Reprint from
The *Mathematica* Conference

June, 1992   Boston, MA

875.15.10.1992

Solving Equations Symbolically with *Mathematica*

Alexei V. Bocharov

1 Abstract

The course concentrates on the equation-solving facilities of *Mathematica* including solving algebraic equations with `Solve` as well as solving differential equations with `DSolve`. For completeness attention is given to the `GroebnerBasis` function.

The course consists of two layers: the first serves to show and teach using the corresponding *Mathematica* functions for a variety of equation-solving tasks. The intention of the second layer is a thorough discussion of what classes of equations are likely to be solvable at the state-of-the-art level of computer science and hardware/software technology. The basic understanding of the *complexity* of solving procedures is given, the solvability/unsolvability trade-off is shown for several examples.

2 Introduction

Equations of diverse types arise in applications as relations between the involved quantities.

Consider, for instance, the following classroom-level problem:

> John started from Champaign to Chicago at noon and he is travelling at 65 mph, and Jim started from Chicago to Champaign at 1pm and he is travelling at 67 mph.
>
> At what time do they meet if neither one of them changes speed?

Now, assuming that *t* is the unknown time (pm) we get the following equation:

```
In[1]:= eq1 = 65 t + 67 (t-1) == 132

Out[1]= 67 (-1 + t) + 65 t == 132
```

Another textbook example is the following problem in physics:

> If an object is pushed down from the height of 100 ft with the initial speed of 10 ft/sec. How long will it take to reach the ground assuming that it falls freely?

Again, assuming *t* is the unknown time we obtain the following relation where the 32.185 is the well-known free-fall acceleration:

```
In[2]:= eq2 = 10 t + 32.185 t^2/2 == 100

                              2
Out[2]= 10 t + 16.0925 t  == 100
```

Now both the equations are immediately solved, of course, though there is an important distinction between them that we point out below.

```
In[3]:= Solve[eq1,t]

                  199
Out[3]= {{t -> ---}}
                  132

In[4]:= Solve[eq2,t]

Out[4]= {{t -> 2.20139}, {t -> -2.8228}}
```

Now, the difference between the first and the second answer is not only that the second answer in inexact, but also that we seem to have two answers in the second case.

The correct decision here is, of course, that the second value (-2.8228) is extraneous and apparently physically meaningless. So we retain the first value and we discard the second one.

We should note however for the future that such a situation is characteristic of nonlinear relations (leading to systems of nonlinear equations).

For those who are not satisfied with how we dealt with the extraneous solution I would confess that actually it is not totally meaningless because $t = 2.8228$ is the answer in the following problem:

> An object is at the height of 100 ft and is tossed up with the speed of 10 ft/sec. How soon will it fall to the ground?

Inquisitive minds may give more thought to this puzzling fact.

We can, however, go for generalizations and ask questions like the following:

> John started at noon from Thiscity to Thatcity at a speed of *u* mph, while Jim started from Thatcity to Thiscity at *X* pm at a speed of *v* mph. How soon will they meet provided they do not alter their speeds and the distance between the cities is known to be *r* miles.

For this question we get an equation with all symbolic parameters:

```
In[5]:= eq3 = u t + v (t - X) == r

Out[5]= t u + v (t - X) == r
```

And *Mathematica*'s `Solve` gives a fairly general formula as the answer:

```
In[6]:= Solve[eq3,t]

                 -r - v X
Out[6]= {{t -> -(--------)}}
                  u + v
```

In similar manner we may ask:

> An object is released at height *h* with the vertical component of velocity *v*. In what time will it reach the ground?

We get a general symbolic equation to solve the problem, where g is the natural gravitational acceleration.

```
In[7]:= eq4 = v t + g t^2 /2  == h

           2
        g t
Out[7]= ---- + t v == h
         2

In[8]:= Solve[eq4,t]

                                      2
               -2 v   2 Sqrt[2 g h + v ]
               ---- + ------------------
                g             g
Out[8]= {{t -> -------------------------},
                           2

                                2
        -2 v   2 Sqrt[2 g h + v ]
        ---- - ------------------
         g             g
  {t -> -------------------------}}
                    2
```

```
In[9]:= Simplify[%]

                                    2
                -v + Sqrt[2 g h + v ]
Out[9]= {{t -> ---------------------},
                          g

                              2
             v + Sqrt[2 g h + v ]
  {t -> -(--------------------)}}
                      g
```

At this time we are tempted to discard the second root altogether and we may well do that provided we will be careful afterwards in selecting the signs of `v` and `g`. Thus we have

```
In[10]:= %[[1]]

                                    2
                -v + Sqrt[2 g h + v ]
Out[10]= {t -> ---------------------}
                          g
```

and an additional advantage of this formula is that it is valid not only for the Earth but also for the Moon and other planets when the value of `g` is properly selected. This is good news.

The bad news is that it is technically impossible to make `Solve` automatically discriminate between the two solutions. Conceivably, we could make the formal assumption that $v > 0$ and require that negative values for time be discarded. Unfortunately at present it is technically impossible to do that in *Mathematica*'s `Solve`. Moreover *all the solving algorithms in Mathematica tend to treat all the involved variables as complex variables*.

Therefore, `Solve` is no magic box (especially in nonlinear cases) and further inspection and analysis should be applied to the answers it returns.

Let us look now into how *Mathematica* treats differential equations.

As a good starting example recall the description of the small-amplitude motion of a pendulum.

Assuming that `t` is time and `y[t]` is the pendulum displacement from equilibrium (measured in any appropriate way), we then have the basic law of its movement in the for:

```
In[11]:= eq5 = y''[t]==-k^2 y[t]

                         2
Out[11]= y''[t] == -(k  y[t])
```

where `k` is the inertia coefficient depending on the choice of the y coordinate.

This is a math model of the pendulum and diverse questions may be asked about it varying with the person asking the question, but the overall behavior of the system should be of general interest, *i.e.*, not a single numeric value but the whole function `y[t]` is to be described. This is exactly what the theory of differential equations is about and this is exactly what the `DSolve` function of *Mathematica* is trying to do.

```
In[12]:= DSolve[eq5,y[t],t]

                      -I k t          I k t
Out[12]= {{y[t] -> E       C[1] + E      C[2]}}
```

This is a very typical and important answer so let us give it full attention.

First of all, we see that we have what seems to be a complex function despite of the fact that the motion of the pendulum is completely real and there is nothing "imaginary" about it. Also important is the presence of integration constants `C[1]` and `C[2]` in the answer. These are arbitrary constants generated by `DSolve`, and so instead of a specific solution we get a two-parameter family of solutions for the equation. This seems to indicate that the equation in itself makes a too general description of the pendulum motion (and is in a sense underdetermined).

This is indeed true because information about a specific motion of a pendulum requires knowledge of its initial position and its initial speed. Mathematically speaking, knowledge of $y[0]$ and $y'[0]$ is required. Let us play with it for a while.

Assume that the pendulum is released with the initial displacement of 1/2 and zero initial speed:

```
In[13]:= ans = DSolve[{eq5,y[0]==1/2,y'[0]==0},y[t],t]

                -I k t    I k t
               E         E
Out[13]= {{y[t] -> ------- + ------}}
                  4         4
```

Though this is a specific function of `t`, the representation is a bit unfortunate.

