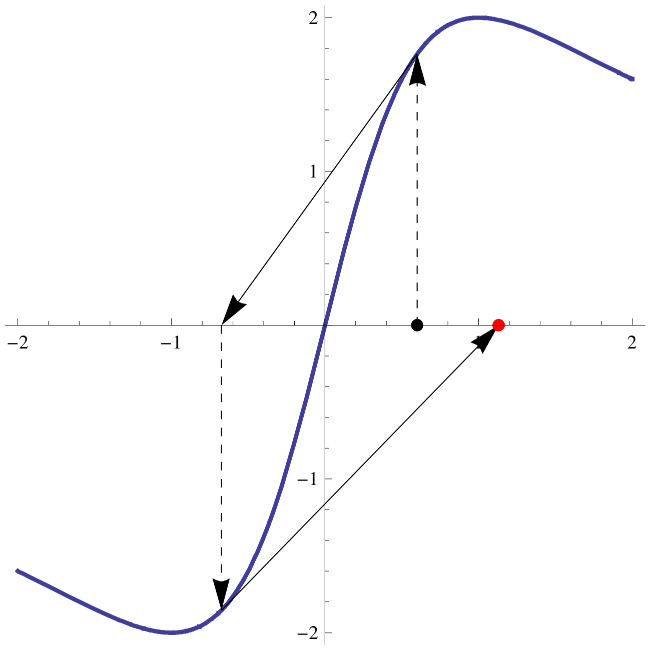
Out[155]=



or even

```
In[156]:= F[x_] = 4 x/(1 + x^2);   kMax = 3;
          FindRootNewton[F, 0.6]
          kShow = 2;
          ShowHowNewtonWorks[kShow, -2, 2]

Out[157]= 1.12991
```

Out[159]=

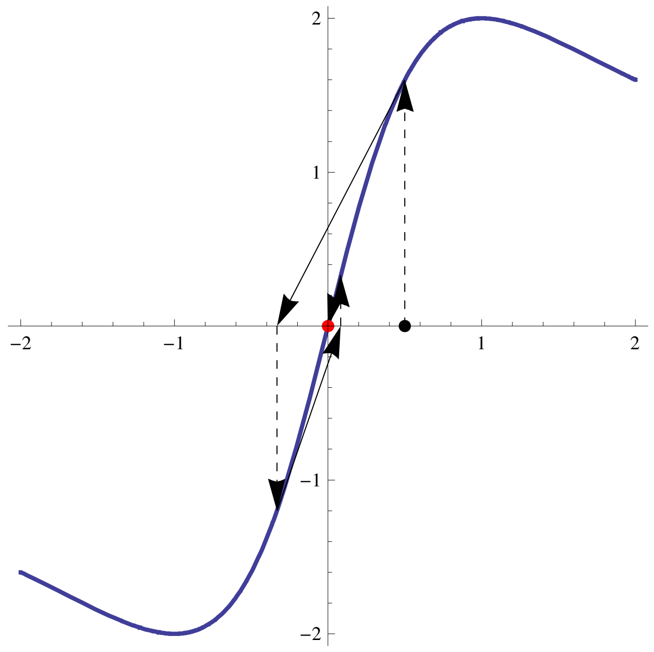


But if the starting point is closer to the root, the procedure converges:

```
In[160]:= F[x_] = 4 x/(1 + x^2);   kMax = 5;
          FindRootNewton[F, 0.5]
          kShow = 3;
          ShowHowNewtonWorks[kShow, -2, 2]
```

```
Out[161]= 3.16642 × 10^-9
```

Out[163]=



For such "pathological" functions, one can apply at first the slower converging bisection method (if the root is bracketed) and then switch to the fast-converging Newton method closer to the root. Professional-grade nonlinear-equation solvers usually employ both methods with the internal monitoring of the convergence.

Systems of equations

Generally, for a system of equations to be solvable, the number of unknowns should be equal to the number of equations.

If the number of equations is smaller than the number of unknowns, this system of equations is called underdetermined. In this case one should look for additional equations.

If the number of equations is larger than the number of unknowns, this system of equations is called overderdetermined. In this case one should check whether some equations follow from the others and thus can be dropped. If the situation persists, something is fundamentally wrong with the model.

Below we consider only systems of equations with equal numbers of equations and unknowns.

