11. Solving Differential Equations (and Systems)

2015-05-15

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

11.1 Analytical Solution

K. Rektorys: Survey of applicable Mathematics. MIT Press, 1969 contains a long list of ordinary differential equations with solutions.
E. Kamke, Differentialgleichungen. Lösungsmethoden u. Lösungen. [Bd] 1.2 Stuttgart: Teubner 1979-1983,
list ordinary and partial differential equations together with their solutions.
A.D. Polyanin, V.F. Zaitsev, Handbuch der linearen Differentialgleichungen.
Spektrum Akademischer Verlag, 1996.

For many but not all ordinary differential equations enumerated in these books analytical solutions can be obtained by the following commands.

11.1.1 Solving a single differential equation

DSolve[*eqn*, y[x], x] ---------	render the general solution of the differential equation given in ***eqn*,** i.e. **y[x]**, taking **x** as the independent variable
DSolve[{*eqn*,*indat*},y[x], x]	render the solution belonging to initial data given in ***indat*.**

General solution: (C[1], C[2], C[3], ... are the integration constants.)

11.1.1.1 Some examples

```
Clear[x, y, a]
DSolve[ y'[x] == a y[x], y[x], x]
```

$\{\{y[x] \to e^{a x} C[1]\}\}$

```
DSolve[ y''[x] + k^2 y[x] == 0, y[x], x]
```

```
{{y[x] → C[1] Cos[k x] + C[2] Sin[k x]}}
```

```
DSolve[ y''''[x] + k^4 y[x] == 0, y[x], x]
```

$\{\{y[x] \to e^{(-1)^{3/4} k x} C[1] + e^{-(-1)^{1/4} k x} C[2] + e^{-(-1)^{3/4} k x} C[3] + e^{(-1)^{1/4} k x} C[4]\}\}$

```
DSolve[ y''[x] + y'[x] / x  +  (1 - n^2 / x^2) y[x] == 0,
        y[x], x ]
```

```
{{y[x] → BesselJ[n, x] C[1] + BesselY[n, x] C[2]}}
```

This is Bessel's differential equation. If n is not an integer, so n = v, the two functions $J_v(x)$ and $J_{-v}(x)$ are linealy independent. Howerver, for n = integer, these two solutions are dependent; so the Neumann solution Y_n(x) must be used as a second solution.

```
DSolve[ y''[x] + y'[x] / x  +  (1 - 4 / x^2) y[x] == 0,
        y[x], x ]
```

```
{{y[x] → BesselJ[2, x] C[1] + BesselY[2, x] C[2]}}
```

```
DSolve[ y''[x] + 2 y'[x] / x + (1 - n (n + 1) / x^2) y[x] == 0,
        y[x], x ] // ExpandAll
```

```
{{y[x] → C[1] SphericalBesselJ[n, x] + C[2] SphericalBesselY[n, x]}}
```

The above differential equation is obtained when the Hemholtz equation $(\Delta + k^2)\psi = 0$ is solved in spherical coordinates by separating the variables $\psi(r,\theta,\phi) := R(r)\,\Theta(\theta)\,\Phi(\phi)$. The resulting equation for the radial variable R(r) is the above differential equation with k^2 in place of 1. So the radial solution is R(r) = z_n(k r), where **the spherical Bessel function, $z_n(x) = \sqrt{2/\pi x}\; Z_{n+1/2}(x)$,** is propotional to a common Bessel function of order n + 1/2.

```
so = DSolve[y''''[x] + 3 y'''[x] + k^4 y[x] == 0, y[x], x]
```

$$\left\{\left\{y[x] \to e^{x\,\text{Root}\left[k^4+3\,\#1^3+\#1^4\&,1\right]}\,C[1] + e^{x\,\text{Root}\left[k^4+3\,\#1^3+\#1^4\&,2\right]}\,C[2] + e^{x\,\text{Root}\left[k^4+3\,\#1^3+\#1^4\&,3\right]}\,C[3] + e^{x\,\text{Root}\left[k^4+3\,\#1^3+\#1^4\&,4\right]}\,C[4]\right\}\right\}$$

A linear differential equation with constant coefficients as given above can be solved by an exponential $e^{\alpha x}$, where α must be a root of the characteristic polynomia, which in the present case is $\alpha^4 + 3\alpha^3 + k^4$.