Since there are no built-in simplifications providing for the Euler's formula, let us write out an appropriate function for that:

```
In[14]:= R[expr_]:= Simplify[expr/.{Exp[Complex[0,a_] x_]:>
         (Cos[a x] + I Sin[a x])}]

In[15]:= R[ans]

                   Cos[k t]
Out[15]= {{y[t] -> --------}}
                      2
```

So in this case we eventually get a real-valued description of the motion we were interested in, and the description is found in terms of elementary function `Cos`.

Generally, however, we may well be getting special functions for an answer as, for instance, in the following example:

```
In[16]:= DSolve[y''[t] == t y[t] , y[t] , t ]

Out[16]= {{y[t] -> AiryBi[t] C[1] + AiryAi[t] C[2]}}
```

Here the `AiryAi` and `AiryBi` are special functions serving as fundamental solutions of the equation `y''[t]==t  y'[t]`. These are in no way elementary functions and it may seem at the first glance that we gain nothing from such an answer. But, in fact, we do, because there is elaborate built-in code supporting these special functions, and so we know everything we need to know about them.

For instance, we are able to differentiate them:

```
In[17]:= D[AiryAi[t],t]

Out[17]= AiryAiPrime[t]

In[18]:= D[AiryAiPrime[t],t]

Out[18]= t AiryAi[t]
```

We are, of course, able to compute them in any given point:

```
In[19]:= AiryBi[0]

                   1
Out[19]= -------------------
          1/6         2
         3    Gamma[-]
                      3

In[20]:= AiryBi Prime[0]

            1/6
           3
Out[20]= --------
               1
         Gamma[-]
               3
```

This enables one to do the initial value problems like the following:

```
In[21]:= DSolve[{y''[t]==t y[t],y[0]==1,y'[0]==0},y[t],t]

Out[21]= {{y[t] ->

     1/6                                      2
    3    (Sqrt[3] AiryAi[t] + AiryBi[t]) Gamma[-]
                                              3
    ----------------------------------------------}}
                          2
```

There are additional phenomena to be accounted for when it comes to *nonlinear* differential equations.

Consider, for example, the following equation of a nonlinear oscillator:

```
In[22]:= eq6 = y''[t] + a y'[t]^2 + b y[t] == 0

                              2
Out[22]= b y[t] + a y'[t]  + y''[t] == 0
```

Assume zero-mass case first:

```
In[23]:= eq7 = %/.b:>0

                       2
Out[23]= a y'[t]  + y''[t] == 0
```

Then, looking for the general solution, we get

```
In[24]:= <<Calculus`DSolve`
```

This loads the package dealing with the nonlinear equations:

```
In[25]:= DSolve[eq7,y[t],t]

Union::heads:
   Heads Pattern and List at positions 2 and 1
     are expected to be the same.

                                Log[-(a t) + C[1]]
Out[25]= {{y[t] -> C[2] + ------------------}}
                                        a
```

However, for nonzero mass (b nonzero), we have the following typically nonlinear thing:

```
In[26]:= DSolve[eq6,y[t],t]

                                                     1
Out[26]= {-(Sqrt[2] Integrate[------------------------------------,
                                          b       4 C[1]       2 b y[t]
                                    Sqrt[-- + ---------- - --------]
                                           2    2 a y[t]          a
                                          a     E

       y[t]]) == t + C[2],

                                         1
   Sqrt[2] Integrate[------------------------------------, y[t]] ==
                               b       4 C[1]       2 b y[t]
                         Sqrt[-- + ---------- - --------]
                                2    2 a y[t]          a
                               a     E

    t + C[2]}
```

From the mathematical viewpoint we have an answer in quadratures and the solutions seem to have at least two branches. From the physical viewpoint both branches are pretty much the same because they differ only in the direction of the time-axis. So we might as well pick up the second branch:

```
In[27]:= %[[2]]

                                           1
Out[27]= Sqrt[2] Integrate[-------------------------------,
                                b     4 C[1]     2 b y[t]
                           Sqrt[-- + -------- - --------]
                                 2   2 a y[t]       a
                                a    E

    y[t]] == t + C[2]
```

This gives the solution as implicit functions of `t`. The integral on the left-hand side of this relation is not computed symbolically to any suitable closed form, but even if it were it would produce an equation with special functions and it would still be impossible to solve for `y[t]` in closed form.

Despite this the formula produced by `DSolve` is extremely useful because it provides for easier numerical simulation of `y[t]`. But an even more obvious advantage of this formula is that it gives a direct answer to the question "at what moment of time does the oscillator reach the given position $y1$?"

Thus we should agree that *Mathematica*'s solving features are not to be treated lightly.

While `Solve` and `DSolve` relieve the user of a great amount of routine work and help to avoid errors while doing problems symbolically, the answers given need proper understanding sometimes in a wider and informal context.

In what follows we discuss the respective symbolic solvers of *Mathematica* in more detail and with more examples.

3 Systems of Linear Algebraic Equations

■ 3.1 Equations with numeric coefficients

Mathematica's `Solve` gives clean and exact solutions to well-determined linear systems with exact numeric coefficients:

```
In[28]:= sys1= {x+2y+3z==12, 4x+5y+6z==13, 7x+8y+11z==14 }

Out[28]= {x + 2 y + 3 z == 12, 4 x + 5 y + 6 z == 13,
  7 x + 8 y + 11 z == 14}

In[29]:= Solve[sys1,{x,y,z}]

                  34         35
Out[29]= {{x -> -(--), y -> --, z -> 0}}
                  3          3
```

If the system happens to be singular or underdetermined, `Solve` does the best possible reductions, choosing some unknowns as principal and expressing those via the remaining ("free") unknowns:

```
In[30]:= sys2= {x+2y+3z==12, 4x+5y+6z==13, 7x+8y+9z==14}

Out[30]= {x + 2 y + 3 z == 12, 4 x + 5 y + 6 z == 13,
  7 x + 8 y + 9 z == 14}

In[31]:= Solve[sys2,{x,y,z}]

                  34             35
Out[31]= {{x -> -(--) + z, y -> -- - 2 z}}
                  3              3
```

Here, because the third equation of the `sys2` is a linear combination of the first and the second, the rank of the system is actually 2, and `Solve` has automatically chosen `x` and `y` as principal unknowns and `z` as the free unknown.