■ Systems of linear equations

System of linear equations can be solved except for the cases when equations are contradictory, such as

```
x + y == 0
x + y == 1
```

or

```
x + y + z == 0
x + y + 2 z == 1
2 x + 2 y + 3 z == 2
```

Here adding the first and second equation, one obtains $2x + 2y + 3z == 2$ that contradicts the third equation. In all these cases the determinant of the system of equations is zero (see below).

Consider a rigid rod of mass M and length L lying on two supports. Support 1 is located at the distance x_1 from the left end of the rod and support 2 is located at the distance x_2 from the left end of the rod. Let us find the upward reaction forces F_1 and F_2 acting on the rod from each support.

There are two linear equations

```
F1 + F2 - M g == 0  (* Sum of all forces is zero *)
F1 x1 + F2 x2 - M g L / 2 == 0
  (* Sum of all torques (here with respect to the left end) is zero *)
```

This system of equation can be solved with *Mathematica*

```
In[104]:= Solve[{F1 + F2 - M g == 0, F1 x1 + F2 x2 - M g L / 2 == 0}, {F1, F2}] // Simplify
```

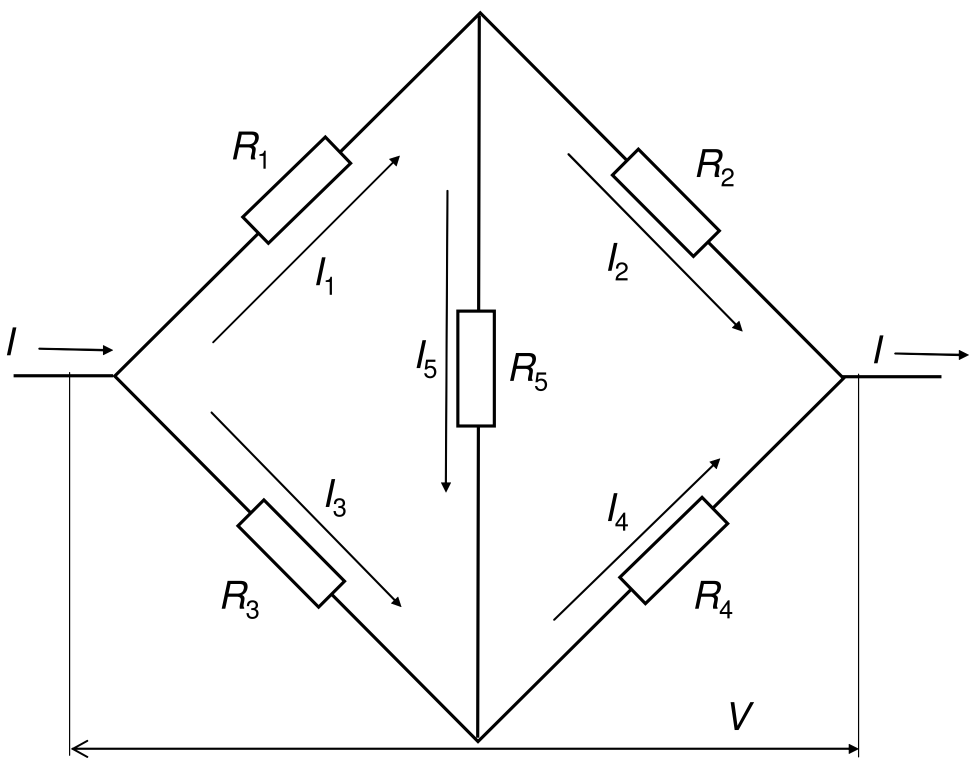
$$\text{Out[104]=}\quad \left\{\left\{F_1 \to \frac{g M (L - 2 x_2)}{2 (x_1 - x_2)},\ F_2 \to -\frac{g M (L - 2 x_1)}{2 (x_1 - x_2)}\right\}\right\}$$

or by hand. The human form of this result is

$$F_1 = \frac{L/2 - x_2}{x_1 - x_2} M g, \qquad F_2 = \frac{L/2 - x_1}{x_2 - x_1} M g.$$

In the case of three or more supports, the number of unknowns is the number of supports, whereas there are still two equations. Thus the problem becomes underdetermined. It can be made well determined if one renounces the model of the rigid rod and takes into account its elasticity. In this case, however, the problem becomes much more complicated.