In quantum mechanics the eigenvalue equation of the one-dimensional harmonic oscillator is given by the following second order differential equation, see also § 11.2.7.

```
seh = DSolve[ y''[x] + (ε - x^2) y[x] == 0, y[x], x]
```

$$\left\{\left\{y[x] \to C[2]\,\text{ParabolicCylinderD}\left[\frac{1}{2}(-1-\varepsilon),\, i\sqrt{2}\,x\right] + C[1]\,\text{ParabolicCylinderD}\left[\frac{1}{2}(-1+\varepsilon),\, \sqrt{2}\,x\right]\right\}\right\}$$

```
sh = DSolve[ y''[x] - 2 x y'[x]+ 2 n y[x] == 0, y[x], x]
```

$$\left\{\left\{y[x] \to C[1]\,\text{HermiteH}[n,x] + C[2]\,\text{Hypergeometric1F1}\left[-\frac{n}{2},\frac{1}{2},x^2\right]\right\}\right\}$$

For nonnegative integers n, the function **HermiteH[n, x]** beomes *a* polynomial and the product $e^{-x^2/2}$ **HermiteH[n, x] fulfils the boundary conditions** $y(\pm\infty) = 0$. This fixes the dimensionless energy parameter ε by $(\varepsilon - 1)/2 = 2n$, $n = 1, 3, 5, \ldots$; i.e. $\varepsilon = k + 1/2$, $k = 0, 1, 2, \ldots$.

11.1.1.2 Solution Matching Given Data as eg. in Initial Value Problems

```
DSolve[ {y'[x] == a y[x], y[0] == y0}, y[x], x]
```

$$\{\{y[x] \to e^{a x}\,y0\}\}$$

```
DSolve[{y'[x] - a y[x] == d Exp[c x], y[0] == y0}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{e^{a x-(a-c) x}\left(-d + d\,e^{(a-c) x} + a\,e^{(a-c) x}\,y0 - c\,e^{(a-c) x}\,y0\right)}{a-c}\right\}\right\}$$

```
DSolve[{x[0]==3, x'[0]==0, x''[t]+x'[t]-x[t]==6},x[t],t]//Flatten
```

$$\left\{x[t] \to -\frac{3}{10}\left(20 - 15\,e^{\left(-\frac{1}{2}-\frac{\sqrt{5}}{2}\right)t} + 3\sqrt{5}\,e^{\left(-\frac{1}{2}-\frac{\sqrt{5}}{2}\right)t} - 15\,e^{\left(-\frac{1}{2}+\frac{\sqrt{5}}{2}\right)t} - 3\sqrt{5}\,e^{\left(-\frac{1}{2}+\frac{\sqrt{5}}{2}\right)t}\right)\right\}$$

The data need not be initial data; for example, the data may be the value of the wanted solution at two places:

```
DSolve[{x[0]==3, x[5]==9, x''[t]+x'[t]-x[t]==6},x[t],t]//Flatten
```

$$\left\{x[t] \to \frac{1}{-1+e^{5\sqrt{5}}}\,3\left(2 - 2\,e^{5\sqrt{5}} - 3\,e^{\left(-\frac{1}{2}+\frac{\sqrt{5}}{2}\right)t} - 5\,e^{\frac{5}{2}+\frac{5\sqrt{5}}{2}+\left(-\frac{1}{2}-\frac{\sqrt{5}}{2}\right)t} + 3\,e^{5\sqrt{5}+\left(-\frac{1}{2}-\frac{\sqrt{5}}{2}\right)t} + 5\,e^{\frac{5}{2}+\frac{5\sqrt{5}}{2}+\left(-\frac{1}{2}+\frac{\sqrt{5}}{2}\right)t}\right)\right\}$$

11.1.1.3 Plotting Solutions

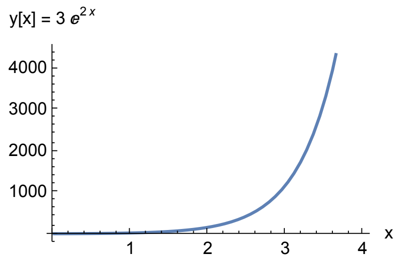
```
Clear[x,y,f];
so = DSolve[ {y'[x] == 2 y[x], y[0] == 3}, y[x], x]//Flatten
```

$\{y[x] \to 3\, e^{2x}\}$

```
g = y[x] /. so
```

$3\, e^{2x}$

```
Plot[g, {x, 0, 4}, AxesLabel → {"x", "y[x] = 3 e^(2 x)"}, ImageSize → 200]
```



11.1.1.4 Printing Numerical Values of Solutions