`Solve` returns an empty list when the system appears to be incompatible:

```
In[32]:= sys3= {x+2y+3z==12, 4x+5y+6z==13, 7x+8y+9z==15}

Out[32]= {x + 2 y + 3 z == 12, 4 x + 5 y + 6 z == 13,
  7 x + 8 y + 9 z == 15}

In[33]:= Solve[sys3,{x,y,z}]

Out[33]= {}
```

Mathematica is capable of solving fairly large systems:

```
In[34]:= sys4= {-7 x0 - 8 x1 - 4 x2 + x3 - 4 x4 + 9 x5 + 2 x6 -
         8 x7 - 4 x9 == 19,
         -3 x0 + 3 x1 - 4 x2 + 5 x4 - 4 x6 - 5 x7 + 9 x9 == -2,
         10 x0 - 7 x1 + x2 - 9 x4 - x5 + 5 x6 + x7 - 8 x8 -
         10 x9 == -27,
         -7 x0 + 6 x1 + 3 x2 + 4 x3 + 9 x4 - 2 x5 - 9 x6 - 8 x7 +
          5 x8 - 7 x9 == 123,
          -6 x0 + 5 x1 + 6 x2 - 5 x3 + 7 x4 + 7 x5 - 10 x6 + 9 x7
         -
          2 x8 - 3 x9 == 75,
          8 x0 + 2 x1 - 8 x2 + 10 x3 + 8 x4 + 10 x5 - 9 x6 +
          8 x7 + 5 x8 + 4 x9 == -66,
          10 x0 + 5 x1 + 2 x2 - 2 x3 - 6 x4 + 9 x5 - 10 x6 + 3 x7
         +
          2 x8 - 3 x9 == -24,
          -9 x0 - 2 x1 + 10 x2 - 10 x3 - x4 + 8 x5 - 7 x6 - 6 x7
         +
          4 x8 +  5 x9 == 33,
          -4 x0 - 9 x1 + x2 + 4 x3 + 9 x4 + 8 x5 - 7 x6 + x7 +
          10 x8 - 8 x9 == 11,
          -5 x0 - 9 x1 - x2 - 9 x3 + x4 + 8 x5 - 3 x6 - 7 x7 -
          9 x8 == 13}

Out[34]= {-7 x0 - 8 x1 - 4 x2 + x3 - 4 x4 + 9 x5 + 2 x6 -
    8 x7 - 4 x9 == 19, -3 x0 + 3 x1 - 4 x2 + 5 x4 - 4 x6 -
    5 x7 + 9 x9 == -2, 10 x0 - 7 x1 + x2 - 9 x4 - x5 + 5 x6 +
    x7 - 8 x8 - 10 x9 == -27,
  -7 x0 + 6 x1 + 3 x2 + 4 x3 + 9 x4 - 2 x5 - 9 x6 - 8 x7 +
    5 x8 - 7 x9 == 123, -6 x0 + 5 x1 + 6 x2 - 5 x3 + 7 x4 +
    7 x5 - 10 x6 + 9 x7 - 2 x8 - 3 x9 == 75,
  8 x0 + 2 x1 - 8 x2 + 10 x3 + 8 x4 + 10 x5 - 9 x6 + 8 x7 +
    5 x8 + 4 x9 == -66, 10 x0 + 5 x1 + 2 x2 - 2 x3 - 6 x4 +
    9 x5 - 10 x6 + 3 x7 + 2 x8 - 3 x9 == -24,
  -9 x0 - 2 x1 + 10 x2 - 10 x3 - x4 + 8 x5 - 7 x6 - 6 x7 +
    4 x8 + 5 x9 == 33, -4 x0 - 9 x1 + x2 + 4 x3 + 9 x4 + 8 x5 -
    7 x6 + x7 + 10 x8 - 8 x9 == 11,
  -5 x0 - 9 x1 - x2 - 9 x3 + x4 + 8 x5 - 3 x6 - 7 x7 - 9 x8 ==
   13}

In[35]:= Solve[sys4,{x1,x2,x3,x4,x5,x6,x7,x8,x9,x0}]

Out[35]= {{x1 -> 4, x2 -> 3, x3 -> 2, x4 -> 1, x5 -> 0,
   x6 -> -1, x7 -> -2, x8 -> -3, x9 -> -4, x0 -> -5}}
```

Likewise, equations with floating-point coefficients are solved.

Here is a short listing of the previous input.

```
In[36]:= sys5=
           {-2.284955953399043 + 0.98072083788745*a +
              0.5149245054089994*b +
              0.01876256215723172*c + 0.4134333648162499*d +
              0.88092588739555*e +
              0.1495961656835541*f + 0.5909832691302113*g +
              0.3025510368490691*h +
              0.98357147991662*i + 0.2982400077523538*j +
              0.01132919047002376*k +
              0.04133740323021599*l == 0,
            -2.360869998398285 + 0.588228939428387*a +
              0.602143114881585*b +
              0.4497057991799297*c + 0.98734026529388*d +
              0.4578194070416024*e +
              0.4195505691950744*f + 0.3405819682644322*g +
              0.2979974108960765*h +
              0.6868184243804951*i + 0.4644425209370464*j +
              0.1843755485393622*k +
              0.2428631627777372*l == 0,

                                 .
                                 .
                                 .

            -1.744305325507481 + 0.4339522262147568*a +
              0.6195479108174573*b +
              0.3133158557788148*c + 0.006718835617960813*d +
              0.02783245354744254*e +
              0.01014408441395737*f + 0.3835946363003488*g +
              0.1424359028138075*h +
              0.7363152578418685*i + 0.0544006883495025*j +
              0.3273749586365076*k +
              0.3993443502097187*l == 0};
In[37]:= Solve[sys5,{a,b,c,d,e,f,g,h,i,j,k,l}]//Timing

Out[37]= {1.85 Second, {{a -> 0.405433, b -> 0.285194,
    c -> 0.896831, d -> 0.0917717, e -> 0.3776, f -> 0.27505,
    g -> 0.513236, h -> 0.949336, i -> 0.641285, j -> 0.22065,
    k -> 0.185861, l -> 0.549991}}}
```

In solving floating-point equations one can also play with the precision at which all the computations are done. Be prepared, however, for a slowdown when precision becomes significantly greater than the machine precision on the computer you are using:

```
In[38]:= Precision[%]

Out[38]= 16

In[39]:= SetPrecision[sys5,32];
```

```
In[40]:= Solve[%,{a,b,c,d,e,f,g,h,i,j,k,l}]//Timing

Out[40]= {3.08333 Second,
  {{a -> 0.405432691520548755498965 6,
    b -> 0.2851943302212723951296390 5,
    c -> 0.8968307475074912089144453 6,
    d -> 0.0917717011976634317585727 4,
    e -> 0.3776002379731078905243535 6,
    f -> 0.2750502458073153019351651 1,
    g -> 0.5132361112071398927912267 8,
    h -> 0.94933579838385669286420948 9,
    i -> 0.6412849801312379063781839 9,
    j -> 0.22064955745781021019306045 5,
    k -> 0.18586115257063220343670520 6,
    l -> 0.54999144817413592435116166 1}}}
```

3.2 Equations with symbolic coefficients

No case study is performed while solving systems of linear equations symbolically. In other words, if an expression to divide by does not simplify to be identically zero, it is considered to be a non-zero.

```
In[41]:= sys6={ a x + b y == c , d x + e y == f }

Out[41]= {a x + b y == c, d x + e y == f}

In[42]:= Solve[sys6,{x,y}]

                   -(c (-(b d) + a e)) - b (c d - a f)
Out[42]= {{x -> -(-----------------------------------),
                           a (-(b d) + a e)

             c d - a f
    y -> -(------------)}}
           -(b d) + a e

In[43]:= Simplify[%]

                c e - b f        -(c d) + a f
Out[43]= {{x -> ------------, y -> ------------}}
               -(b d) + a e      -(b d) + a e
```