The problem with two supports is very simple and it does not require *Mathematica*. Let us find the effective resistance of the so-called bridge consisting of five resistors.



The currents in the nodes are related by the obvious Kirchhof's equations

$$I == I_1 + I_3$$
$$I_1 == I_2 + I_5$$
$$I_3 + I_5 == I_4$$

while the equation for the rightmost node $I_2 + I_4 == I$ should be discarded because it follows from the above three equations. Ohm's law for the three ways from left to right have the form

$$R_1\ I_1 + R_2\ I_2 == V$$
$$R_3\ I_3 + R_4\ I_4 == V$$
$$R_1\ I_1 + R_5\ I_5 + R_4\ I_4 == V$$

One can write down more equations of this type but they follow from the above equations. For the given total voltage V, one finds 6 currents from the 6 equations and then defines the effective resistance as

$$R = \frac{V}{I}$$

Solution of this problem by hand is cumbersome. One can eliminate some currents from the first three equations but then one has to solve the remaining system of three equations with three unknowns. Mathematica solves this problem as

```
In[1]:= sol = Solve[{
           II == II_1 + II_3,
           II_1 == II_2 + II_5,
           II_3 + II_5 == II_4,
           R_1 II_1 + R_2 II_2 == V,
           R_3 II_3 + R_4 II_4 == V,
           R_1 II_1 + R_5 II_5 + R_4 II_4 == V
          }, {II, II_1, II_2, II_3, II_4, II_5}
         ]
```

Out[1]=

$$\Big\{\Big\{\mathrm{II} \to -\frac{-V R_1 R_2 - V R_2 R_3 - V R_1 R_4 - V R_3 R_4 - V R_1 R_5 - V R_2 R_5 - V R_3 R_5 - V R_4 R_5}{R_1 R_2 R_3 + R_1 R_2 R_4 + R_1 R_3 R_4 + R_2 R_3 R_4 + R_1 R_3 R_5 + R_2 R_3 R_5 + R_1 R_4 R_5 + R_2 R_4 R_5},$$

$$\mathrm{II}_2 \to -\frac{-V R_1 R_4 - V R_3 R_4 - V R_3 R_5 - V R_4 R_5}{R_1 R_2 R_3 + R_1 R_2 R_4 + R_1 R_3 R_4 + R_2 R_3 R_4 + R_1 R_3 R_5 + R_2 R_3 R_5 + R_1 R_4 R_5 + R_2 R_4 R_5},$$

$$\mathrm{II}_1 \to -\frac{-V R_2 R_3 - V R_3 R_4 - V R_3 R_5 - V R_4 R_5}{R_1 R_2 R_3 + R_1 R_2 R_4 + R_1 R_3 R_4 + R_2 R_3 R_4 + R_1 R_3 R_5 + R_2 R_3 R_5 + R_1 R_4 R_5 + R_2 R_4 R_5},$$

$$\mathrm{II}_5 \to -\frac{-V R_2 R_3 + V R_1 R_4}{R_1 R_2 R_3 + R_1 R_2 R_4 + R_1 R_3 R_4 + R_2 R_3 R_4 + R_1 R_3 R_5 + R_2 R_3 R_5 + R_1 R_4 R_5 + R_2 R_4 R_5},$$

$$\mathrm{II}_3 \to -\frac{-V R_1 R_2 - V R_1 R_4 - V R_1 R_5 - V R_2 R_5}{R_1 R_2 R_3 + R_1 R_2 R_4 + R_1 R_3 R_4 + R_2 R_3 R_4 + R_1 R_3 R_5 + R_2 R_3 R_5 + R_1 R_4 R_5 + R_2 R_4 R_5},$$

$$\mathrm{II}_4 \to -\frac{(-V R_1 + V (R_1 + R_2)) R_3 + V (R_1 R_2 + (R_1 + R_2) R_5)}{R_3 (-R_1 R_2 - (R_1 + R_2) R_4 - (R_1 + R_2) R_5) - R_4 (R_1 R_2 + (R_1 + R_2) R_5)}\Big\}\Big\}$$

and the effective resistance is given by

```
In[2]:= Simplify[ 1/(II /. sol[[1]]) V ]
```

Out[2]=

$$\frac{R_2 (R_4 R_5 + R_3 (R_4 + R_5)) + R_1 (R_2 (R_3 + R_4) + R_4 R_5 + R_3 (R_4 + R_5))}{R_3 R_4 + R_3 R_5 + R_4 R_5 + R_2 (R_3 + R_5) + R_1 (R_2 + R_4 + R_5)}$$