```
tso = Table[{x, g} // Flatten, {x, 0, 4, .5}]
```

```
{{0., 3.}, {0.5, 8.15485}, {1., 22.1672}, {1.5, 60.2566},
 {2., 163.794}, {2.5, 445.239}, {3., 1210.29}, {3.5, 3289.9}, {4., 8942.87}}
```

```
TableForm[{"x    y(x)"}]
TableForm[tso]
```

```
x    y(x)

0.     3.
0.5    8.15485
1.     22.1672
1.5    60.2566
2.     163.794
2.5    445.239
3.     1210.29
3.5    3289.9
4.     8942.87
```

11.1.2 Solving for solutions expressed as pure functions

DSolve[eqn, y , x] ---	give a solution for y in pure function form

Note the difference in the calls of **NDSolve[]** above and at the beginning of 11.1.1. This has farreaching consequences for the storage of the result and the reaction on further commands. When the solution obtained as a common function is inserted in an expression requiring the solution, the substitution is only done for the function, but not for the derivative nor for the inital values. Whereas for a solution obtained as a pure function, the substitution is done for the function, the derivative(s) and the intial data. The difference is shown below.

```
Clear[x, y, f];
f = DSolve[ {y'[x] == 2 y[x], y[0] == 3}, y[x], x] // Flatten
```

$\{y[x] \to 3\, e^{2x}\}$

```
y[x] + 2 y'[x] + y[0] /. f
```

$3\, e^{2x} + y[0] + 2\, y'[x]$

```
Clear[x, y, f];
f = DSolve[ { y'[x] == a y[x] }, y, x] // Flatten
```
$\{y \to \text{Function}[\{x\}, e^{a x} C[1]]\}$

```
y[x] + 2 y'[x] + y[0] /. f
```
$C[1] + e^{a x} C[1] + 2 a e^{a x} C[1]$

```
Clear[x, y, h];
h = DSolve[ {y'[x] == 2 y[x], y[0] == 3}, y, x] // Flatten
```
$\{y \to \text{Function}[\{x\}, 3 e^{2 x}]\}$

```
y[x] + 2 y'[x] + y[0] /. h
```
$3 + 15 e^{2 x}$

```
g = f /. {a -> 2, C[1] -> 3}
```
$\{y \to \text{Function}[\{x\}, e^{2 x} 3]\}$

```
Plot[y[x] /.g, {x, 0, 4}, AxesLabel → {"x", "y[x] = 3 e^(2 x)"}, ImageSize → 200];
```
gives same figure as above.

```
Clear[y, x]
syseqn = { y''[x] + k^2 y[x] == 0}
```
$\{k^2 y[x] + y''[x] == 0\}$

```
DSolve[syseqn, y[x], x]
```
$\{\{y[x] \to C[1] \text{Cos}[k x] + C[2] \text{Sin}[k x]\}\}$

```
Clear[y, x, f, g]
syseqn = { y''[x] + k^2 y[x] == 0, y[0] == 0, y'[0] == 1 }
```
$\{k^2 y[x] + y''[x] == 0, y[0] == 0, y'[0] == 1\}$

```
so = DSolve[syseqn, y[x], x] // Flatten
```
$\left\{y[x] \to \frac{\text{Sin}[k x]}{k}\right\}$

```
su = DSolve[syseqn, y, x] // Flatten
```
$\left\{y \to \text{Function}\left[\{x\}, \frac{\text{Sin}[k x]}{k}\right]\right\}$

```
ExpandAll[su]
```
$\left\{y \to \text{Function}\left[\{x\}, \frac{\text{Sin}[k x]}{k}\right]\right\}$

```
g = y[x] /. su
```
$\frac{\text{Sin}[k x]}{k}$

11.1.3 Solving Systems of Differential Equations

DSolve[{eq1,eq2,...}, {y1[x],y2[x],...}, x]	solve a list of differential equations
DSolve[{eq1,eq2,...}, {y1,y2,...}, x]	same task, pure functions as result

The following system:

$$x' - y = t, \quad y' - 4x = -2$$

is solved:

```
Clear[x, y, f, g];
sy = {x'[t] - y[t] == t, y'[t] - 4 x[t] == - 2}
```

$$\{-y[t]+x'[t]==t,\ -4\,x[t]+y'[t]==-2\}$$