If, however, an exact reduction of an equation with respect to others does take place, then it is accounted for as in the following examples:

```
In[44]:= sys7={a x + b y + c z == d , b x + a y + e z == f ,
           (a+b)(x+y)+(c+e)z == (d+f)}

Out[44]= {a x + b y + c z == d, b x + a y + e z == f,
  (a + b) (x + y) + (c + e) z == d + f}

In[45]:= Solve[sys7,{x,y,z}]

Out[45]= {{x ->

         2    2
     -((a  - b ) d) - b (b d - a f)
  -(------------------------------) -
                2    2
           a (a  - b )

        2    2
    ((a  - b ) c - b (-(b c) + a e)) z
    ----------------------------------,
                 2    2
            a (a  - b )

            b d - a f     (-(b c) + a e) z
    y -> -(---------) - ----------------}}
             2    2           2    2
            a  - b           a  - b
```

```
In[46]:= Simplify[%]

                  -(a d) + b f + a c z - b e z
Out[46]= {{x -> ----------------------------,
                            2    2
                          -a  + b

       b d - a f - b c z + a e z
  y -> -------------------------}}
                  2    2
                -a  + b
```

(This happened because the third equation of the `sys7` has been the sum of the two preceding equations, so that the system is in fact underdetermined. As was already explained in the previous subsection, in such a case `Solve` would select principal variables equal in number to the rank of the system and then express those in terms of the remaining "free" variables.)

One more example of the successful reduction is

```
In[47]:= sys8 = {a x + b y + c z == 1, b x + a y + e z == 2 ,
            (a+b)(x+y)+(c+e)z == g }

Out[47]= {a x + b y + c z == 1, b x + a y + e z == 2,
  (a + b) (x + y) + (c + e) z == g}

In[48]:= Solve[sys8,{x,y,z}]

Out[48]= {}
```

Let us give this example the attention it deserves.

The system, `sys8`, is compatible if and only if `g===3`, so it is incompatible for the generic `g` and this is exactly what `Solve` shows.

As alternative is the `Reduce` function of *Mathematica* that does perform the case study after all:

```
In[49]:= Reduce[sys8,{x,y,z}]

Out[49]= (a - b) (a + b) != 0 && g == 3 &&

       a - 2 b - a c z + b e z           2 a - b + b c z - a e z
  x == ----------------------- && y == -----------------------\
                2    2                          2    2
               a  - b                          a  - b

                                            2     2
   || b (c - e) (c + e) != 0 && g == 3 && a  == b  &&

                            3            2  2              3
            2          2 a c  e         a c  e        2 a c e
  x == (2 c  - c e + ---------- - ---------- - ---------- +
                        2    2         2    2       2    2
                     b (c  - e )   b (c  - e )   b (c  - e )

          4
       a e            2        2            2    2
     ---------- - a c  y + a e  y) / (b (c  - e )) &&
        2    2
     b (c  - e )

       -2 a c + b c + a e - 2 b e
  z == -------------------------- ||
                  2    2
               b (c  - e )

 2   3                    3          4
b  c e  != 0 && (2 c - e) e  != 0 && b e  != 0 && g == 3 &&

  2     2                                             2     2
 c  == e  && a e == -(b c) && a c == -(b e) && a  == b  &&

       -c + 2 e + 2 b c y         c + 2 e
  x == ------------------ && z == ------- ||
             2 b e                    2
                                   2 e

                                                              2
 e != 0 && g == 3 && 2 c == e && b == 0 && a == 0 && z == -
                                                              e
```

Those who do not find this answer helpful may get a simpler one by appending suitable nondegeneracy assumptions to the system `sys8`. For example, the leading minor of the system is $a^2 - b^2$ and we may wish it to be nonzero:

```
In[50]:= Reduce[Append[sys8,a^2-b^2!=0],{x,y,z}]

Out[50]= (a - b) (a + b) != 0 && g == 3 &&

      a - 2 b - a c z + b e z            2 a - b + b c z - a e z
 x == ----------------------- && y == -----------------------
               2    2                          2    2
              a  - b                          a  - b
```

Now the logic of the answer is quite clear and we clearly should consider the `g == 3` clause as the compatibility condition derived automatically by the `Reduce`.

3.3 Eigenvalue problems

There are lots of applications with systems of linear algebraic equations where the basic request is not to find solutions but rather to find eigenvalues of the corresponding matrices. Recall briefly that k is an eigenvalue of the square matrix A if for some vector v (called eigenvector) `A v === k v`.

So in this setting not only v but also k is to be found. It turns out that for the last system to be compatible with the respect to v, the unknown eigenvalue k must satisfy a certain polynomial equation (called "characteristic equation").

Let us consider an example:

```
In[51]:= A= {{19/6, -8/3, 1}, {23/3, -25/6, 1}, {41/3, -26/3,
         5/2}}

            19    8         23    25         41    26   5
Out[51]= {{--, -(-), 1}, {--, -(--), 1}, {--, -(--), -}}
            6     3          3     6          3     3   2

In[52]:= Eigenvalues[A]

             1   3  1
Out[52]= {-(-), -, -}
             2   2  2
```

Another function will provide the eigenvectors

```
In[53]:= Eigenvectors[A]

Out[53]= {{1, 4, 7}, {2, 5, 8}, {3, 6, 11}}
```

At last, the `Eigensystem` function will give both:

```
In[54]:= Eigensystem[A]

              1   1  3
Out[54]= {{-(-), -, -}, {{1, 4, 7}, {2, 5, 8}, {3, 6, 11}}}
              2   2  2
```

It should be understood, however, that the eigenvalues are roots of the high-order characteristic polynomial and therefore sometimes the output will be very technical and not easy to read. For example:

```
In[55]:= B={{1,2,3},{4,5,6},{7,8,11}}

Out[55]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 11}}
```

```
In[56]:= Eigenvalues[B]

           17                307
Out[56]= {-- + ------------------------------ +
           3                           1/3
               3 (5291 + 9 I Sqrt[11602])

                               1/3
   (5291 + 9 I Sqrt[11602])
   --------------------------,
                3

  17   I                    -307
  -- + - Sqrt[3] (------------------------------ +
  3    2                                    1/3
                  3 (5291 + 9 I Sqrt[11602])

                                  1/3
      (5291 + 9 I Sqrt[11602])
      ---------------------------) -
                   3

                                                                1/3
                 307                  (5291 + 9 I Sqrt[11602])
   ----------------------------- + ----------------------------
                             1/3                 3
   3 (5291 + 9 I Sqrt[11602])
   ------------------------------------------------------------,
                                2

  17   I                    -307
  -- - - Sqrt[3] (------------------------------ +
  3    2                                    1/3
                  3 (5291 + 9 I Sqrt[11602])

                                  1/3
      (5291 + 9 I Sqrt[11602])
      ---------------------------) -
                   3

                                                                1/3
                 307                  (5291 + 9 I Sqrt[11602])
   ----------------------------- + ----------------------------
                             1/3                 3
   3 (5291 + 9 I Sqrt[11602])
   ------------------------------------------------------------}
                                2
```

Such an output may be made more manageable, if floating-point numerics and round-offs are permitted. The `N` function of *Mathematica* does it:

```
In[57]:= N[%]

                                     -16
Out[57]= {17.3263 - 1.11022 10      I,

                             -15                          -15
  -0.77382 - 2.25204 10     I, 0.447513 + 2.36307 10     I}
```

(This way you may appreciate the fact that the eigenvalues are almost real and even try to discard the imaginary parts within the tolerance of 10^{-15} if your application admits it.)