(*V* has been put after the replacement because otherwise the command does not work. Also the assignment *R* = ... does not work.). The human form of the result is (Excersize: Try to obtain this formula with *Mathematica* - could be difficult or impossible*)*

$$R = \frac{(R_1 + R_2)(R_3 + R_4) R_5 + (R_1 + R_3) R_2 R_4 + R_1 R_3 (R_2 + R_4)}{(R_1 + R_2 + R_3 + R_4) R_5 + (R_1 + R_3)(R_2 + R_4)}.$$

This formula is symmetric with respect to $\{R_1 \rightleftharpoons R_2,\ R_3 \rightleftharpoons R_4\}$ and $\{R_1 \rightleftharpoons R_3,\ R_2 \rightleftharpoons R_4\}$. Check:

$$\frac{\frac{R_2 (R_4 R_5 + R_3 (R_4 + R_5)) + R_1 (R_2 (R_3 + R_4) + R_4 R_5 + R_3 (R_4 + R_5))}{R_3 R_4 + R_3 R_5 + R_4 R_5 + R_2 (R_3 + R_5) + R_1 (R_2 + R_4 + R_5)} ==}{\frac{(R_1 + R_2)(R_3 + R_4) R_5 + (R_1 + R_3) R_2 R_4 + R_1 R_3 (R_2 + R_4)}{(R_1 + R_2 + R_3 + R_4) R_5 + (R_1 + R_3)(R_2 + R_4)}\ //\ \mathrm{Simplify}}$$

```
Out[16]= True
```

In the case $R_5 = 0$ the result simplifies to

$$R = \frac{R_1 R_3}{R_1 + R_3} + \frac{R_2 R_4}{R_2 + R_4}$$

that can be obtained in a simple way. In the case $R_5 = \infty$ the result simplifies to

$$R = \frac{(R_1 + R_2)\ (R_3 + R_4)}{(R_1 + R_2) + (R_3 + R_4)}$$

that also can be obtained elementarily.

Systems of linear equations in the matrix form

Systems of linear equations can be written in the matrix form

```
A.X == B,
```

where is the square matrix of the coefficients of the system of equations, *X* is the vector (List) of unknowns, and *B* is the vector (List) of right parts. In particular, for the problem of a rod on two supports described by the system of equations

```
F₁ + F₂ - M g == 0  (* Sum of all forces is zero *)
F₁ x₁ + F₂ x₂ - M g L / 2 == 0
  (* Sum of all torques (here with respect to the left end) is zero *)
```

one has

In[36]:= $A = \begin{pmatrix} 1 & 1 \\ x_1 & x_2 \end{pmatrix}$

```
B = {M g, M g L / 2}
X = {F₁, F₂}
```

```
Out[36]= {{1, 1}, {x₁, x₂}}
```

Out[37]= $\left\{g\,M,\ \frac{g\,L\,M}{2}\right\}$

```
Out[38]= {F₁, F₂}
```

It looks natural to define

In[20]:= $B = \begin{pmatrix} M\,g \\ M\,g\,L/2 \end{pmatrix}$

$X = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$

Out[20]= $\left\{\{g\,M\},\ \left\{\frac{g\,L\,M}{2}\right\}\right\}$

```
Out[21]= {{F₁}, {F₂}}
```

but, unfortunately, the current version of *Mathematica* understands this as Lists with two indices (Lists of Lists) that are inappropriate for us. Please, rerun now the previous definitions!

One can see that technically matrix is a List with two indices, whereas a vector is a list with two indices. The equations for the rod are

```
In[25]:= A.X == B
```

Out[25]= $\{F_1 + F_2,\ F_1\,x_1 + F_2\,x_2\} == \left\{g\,M,\ \frac{g\,L\,M}{2}\right\}$

that is the same as equations written above.

The advantage of the matrix form is that properties of matrices are very well investigated and many operations can be performed on matrices as the whole rather than on their components.There are functions of matrices etc. The same pertains to vectors. If the problem is put into the matrix-vector form (that is, in the form of Lists) , one says that the problem is vectorized. Vectorized problems are solved faster because there are special highly efficient algorithms for Lists.