```
DSolve[sy, {x[t], y[t]}, t] // Flatten
```

$$\Big\{x[t]\to\frac{1}{16}e^{-4t}\left(-1+e^{4t}\right)\left(1-2t-e^{4t}(1+2t)\right)+\frac{1}{16}e^{-4t}\left(1+e^{4t}\right)\left(1-2t+e^{4t}(1+2t)\right)+\frac{1}{2}e^{-2t}\left(1+e^{4t}\right)C[1]+\frac{1}{4}e^{-2t}\left(-1+e^{4t}\right)C[2],$$
$$y[t]\to\frac{1}{8}e^{-4t}\left(1+e^{4t}\right)\left(1-2t-e^{4t}(1+2t)\right)+\frac{1}{8}e^{-4t}\left(-1+e^{4t}\right)\left(1-2t+e^{4t}(1+2t)\right)+e^{-2t}\left(-1+e^{4t}\right)C[1]+\frac{1}{2}e^{-2t}\left(1+e^{4t}\right)C[2]\Big\}$$

```
Clear[x, y, f, g];
sy = {x'[t] - y[t] == t, y'[t] - 4 x[t] == - 2,
      x[0] == 1,      y[0] ==  1 }
```

$$\{-y[t]+x'[t]==t,\ -4\,x[t]+y'[t]==-2,\ x[0]==1,\ y[0]==1\}$$

```
DSolve[sy, {x[t], y[t]}, t]
```

$$\left\{\left\{x[t]\to\frac{1}{8}e^{-2t}\left(1+2\,e^{2t}+5\,e^{4t}\right),\ y[t]\to\frac{1}{4}e^{-2t}\left(-1+5\,e^{4t}-4\,e^{2t}\,t\right)\right\}\right\}$$

```
DSolve[sy, {x, y}, t]
```

$$\Big\{\Big\{x\to\text{Function}\left[\{t\},\ \frac{1}{8}e^{-2t}\left(1+2\,e^{2t}+5\,e^{4t}\right)\right],$$
$$y\to\text{Function}\left[\{t\},\ \frac{1}{4}e^{-2t}\left(-1+5\,e^{4t}-4\,e^{2t}\,t\right)\right]\Big\}\Big\}$$

11.1.4 Motion of Charged Particle in Static Crossed Homogeneous Electric and Magnetic Fields

D.M. Cook et al., Comp. in Phys., 6 (#5, Sep./Oct.92), 532f.

Equation of motion:

$$\frac{m\ d^2\,\mathbf{r}}{dt^2} = q\,\frac{d\mathbf{r}}{dt}\times\mathbf{B} + q\,\mathbf{E}$$

Initial data:

E = (0, E0, 0), B = (0, 0, B0) ; r(0) = (0,0,0), dr(0)/dt = (vx, vy, vz).

11.1.4.1 Setting up the Equations of Motion and Solving Them

```
Clear[m, r, v, a, B0, E0]
Efield = {0, E0, 0};  Bfield = {0, 0, B0};

r = { x[t], y[t], z[t]}; v = D[r, t]; a = D[v, t];

Force = q Efield + q Cross[ v, Bfield ]
```

$$\{B0\,q\,y'[t],\ E0\,q-B0\,q\,x'[t],\ 0\}$$

```
eq = Table[ Force[[i]] == m a[[i]], {i, 3}]
```

$$\{B0\,q\,y'[t]==m\,x''[t],\ E0\,q-B0\,q\,x'[t]==m\,y''[t],\ 0==m\,z''[t]\}$$

```
id = { x[0] == 0, x'[0] == vx, y[0] == 0, y'[0] == vy,
   z[0] == 0, z'[0] == vz };
```

```
Soln = DSolve[ Union[eq, id], {x, y, z}, t ] // Flatten;
TableForm[Soln]
```

$$z \to \text{Function}[\{t\}, t\,vz]$$

$$x \to \text{Function}\left[\{t\}, \frac{B0\,E0\,q\,t + B0\,m\,vy - B0\,m\,vy\,\text{Cos}\left[\frac{B0\,q\,t}{m}\right] - E0\,m\,\text{Sin}\left[\frac{B0\,q\,t}{m}\right] + B0\,m\,vx\,\text{Sin}\left[\frac{B0\,q\,t}{m}\right]}{B0^2\,q}\right]$$

$$y \to \text{Function}\left[\{t\}, \frac{-B0\,m\,vx - E0\,m\,\text{Cos}\left[\frac{B0\,q\,t}{m}\right] + B0\,m\,vx\,\text{Cos}\left[\frac{B0\,q\,t}{m}\right] + E0\,m\,\text{Cos}\left[\frac{B0\,q\,t}{m}\right]^2 + B0\,m\,vy\,\text{Sin}\left[\frac{B0\,q\,t}{m}\right] + E0\,m\,\text{Sin}\left[\frac{B0\,q\,t}{m}\right]^2}{B0^2\,q}\right]$$

11.1.4.2 Checking the Solution to fulfil the euations of motion and the initial data