One should be even more cautious with the symbolic matrices:

```
In[58]:= F={{a,b,c},{d,e,f},{g,h,k}}

Out[58]= {{a, b, c}, {d, e, f}, {g, h, k}}
```

```
In[59]:= Short[Eigenvalues[F], 10]

Out[59]//Short=
  -(-a - e - k)
 {------------- + <<1>> + 
        3

            3                2                    2      3
   Power[2 a  + 9 a b d - 3 a  e + 9 b d e - 3 a e  + 2 e  +

                    2      3
      <<13>> - 3 e k  + 2 k  +

       3/2         2  2  2      3  3       3
      3    Sqrt[-(a  b  d  ) - 4 b  d  + 2 a  b d e +

                  2  2      4  2       2      2  2  2
         10 a b  d  e - a  e  - 8 a  b d e  - b  d  e  +

            3  3            3    2  4      2
         2 a  e  + 2 a b d e  - a  e  - 2 a  b c d g -

             2    2        3
         12 b  c d  g - 2 a  c e g + 2 a b c d e g -

            2    2               2            3          4
         2 a  c e  g - 20 b c d e  g + 8 a c e  g - 4 c e  g +

                                   3            3            3
         <<117>> - 2 c e g k  + 4 b f g k  + 4 c d h k  -

                  3            3    2  4          4
         2 a f h k  + 2 e f h k  - a  k  - 4 b d k  +

                4    2  4    1        1/3
         2 a e k  - e  k  ], -] / (3 2   ), <<1>>,
                             3

  -<<1>>
  ------ + <<2>>}
    3
```

which is not very helpful.

Of course, this does not mean that symbolic eigenvalues will not be useful at all. For instance, they work reasonably in the following example:

```
In[60]:= {{1,1/E^t},{-E^(-2*t),0}}

               -t       -2 t
Out[60]= {{1, E  }, {-E    , 0}}

In[61]:= Eigensystem[%]

            (3 t)/2              3 t
           E        - Sqrt[-4 + E   ]
Out[61]= {{---------------------------, 
                      (3 t)/2
                   2 E

    (3 t)/2              3 t
   E        + Sqrt[-4 + E   ]
   ---------------------------},
              (3 t)/2
           2 E

                 t/2
              -2 E
  {{---------------------------, 1}, 
      (3 t)/2              3 t
     E        + Sqrt[-4 + E   ]

                 t/2
              -2 E
   {---------------------------, 1}}}
      (3 t)/2              3 t
     E        - Sqrt[-4 + E   ]
```

4 Systems of Nonlinear Equations

4.1 Finite type equations

When a system of algebraic equations has only a finite number of solutions it is called a finite-type system.

Scalar polynomial equations are definitely finite-type systems. As is well known, an n-degree polynomial has exactly n roots (if counted with their multiplicities) in the complex domain. Let us recall once again that all the *Mathematica* solvers work over the complex numbers.

For example, we have :

```
In[62]:= Solve[x^2+x+1==0,x]

                 -1 + I Sqrt[3]         -1 - I Sqrt[3]
Out[62]= {{x -> ---------------}, {x -> ---------------}}
                       2                       2
```

where I is the imaginary root of (-1).

Note the way `Solve` does the multiplicities:

```
In[63]:= Solve[x^3-x^2-x+1==0,x]

Out[63]= {{x -> 1}, {x -> 1}, {x -> -1}}
```

Here the `x->1` solution is repeated twice because 1 is a root of multiplicity 2.

`Solve` does reasonably well for equations involving symbolic parameters:

```
In[64]:= Solve[ a x^2 + b x + c == 0 , x ]

                                 2
                  b     Sqrt[b  - 4 a c]
                -(-) + -----------------
                  a            a
Out[64]= {{x -> -------------------------},
                            2

                         2
          b     Sqrt[b  - 4 a c]
        -(-) - -----------------
          a            a
   {x -> -------------------------}}
                    2

In[65]:= Simplify[%]

                              2
                 -b + Sqrt[b  - 4 a c]
Out[65]= {{x -> ----------------------},
                          2 a

                         2
         -(b + Sqrt[b  - 4 a c])
   {x -> ------------------------}}
                   2 a
```

```
In[66]:= Short[Solve[ a x^3 + b x^2 + c x + d ==0 , x], 10]

Out[66]//Short=
        -b      1/3    2
 {{x -> --- - (2    (-b  + 3 a c)) /
        3 a

                     3                 2
      (3 a Power[-2 b  + 9 a b c - 27 a  d +

           3/2          2 2         3      3
          3   a Sqrt[-(b  c ) + 4 a c  + 4 b  d - 18 a b c d +

                 2  2   1
             27 a  d ], -]) +
                        3

               3                 2
     Power[-2 b  + 9 a b c - 27 a  d +

         3/2          2 2         3      3
        3   a Sqrt[-(b  c ) + 4 a c  + 4 b  d - 18 a b c d +

               2  2   1        1/3             -b
           27 a  d ], -] / (3 2    a)}, {x -> --- + <<2>>},
                      3                        3 a

        -b                I
  {x -> --- + <<1>> - - Sqrt[3]
        3 a               2

         1/3    2
      ((2    (-b  + 3 a c)) /

                        3                 2
         (3 a Power[-2 b  + 9 a b c - 27 a  d +

              3/2          2 2         3      3
             3   a Sqrt[-(b  c ) + 4 a c  + 4 b  d -

                                 2  2   1
                18 a b c d + 27 a  d ], -]) + <<1>>)}}
                                        3
```

It does well, except that there is no formula for a generic equation of the order greater than 4, and even for the 4th order the corresponding formula is not at all handy.