The solution of the rod problem with *Mathematica* can be obtained as

```
In[39]:= Solve[A.X == B, X] // Simplify
```

Out[39]= $\left\{\left\{F_1 \to \frac{g\,M\,(L-2\,x_2)}{2\,(x_1-x_2)},\ F_2 \to -\frac{g\,M\,(L-2\,x_1)}{2\,(x_1-x_2)}\right\}\right\}$

Also one can solve the matrix equation simply by left-multiplying the right and left parts by the inverse *A* matrix

```
In[28]:= Inverse[A]
```

Out[28]= $\left\{\left\{\frac{x_2}{-x_1+x_2},\ -\frac{1}{-x_1+x_2}\right\},\ \left\{-\frac{x_1}{-x_1+x_2},\ \frac{1}{-x_1+x_2}\right\}\right\}$

and using

In[31]:= **Inverse[A].A ==** $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ **// Simplify**

```
Out[31]= True
```

or, in other words

```
In[32]:= Inverse[A].A == IdentityMatrix[2] // Simplify

Out[32]= True
```

This yields the solution

```
In[34]:= X = Inverse[A].B // Simplify
```

Out[34]= $\left\{\frac{g\,M\,(L-2\,x_2)}{2\,(x_1-x_2)},\ -\frac{g\,M\,(L-2\,x_1)}{2\,(x_1-x_2)}\right\}$

Note that the command

In[27]:= $\mathbf{A}^{-1}$

Out[27]= $\left\{\{1,\ 1\},\ \left\{\frac{1}{x_1},\ \frac{1}{x_2}\right\}\right\}$

does not yield the inverse matrix, it simply inverts each element. All operations on matrices have special names. Similarly, the exponential of a matrix is given by MatrixExp[*A*] and not by Exp[*A*] (the latter exponentiates each element).

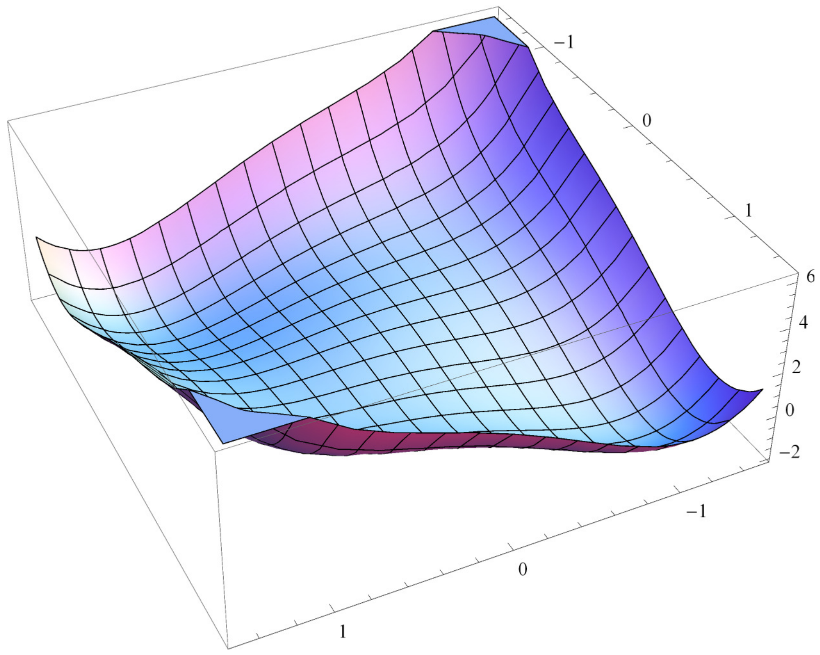
■ Systems of nonlinear equations

Physically, systems of nonlinear equations arise in finding the equilibrium state of a system corresponding to the minimal potential energy and in other problems. Systems of nonlinear equations can be solved by *Mathematica* by the same methods as nonlinear equations.