```
Simplify[eq /. Soln]
```

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

```
Simplify[{x[0], y[0], z[0]} /. Soln]
```

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

```
Simplify[{x'[0], y'[0], z'[0]} /. Soln]
```

```
{vx, vy, vz}
```

11.1.4.3 Getting Expressions

```
X[t_] := Expand[ x[t] /. Soln]
Y[t_] := Expand[ y[t] /. Soln]
Z[t_] := Expand[ z[t] /. Soln]
```

```
rs = {X[t], Y[t], Z[t]} /. {m -> q B0 / ω, E0 -> vd B0};
TableForm[rs]
```

$$t\,vd + \frac{vy}{\omega} - \frac{vy\,\text{Cos}[t\,\omega]}{\omega} - \frac{vd\,\text{Sin}[t\,\omega]}{\omega} + \frac{vx\,\text{Sin}[t\,\omega]}{\omega}$$

$$-\frac{vx}{\omega} - \frac{vd\,\text{Cos}[t\,\omega]}{\omega} + \frac{vx\,\text{Cos}[t\,\omega]}{\omega} + \frac{vd\,\text{Cos}[t\,\omega]^2}{\omega} + \frac{vy\,\text{Sin}[t\,\omega]}{\omega} + \frac{vd\,\text{Sin}[t\,\omega]^2}{\omega}$$

$$t\,vz$$

```
rf = rs /. {ω -> 1, vd -> 0.04, vx -> 0.05,
    vy -> .033, vz -> 0.03}; TableForm[rf]
```

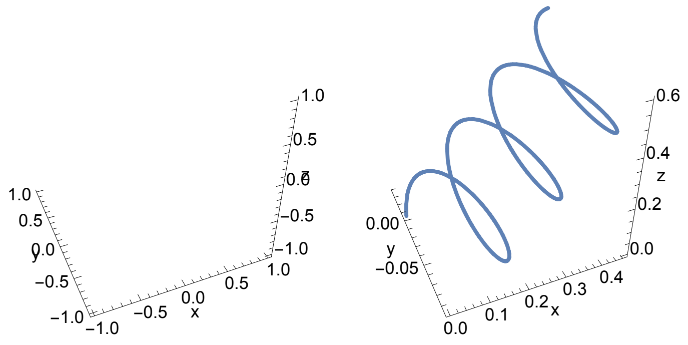
```
0.033 + 0.04 t - 0.033 Cos[t] + 0.01 Sin[t]
- 0.05 + 0.01 Cos[t] + 0.04 Cos[t]^2 + 0.033 Sin[t] + 0.04 Sin[t]^2
0.03 t
```

11.1.4.3 Drawing Curves

```
pleb1 = ParametricPlot3D[ Evaluate[rf1], {t,0,20}, Boxed -> False,
 AxesLabel -> {"x","y","z"}, DisplayFunction -> Identity,
BoxRatios -> {1,1,1},ViewPoint->{-1.010, -2.400, 2.000}];
```

```
rf2 = rs /. {ω -> 1, vd -> 0.02, vx -> 0.05,
    vy -> .06, vz -> 0.03} ;
pleb2 = ParametricPlot3D[ Evaluate[rf2], {t,0,20},Boxed -> False,
 AxesLabel -> {"x","y","z"}, DisplayFunction -> Identity,
BoxRatios -> {1,1,1},ViewPoint->{-1.010, -2.400, 2.000}];
```

```
Show[GraphicsRow[{pleb1, pleb2}]]
```



11.1.5 Changing the Dependent Variable of a Differential Equation

A differential equation can be transformed by changing the dependent variable $y(x)$ of this equation. In most cases this is done by assuming that the new dependent variable is $z(x) = y(x)/w(x)$ with the function $w(x)$ chosen such that the new differential equation has a form more suitable for a given purpose. For example, in a second order differential equation one may remove the first derivative. Or the new differential equation has standard functions as solutions. Here, in a specific example, it is shown how this task may be accomplished. We start from Bessel's equation and transform it into a new equation without the first derivative.