It does not mean that `Solve` does not do higher order equations, those are done whenever factoring into low-order factors is possible. For instance:

```
In[67]:= Solve[x^7+x^6+x^5+x^4+x^3+x^2+x+1==0,x]

                                                    1/4
Out[67]= {{x -> -1}, {x -> I}, {x -> -I}, {x -> (-1)   },

             3/4              5/4              7/4
  {x -> (-1)   }, {x -> (-1)   }, {x -> (-1)   }}
```

Factoring helps whenever applicable also when symbolic parameters are present:

```
In[68]:=   Solve[1 - a + 2*x - a*x + 2*x^2 - a*x^2 + 2*x^3 -
        a*x^3 +
          2*x^4 - a*x^4 + 2*x^5 -  a*x^5 + x^6 == 0,x]

                                          1 + I Sqrt[3]
Out[68]= {{x -> -1 + a}, {x -> -1}, {x -> -------------},
                                                2

        1 - I Sqrt[3]         -1 + I Sqrt[3]
  {x -> -------------}, {x -> --------------},
              2                     2

        -1 - I Sqrt[3]
  {x -> --------------}}
              2
```

Systems of finite type are recognized and dealt with whenever possible by the `Solve` command. The following example seems to be a rather sophisticated system of equations.

```
In[69]:= s = {-1 + x^2 + y + x*y^2 == 0, 1 + x + x^2*y == 0}

                    2              2                2
Out[69]= {-1 + x  + y + x y  == 0, 1 + x + x  y == 0}
```

But then

```
In[70]:= Short[Solve[s,{x,y}], 10]

Out[70]//Short=
 {{y -> 0, x -> -1}, {y ->

          3/2                     1/6
    ((2 3    (9 + I Sqrt[687])    ) /

                    1/3                       1/3
        Sqrt[73728    + 3 (9 + I Sqrt[687])    +

            1/3                     2/3
          96    (9 + I Sqrt[687])    ] + <<9>>) / 64,

          1
   x -> - + <<2>>}, {y ->
          4

          3/2                     1/6
    ((2 3    (9 + I Sqrt[687])    ) /

                    1/3                       1/3
        Sqrt[73728    + 3 (9 + I Sqrt[687])    +

            1/3                     2/3
          96    (9 + I Sqrt[687])    ] + <<8>> -

            2                  3/2
       Sqrt[-] <<1>> Sqrt[3     Sqrt[9 + I Sqrt[687]] + <<3>>]
            3
       ---------------------------------------------------------) \
                          Sqrt[9 + I Sqrt[687]]

      / 64, <<1>>}, <<1>>,

                 3/2                     1/6
  {y -> ((-2 3    (9 + I Sqrt[687])    ) /

                    1/3                       1/3
        Sqrt[73728    + 3 (9 + I Sqrt[687])    +

            1/3                     2/3
          96    (9 + I Sqrt[687])    ] + <<9>>) / 64,

          1
   x -> - + <<2>>}}
          4
```

which doesn't seem very helpful at a first glance.

But we can make it more manageable by looking at the numeric approximations:

```
In[71]:= N[%]

Out[71]= {{y -> 0, x -> -1.},
  {y -> -1.07095 - 1.05537 I, x -> 1.01891 - 0.602565 I},
  {y -> -1.07095 + 1.05537 I, x -> 1.01891 + 0.602565 I},
  {y -> 1.07095 + 0.424335 I, x -> -0.518913 - 0.66661 I},
  {y -> 1.07095 - 0.424335 I, x -> -0.518913 + 0.66661 I}}
```

We do see that the system has exactly five solutions. One of those is an obvious real-valued solution, while the others are nontrivial complex solutions.

To understand what is possible and what is impossible here, we should get acquainted with the inner function that preprocesses systems of polynomial equations before they get solved. The function is `GroebnerBasis` and it rewrites the system into an equivalent minimal form with all the compatibility conditions checked and equations prepared for subsequent resolution. Thus for the above example we have

```
In[72]:= GroebnerBasis[{-1 + x^2 + y + x*y^2 , 1 + x + x^2*y },
           {x,y}]

            5                2                4
Out[72]= {y  - y (-3 + 2 y + y ), 2 + 2 x - y + y }
```

which means that

```
In[73]:= s1=Map[(#==0)&,%29]

                 34         35
Out[73]= {{x -> -(--), y -> --, z -> 0} == 0}
                 3          3
```

is the system equivalent to the original one.

From the form of the equations it is evident that there is only one value of x corresponding to each value of y; therefore the system has exactly five solutions.

The `GroebnerBasis` procedure is run internally by `Solve` for each system of polynomial equations and theoretically the procedure provides for deciding whether the system is compatible and if compatible, whether it has only a finite number of solutions. However, in practice the computational complexity of this procedure grows with the number of unknowns and also at least exponentially with the degree of equations.

Therefore multivariate systems of higher degree are not likely to be solved or will take an enormous amount of time to be solved.

Slow as it is, the Groebner basis approach is commonly acknowledged as the most up-to-date technique for dealing with systems of polynomial equations symbolically. So there is not much choice here. Implementation of the approach is, however, what crucially matters and we expect major improvements with the next releases of *Mathematica*.

■ 4.2 Nonpolynomial and transcendental equations

Occasionally the `Solve` procedure will be able to solve the equations involving elementary functions other than polynomials. However, this ability is not to be overestimated, because generically such equations are not algorithmically solvable and also for additional reasons as described below.

Let us start with an important example:

```
In[74]:= Solve[Sqrt[x+1]+Sqrt[x]==2,x]

                9
Out[74]= {{x -> --}}
                16
```

Since almost all the nontrivial elementary functions are multivalued and and are defined in domains containing exceptional points or curves a double check is always needed to make sure the solutions we are getting are not extraneous.

```
In[75]:= (Sqrt[x+1]+Sqrt[x]==2)/.%20

ReplaceAll::reps:
      1/6
     3
   {--------} is neither a list of replacement rules nor a valid
        1
    Gamma[-]
        3
     dispatch table, and so cannot be used for replacing.

                                              1/6
                                             3
Out[75]= Sqrt[x] + Sqrt[1 + x] == 2 /. --------
                                                1
                                            Gamma[-]
                                                3
```

This is OK but let us consider a generic version with a parameter

```
In[76]:= eq=Sqrt[x+1]+Sqrt[x]==a

Out[76]= Sqrt[x] + Sqrt[1 + x] == a
```

```
In[77]:= Solve[eq,x]

                        2    4
                 -(-1 + 2 a  - a )
Out[77]= {{x -> -----------------}}
                          2
                       4 a
```

This is some closed formula but the first suspicious thing is that no qualifications are made for *a* while it is evident that the equation should behave differently for different *a*'s. Let us double check:

```
In[78]:= eq/.%

                          2    4                        2    4
                  -(-1 + 2 a  - a )             -1 + 2 a  - a
Out[78]= {Sqrt[-----------------] + Sqrt[1 - --------------] ==
                          2                             2
                       4 a                           4 a

   a}
```

It is far from the evident that this equality holds for every *a* and in fact it doesn't!

```
In[79]:= %/.a:>(-1)

Out[79]= {False}
```

So at least for some *a*'s the above general solution is spurious.

Before blaming *Mathematica* let us ask whether there is any general algorithmic way to assess the solvability of such an equation. The answer is no, especially since we are working in the complex domain.

So the conclusion is: while granting *Mathematica* the proficiency of handling large complicated calculations properly we must take additional care of mathematical correctness when nonpolynomial expressions are present and always double check the answers for the nonpolynomial equations.

We have to be even more conservative with the equations involving nonalgebraic functions such as exponentials, logarithms and trigonometric functions.