Consider, as an illustration, two particles in one dimension, described by the coordinates x and y that repel each other with the quadratic potential $U_{12}[x,\ y] = -(x-y)^2$. For each particle there is a confinement potential $U_1[x] = x^4$ (and the same for y) that prevents the particles from going too far from each other. As it is seen from the plot below, the minima of the energy correspond to $\{x = \pm 1,\ y = \mp 1\}$.

```
In[24]:= xMax = 1.5;
         U[x_, y_] = - (x - y)^2 + (x^4 + y^4);
         Plot3D[U[x, y], {x, -xMax, xMax}, {y, -xMax, xMax}]
```

Out[26]=



Extrema of a function of two variables satisfy

$$\partial_x U[x,\ y] == 0, \qquad \partial_y U[x,\ y] == 0$$

that form a system of two equations for two unknowns

```
In[31]:= ∂x U[x, y] == 0
         ∂y U[x, y] == 0
```

Out[31]= $4\ x^3 - 2\ (x - y) == 0$

Out[32]= $2\ (x - y) + 4\ y^3 == 0$

This system of equations can be solved analytically. Adding and subtracting the equations, one obtains the equivalent system of equations

$$x^3 + y^3 == 0$$
$$x^3 - y^3 - (x - y) == 0 \qquad \Rightarrow \qquad (x - y)\left(x^2 + xy + y^2 - 1\right) == 0$$

From the first equation one obtains $y = -x$. Substituting this into the second equation, one obtains the equation for x

$$x\left(x^2 - 1\right) == 0$$

that solves to

$$x = 0, \pm 1.$$

Here $x = 0$ is a local maximum while $x = \pm 1$ are minima.

Solution of this problem with *Mathematica* that is good for polynomial functions

```
In[49]:= Solve[{∂ₓU[x, y] == 0, ∂ᵧU[x, y] == 0}]
```

Out[49]= $\{\{x\to -1,\ y\to 1\},\ \{x\to 0,\ y\to 0\},\ \{x\to 0,\ y\to 0\},\ \{x\to 0,\ y\to 0\},$

$\{x\to 1,\ y\to -1\},\ \left\{x\to \frac{1}{2}\sqrt{\frac{1}{4}-\frac{i\sqrt{3}}{4}}+\frac{1}{2}i\sqrt{3\left(\frac{1}{4}-\frac{i\sqrt{3}}{4}\right)},\ y\to \sqrt{\frac{1}{4}-\frac{i\sqrt{3}}{4}}\right\},$

$\left\{x\to \frac{1}{2}\left(-\sqrt{\frac{1}{4}-\frac{i\sqrt{3}}{4}}-i\sqrt{3\left(\frac{1}{4}-\frac{i\sqrt{3}}{4}\right)}\right),\ y\to -\sqrt{\frac{1}{4}-\frac{i\sqrt{3}}{4}}\right\},$

$\left\{x\to \frac{1}{2}\sqrt{\frac{1}{4}+\frac{i\sqrt{3}}{4}}-\frac{1}{2}i\sqrt{3\left(\frac{1}{4}+\frac{i\sqrt{3}}{4}\right)},\ y\to \sqrt{\frac{1}{4}+\frac{i\sqrt{3}}{4}}\right\},$

$\left\{x\to \frac{1}{2}\left(-\sqrt{\frac{1}{4}+\frac{i\sqrt{3}}{4}}+i\sqrt{3\left(\frac{1}{4}+\frac{i\sqrt{3}}{4}\right)}\right),\ y\to -\sqrt{\frac{1}{4}+\frac{i\sqrt{3}}{4}}\right\}\}$

yields all extrema, including unphysical complex solutions. Or, numerically,

```
In[51]:= NSolve[{∂ₓU[x, y] == 0, ∂ᵧU[x, y] == 0}]

Out[51]= {{x → -1., y → 1.}, {x → 1., y → -1.}, {x → -0.612372 + 0.353553 i, y → -0.612372 - 0.353553 i},
          {x → -0.612372 - 0.353553 i, y → -0.612372 + 0.353553 i},
          {x → 0.612372 + 0.353553 i, y → 0.612372 - 0.353553 i},
          {x → 0.612372 - 0.353553 i, y → 0.612372 + 0.353553 i},
          {x → 0. + 6.49367 × 10^-9 i, y → 0. + 6.49367 × 10^-9 i},
          {x → 0. - 6.49367 × 10^-9 i, y → 0. - 6.49367 × 10^-9 i}, {x → 0., y → 0.}}
```

FindRoot can find solve systems of transcedental equations, if the initial points are specified and are sufficiently close to the actual roots

```
In[53]:= FindRoot[{∂ₓU[x, y] == 0, ∂ᵧU[x, y] == 0}, {{x, 0.5}, {y, -0.5}}]

Out[53]= {x → -1., y → 1.}
```

Another example. Three beads electrically charged with positive charges Q_1, Q_2, and Q_3 can freely glide on a circular contour of radius R. Find their equilibrium positions.