Let us play with **logarithms** to see what happens here:

```
In[80]:= Solve[Log[y]==x,y]

                 x
Out[80]= {{y -> E }}
```

So far so good.

```
In[81]:= Solve[Log[y]+Log[1+y]==x,y]

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Out[81]= Solve[Log[y] + Log[1 + y] == x, y]
```

It appears to fail in an apparently trivial situation, but after a closer consideration we see that it is a reasonable conservativeness rather than a failure because in the complex domain (where we always work) a sum of logarithms is *not* equal to the logarithm of the product.

```
In[82]:= Log[y(1+y)]-Log[y]-Log[1+y]/.y:>(-2)

Out[82]= -2 I Pi
```

If we are sure to work in a safer subdomain (for instance over positive reals) we can force the logarithm conversion by providing a *Mathematica* rule like:

```
In[83]:= r={(a_. Log[b_] + c_. Log[d_]):> Log[ b^a d^c ]}

                                                      a  c
Out[83]= {Log[b_] (a_.) + Log[d_] (c_.) :> Log[b  d ]}
```

```
In[84]:= Solve[Log[y]+Log[1+y]==x,y]/.r

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

                                  x
                -1 + Sqrt[1 + 4 E ]
Out[84]= {{y -> -------------------},
                         2

                         x
        -1 - Sqrt[1 + 4 E ]
  {y -> -------------------}}
                 2
```

This looks better but now it is our responsibility to check that these solutions are not spurious. Thus if our domain is positive reals then the second solution is most certainly spurious for any x.

5 Linear Ordinary Differential Equations

■ 5.1 Equations with constant coefficients

Ordinary differential equations with constant coefficients form a class of equations that are solved using straightforward and explicit formulas.

The basic solutions (called fundamental solutions in mathematics) are expressed in terms of polynomial and exponentials of the independent variable.

For example, here is how the general solution of a 3rd-order equation goes:

```
In[85]:= DSolve[y'''[x]==y[x],y[x],x]

                                  2/3               4/3
                    x         (-1)    x         (-1)    x
Out[85]= {{y[x] -> E  C[1] + E          C[2] + E          C[3]}}
```

Here `C[1],C[2],C[3]` are arbitrary constants.

To get hold of a specific solution of the above equation, we should specify the initial data with it, for example

```
In[86]:= DSolve[{y'''[x]==y[x],y[0]==1,y'[0]==2,y''[0]==3},
         y[x],x]

Out[86]= {{y[x] ->

                        2/3                        4/3
               1/3  (-1)    x            2/3   (-1)    x
       x   (-1)    E             (2 + (-1)   ) E
    2 E  + ------------------ - ------------------------}}
                       4/3                 3
             1 + 2 (-1)
```

General solutions of this form are always valid for the equations called "homogeneous" like the one solved above.

A closely neighbouring class of equations admits a term called "nonhomogeneous part" which may explicitly depend on the x-variable provided only that it is not multiplied by y or any of its derivatives.

Equations with such "weak" dependence on x are also solved straightforwardly as the following one:

```
In[87]:= DSolve[y'''[x]==y[x]+x^3,y[x],x]

Out[87]={{y[x] ->

                   2/3              4/3
     x         (-1)    x        (-1)    x
    E  C[1] + E         C[2] + E          C[3] +

                                        4/3
          5/3           -DSolve`t + (-1)    x         3
     ((-1)   Integrate[E                      DSolve`t

                              5/3
                     (1 + (-1)   ) DSolve`s[1]
          Integrate[E

                   1/3       2/3
              ((-1)    + (-1)   ) x
            (E                     -

                    1/3       2/3
               ((-1)    + (-1)   ) DSolve`s[1]
              E                              ),

           {DSolve`s[1], DSolve`t, x}], {DSolve`t, 0, x}]) /
               1/3
      (1 + (-1)   )}}
```

■ 5.2 Equations with variable coefficients

Ordinary differential equations with variable coefficients will generally require the `Calculus/DSolve` package.

Rather sophisticated methods involving hypergeometric functions have been implemented to deal with the equations of that sort. However, it should be understood that the problem of solving a generic differential equation is algorithmically intractable. Therefore `DSolve` "solves what it can and what it can't it leaves". We can claim, however, that solving strategies in the `DSolve` package are very advanced and up-to-date.

Let us look at some examples.

```
In[88]:= <<Calculus`DSolve`
```

This loads the package.

```
In[89]:= DSolve[y'[x]+2x y[x]==x Exp[-x^2],y[x],x]

                    2
                   x  + 2 C[1]
Out[89]= {{y[x] -> -----------}}
                          2
                         x
                     2 E
```

First-order linear differential equations are always solvable and they are solved on a regular basis.

Some of the second-order linear differential equations are solved in elementary functions and elementary quadratures.

```
In[90]:= DSolve[y''[x]+x y'[x]+y[x]==0,y[x],x]

Out[90]= {{y[x] ->

                           2                           2
                         -x          2               -x
            Sqrt[2] Sqrt[---] Sqrt[x ] C[1] Erf[Sqrt[---]]
    C[2]                  2                           2
    ---- - -----------------------------------------------}}
      2                     2
     x /2                  x /2  2
    E                     E     x
```

Some are solved in hypergeometric functions.

```
In[91]:= DSolve[y''[x]-(x^2+1)y[x]==0,y[x],x]

Out[91]= {{y[x] ->

                        2 2
     (-1 + Sqrt[(1 + 2 x ) ])/4      Pi
    E                           Sqrt[--] C[2]
                                     8

                                 2 2
          Sqrt[-1 + Sqrt[(1 + 2 x ) ]]
      Erf[----------------------------] +
                    Sqrt[2]

                         2 2
       (1 - Sqrt[(1 + 2 x ) ])/4                          2 2
     (E                          Sqrt[-1 + Sqrt[(1 + 2 x ) ]]

                                                       2 2
                                       3  -1 + Sqrt[(1 + 2 x ) ]
        C[1] HypergeometricU[1, -, ----------------------]) / 2}
                                       2            2

   }
```

Some cannot be solved at all.

```
In[92]:= DSolve[y''[x]==(1+Tan[x]^2)y[x],y[x],x]

                                          2
Out[92]= DSolve[y''[x] == (1 + Tan[x] ) y[x], y[x], x]
```

6 Nonlinear Ordinary Differential Equations

■ 6.1 First order nonlinear o.d.e.

A first-order nonlinear ordinary differential equation is a relation of the form

```
F[ x  , y[x] , y'[x] ]  == 0.
```

Such an equation is solvable if and only if we are able to find its first integral

```
I [  x  ,  y[x]  ]  ==  C[1]
```

and the task of finding such an integral is generally algorithmically unsolvable. A case study of the first-order equations has been implemented in the `Calculus/DSolve` package.