Solution. The potential energy of the system is a sum of Coulomb potential energies of point charges,

$$U[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3] = \frac{Q_1 Q_2}{|\mathbf{r}_1 - \mathbf{r}_2|} + \frac{Q_2 Q_3}{|\mathbf{r}_2 - \mathbf{r}_3|} + \frac{Q_3 Q_1}{|\mathbf{r}_3 - \mathbf{r}_1|}.$$

This potential energy is invariant with respect to the rotation of the system of charges as the whole. Thus one can fix the position of (say) the third bead as $\{x_3, y_3\} = \{R, 0\}$ (that is, $\phi_3 = 0$) and describe the coordinates of the other two beads in the polar coordinate system

$$\{x_1, y_1\} = \{R \operatorname{Cos}[\phi_1], R \operatorname{Sin}[\phi_1]\}$$
$$\{x_2, y_2\} = \{R \operatorname{Cos}[\phi_2], R \operatorname{Sin}[\phi_2]\}$$
$$\{x_3, y_3\} = \{R, 0\}$$

The potential energy of the system can be rewritten as

$$U[\phi_1, \phi_2] = \frac{Q_1 Q_2}{R\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} + \frac{Q_2 Q_3}{R\sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}} + \frac{Q_3 Q_1}{R\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}}$$

or, in terms of the two angles $\phi_1$and ϕ_2

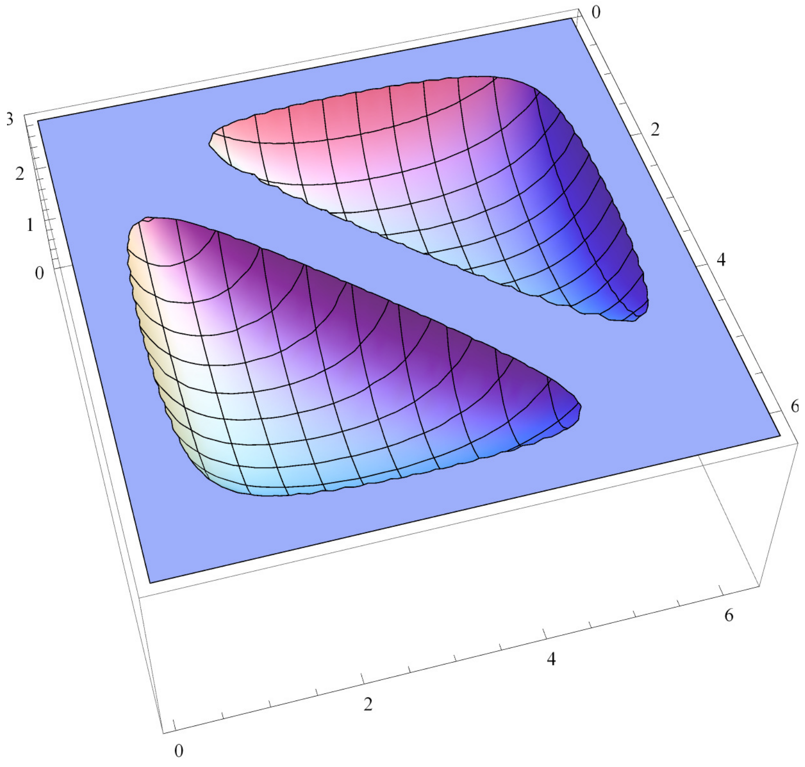
$$U[\phi_1, \phi_2] = \frac{Q_1 Q_2}{R\sqrt{(\operatorname{Cos}[\phi_1] - \operatorname{Cos}[\phi_2])^2 + (\operatorname{Sin}[\phi_1] - \operatorname{Sin}[\phi_2])^2}} + \frac{Q_2 Q_3}{R\sqrt{(\operatorname{Cos}[\phi_2] - 1)^2 + (\operatorname{Sin}[\phi_2])^2}} + \frac{Q_3 Q_1}{R\sqrt{(1 - \operatorname{Cos}[\phi_1])^2 + (\operatorname{Cos}[\phi_1])^2}}.$$

Now the potential energy can be plotted (Sorry, cannot define it as a function of ϕ_1 and ϕ_2)