The first solvable case is the case of **separating variables, i.e.,**

```
y'[x] == f[x] g[y[x]]
```

For instance:

```
In[93]:= DSolve[y'[x]-a x (y[x]^2+1)==0,y[x],x]

                             2
                         a x  + 2 C[1]
Out[93]= {{y[x] -> Tan[-------------]}}
                               2
```

or also

```
In[94]:=
         DSolve[y'[x]==(1+y[x]^2)/((y[x]+Sqrt[1+y[x]])(1+x)^(3/2)),
         y[x],x]

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

                                Sqrt[1 + y[x]]
               (-1 - I) ArcTanh[--------------]
                                 Sqrt[1 - I]
Out[94]= Solve[-------------------------------- +
                         Sqrt[1 - I]

                     Sqrt[1 + y[x]]
    (-1 + I) ArcTanh[--------------]                2
                      Sqrt[1 + I]      Log[1 + y[x] ]
    -------------------------------- + -------------- ==
              Sqrt[1 + I]                    2

       -2
   ----------- + C[1], y[x]]
   Sqrt[1 + x]
```

which is not so easy to understand but which still makes a complete answer though the unknown function `y[x]` is presented in a implicit form here.

The equations with separating variables go together with those that are reduced to such by substitutions of the form `y[x] <-  g[x] + u[x]` or `y[x] <-  g[x] u[x]`.

The next case to analyze is a **shiftable equation**, which basically has the form

```
            y'[x] ==  f[ y[x] + a x ]
```

where *f* is any function whatever.

A criterion is programmed to find such *a* if it exists and then to integrate the equation.

An example would be:

```
In[95]:= DSolve[y'[x]==Cos[a y[x]+b x],y[x],x]

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::ifun:
   Warning: Inverse functions are being used by Solve, so some
     solutions may not be found.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::ifun:
   Warning: Inverse functions are being used by Solve, so some
     solutions may not be found.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

General::stop:
   Further output of Solve::tdep
     will be suppressed during this calculation.

                                              b x + a y[x]
                              Sqrt[a - b] Tan[------------]
                                                   2
                    2 ArcTanh[-----------------------------]
                                       Sqrt[a + b]
Out[95]= Solve[-x + ---------------------------------------- -
                            Sqrt[a - b] Sqrt[a + b]

    a y[x]
    ------ == C[1], y[x]]
      b
```

The third prospective class of solvable equations are **scalable** equations (sometimes also called homogeneous).

A characteristic example would be

```
In[96]:= DSolve[ x^2 y'[x]-y[x]^2 - x y[x] == 0 , y[x], x]

                               x
Out[96]= {{y[x] -> -(──────────────────)}}
                     -C[1] + Log[x]
```

Some equations are not immediately scalable but do reduce to scalable through an appropriate change of variables, like:

```
In[97]:= DSolve[(y[x]+1)y'[x]==y[x]+x,y[x],x]

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

General::stop:
   Further output of Solve::tdep
     will be suppressed during this calculation.

                           3 - x + 2 y[x]
                  ArcTanh[────────────────]
                          Sqrt[5] (-1 + x)
Out[97]= Solve[-(──────────────────────────) +
                           Sqrt[5]

                    2                   2
    Log[-1 - x + x  - 3 y[x] + x y[x] - y[x] ]
    ────────────────────────────────────────── == C[1], y[x]]
                        2
```

Generalized Bernoulli equations come next, which are equations of the form

```
U'[y[x]]  y'[x]  ==  a[x]  U[y[x]]  +  b[x]
```

The solving method for such equations is obvious, so the principal merit of the `DSolve` package is the ability to recognize the generalized Bernoullis under all possible disguises.

Example:

```
In[98]:= DSolve[x^2(x-1)y'[x] - y[x]^2 - x(x-2)y[x]==0,y[x],x]

                           2
                          x
Out[98]= {{y[x] -> ─────────────────}}
                   1 + C[1] - x C[1]
```

The `DSolve` algorithm deals with a list of more sophisticated classes of equations like **Clairaut, Riccatti,** and **generalized homogeneous,** but discussing those will make this tutorial too specialized.

■ 6.2 Second order nonlinear o.d.e.

A second-order equation of this type is any equation containing the second derivative of the unknown function in any form like

```
F[x, y[x] ,  y'[x]  ,  y''[x] ]  ==  0.
```

The second-order nonlinear differential equations present even greater challenge for algorithmic treatment then the first-order ones.

Generally, a second-order equation is solved in closed form only if it is reduced to first order using an appropriate **symmetry reduction**.

Again to avoid being too specialized we will point out how such a reduction works for the equations with missing x.

```
F[y[x]  , y'[x], y''[x]] == 0.
```

This is a text book trick to have new variable $z = y[x]$, $u = y'[x]$, so that

```
y''[x]  = u[z]  u'[z]
```

and

```
F[z, u[z] ,  u[z]  u'[z] ]  == 0.
```

An example would be

```
In[99]:= eq660 = y''[x] == a Sqrt[y'[x]^2 + 1 ]

                                        2
Out[99]= y''[x] == a Sqrt[1 + y'[x] ]

In[100]:= DSolve[eq660,y[x],x]

                                Cosh[a (x + C[1])]
Out[100]= {{y[x] -> C[2] + ------------------}}
                                        a
```

But there is a whole set of other symmetry reduction procedures implemented as the following examples would suggest.

```
In[101]:= eq693 = y''[x] == a ( x y'[x] - y[x] )^2 / x^3

                                                 2
                          a (-y[x] + x y'[x])
Out[101]= y''[x] == ----------------------
                                  3
                                 x

In[102]:= DSolve[eq693,y[x],x]

Solve::ifun:
   Warning: Inverse functions are being used by Solve, so some
     solutions may not be found.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

Solve::ifun:
   Warning: Inverse functions are being used by Solve, so some
     solutions may not be found.

Solve::tdep:
   The equations appear to involve transcendental functions of
     the variables in an essentially non-algebraic way.

                                                  (a y[x])/x
Out[102]= Solve[Log[x] + Log[1 + a E               C[2]] -

      a y[x]
      ------ == C[1], y[x]]
        x

In[103]:= eq697 = y''[x] == ( 2 x y[x] + x^3 ) y'[x] / x^4 -
             4 y[x]^2 / x^4

                                     2       3
                          -4 y[x]     (x  + 2 x y[x]) y'[x]
Out[103]= y''[x] == -------- + ---------------------
                              4                   4
                             x                   x
```

```
In[104]:= DSolve[eq697,y[x],x]

Out[104]= {{y[x] ->

     2
    x  (1 + Sqrt[-1 + C[2]]
        Tan[Sqrt[-1 + C[2]] (-C[1] + Log[x])])}}
```

■ 6.3 Nonlinear differential systems: perspective.

It is much the same situation with systems of nonlinear differential equations as the above.

Because of higher complexity of the tasks only systems of two and three equations are dealt with at present and among them only those are solved that do admit order reduction or dimension reduction.

Just to give idea of how the algorithm works, consider a system of two first-order autonomous equations

```
{  u'[x] == F[ u[x],  v[x] ]  ,  v'[x] == G[u[x]  , v[x]] }
```

It is essential that the right-hand sides of the equations do not depend on x.

If we consider u as an independent variable for a while we get a derived first-order scalar differential equation

```
v'[u] == ( G[u  , v[u]] / F[u, v[u]] ).
```

Solving this we get back a dependence between v and u which then can be further substituted into the original system.

This and a large list of similar reductions is implemented in the upcoming version of the `DSolve` package.