In[1]:=
$$U[\phi 1\_, \phi 2\_] := \frac{Q_1 Q_2}{R\sqrt{(\operatorname{Cos}[\phi 1] - \operatorname{Cos}[\phi 2])^2 + (\operatorname{Sin}[\phi 1] - \operatorname{Sin}[\phi 2])^2}} + \frac{Q_2 Q_3}{R\sqrt{(\operatorname{Cos}[\phi 2] - 1)^2 + (\operatorname{Sin}[\phi 2])^2}} + \frac{Q_3 Q_1}{R\sqrt{(1 - \operatorname{Cos}[\phi 1])^2 + (\operatorname{Sin}[\phi 1])^2}};$$

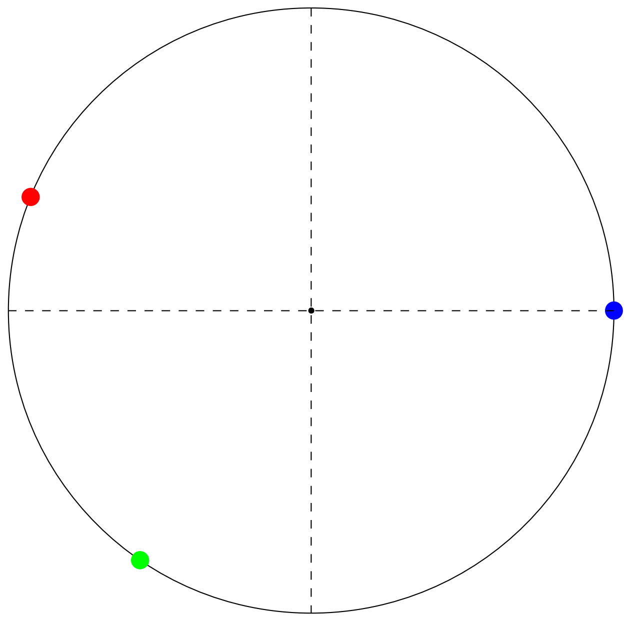
```
ParameterSet = {R → 1, Q1 → 1, Q2 → 1, Q3 → 1};
Plot3D[U[ϕ1, ϕ2] /. ParameterSet, {ϕ1, 0, 2 π}, {ϕ2, 0, 2 π}, PlotRange → {0, 3}]
```

Power::infy : Infinite expression $\frac{1}{0.}$ encountered. ≫

Out[3]=


Obviously, U diverges if two of three particles approach each other. For all charges being the same, as plotted, the minima correspond to $\{\phi_1 = 2\pi/3,\ \phi_2 = 4\pi/3\}$ and $\{\phi_1 = 4\pi/3,\ \phi_2 = 2\pi/3\}$. It is interesting that Solve cannot find this obvious solution. Since there are no local maxima and saddle points, FindRoot works without problems, including the cases of different charges where it is impossible to find their equilibrium positions analytically. At the end we plot equilibrium positions of the beads.

In[151]:=
```
ParameterSet = {R → 1, Q1 → 0.3, Q2 → .1, Q3 → 1};
MyRoots = FindRoot[{(∂ϕ1 U[ϕ1, ϕ2] /. ParameterSet) == 0,
    (∂ϕ2 U[ϕ1, ϕ2] /. ParameterSet) == 0}, {{ϕ1, 1}, {ϕ2, 2}}];

RR = R /. ParameterSet;
ϕ1 = ϕ1 /. MyRoots;
ϕ2 = ϕ2 /. MyRoots;
Graphics[{
  Circle[{0, 0}, RR],
  {Red, Disk[{RR Cos[ϕ1], RR Sin[ϕ1]}, 0.03]},
  {Green, Disk[{RR Cos[ϕ2], RR Sin[ϕ2]}, 0.03]},
  {Blue, Disk[{RR , 0}, 0.03]},
  Disk[{0, 0}, 0.01],
  {Dashed, Line[{{RR , 0}, {-RR , 0}}]},
  {Dashed, Line[{{0 , RR}, {0, -RR}}]}
 }]
```

Out[156]=


The main algorithm used by FindRoot for systems of nonlinear equations is again the Newton method. Equations are linearized near the starting point and the resulting linear equations are solved to give the new starting point. If the starting point is close to the root, the convergence is very fast.