```
Clear[m, w, x, z, y]
oldeq = y''[x] + y'[x] / x + y[x] (1 - m^2 / x^2)
```

$$\left(1-\frac{m^2}{x^2}\right) y[x]+\frac{y'[x]}{x}+y''[x]$$

```
Length[oldeq]
```

```
3
```

```
FullForm[oldeq]
```

```
Plus[Times[Plus[1, Times[-1, Power[m, 2], Power[x, -2]]], y[x]],
 Times[Power[x, -1], Derivative[1][y][x]], Derivative[2][y][x]]
```

```
cold =
Table[Coefficient[oldeq, Derivative[k][y][x]], {k, 0, 2}]
```

$$\left\{1-\frac{m^2}{x^2}, \frac{1}{x}, 1\right\}$$

The above list gives the coefficients of the various derivatives. This is used to find a new differential equation, which is the old differential equations with the product $y(x) = z(x)\, w(x)$ inserted for the old dependent variable. The resulting equation is rearranged to give better insight.

```
neweq =
Sum[ D[z[x] w[x], {x, k}] cold[[k + 1]], {k, 0, 2}] // Expand
```

$$w[x]\, z[x]-\frac{m^2\, w[x]\, z[x]}{x^2}+\frac{z[x]\, w'[x]}{x}+\frac{w[x]\, z'[x]}{x}+2\, w'[x]\, z'[x]+z[x]\, w''[x]+w[x]\, z''[x]$$

```
cnew =
Table[Coefficient[neweq, Derivative[k][z][x]], {k, 0, 2}]
```

$$\left\{w[x]-\frac{m^2\, w[x]}{x^2}+\frac{w'[x]}{x}+w''[x], \frac{w[x]}{x}+2\, w'[x], w[x]\right\}$$

```
fneweq = Sum[D[z[x], {x, k}] cnew[[k + 1]], {k, 0, 2}]
```

$$\left(\frac{w[x]}{x}+2\, w'[x]\right) z'[x]+z[x]\left(w[x]-\frac{m^2\, w[x]}{x^2}+\frac{w'[x]}{x}+w''[x]\right)+w[x]\, z''[x]$$

Now we prescribe the new equation to have no first derivative. This yields a first order differential

differential equation. For this purpose it is advantageous to get the solution for w(x) in a pure function form
(cf. 11.1.2 and 21.3).

```
heq = cnew[[2]] == 0
```

$$\frac{w[x]}{x} + 2\, w'[x] == 0$$

```
s1 = DSolve[heq, w, x] // Flatten
```

$$\left\{w \to \text{Function}\left[\{x\}, \frac{C[1]}{\sqrt{x}}\right]\right\}$$

```
fnew = fneweq /. s1
```

$$\left(\frac{C[1]}{4\, x^{5/2}} - \frac{m^2\, C[1]}{x^{5/2}} + \frac{C[1]}{\sqrt{x}}\right) z[x] + \frac{C[1]\, z''[x]}{\sqrt{x}}$$

```
normf = Coefficient[fnew, Derivative[2][z][x]]
```

$$\frac{C[1]}{\sqrt{x}}$$

```
fnew = fnew / normf // Expand
```

$$z[x] + \frac{z[x]}{4\, x^2} - \frac{m^2\, z[x]}{x^2} + z''[x]$$

```
fdeq = Collect[fnew, z[x] / x^2] == 0
```

$$z[x] + \frac{\left(\frac{1}{4} - m^2\right) z[x]}{x^2} + z''[x] == 0$$

11.2 Numerical Solution of Differential Equations

In *Mathematica* it is quite easy to get numeric solutions of ordinary differential equations; and to plot the solutions.

More information on the corresponding package may be found at:
http://reference.wolfram.com/mathematica/tutorial/NDSolvePackages.html

NDSolve[{eq_1, eq_2, ...}, y, {x, xmin, xmax}]
find a numerical solution for the function y with x in the range xmin to xmax

NDSolve[{eq_1, eq_2, ...}, {y_1, y_2,...}, {x, xmin, xmax}]
find numerical solutions for several functions y_i

In the following is assumed that the numeric solution is labelled **solution**.

sol[x_] = y[x] /. solution use the list of rules for the function y to get a function **sol[x]** for the solution **y[x]**

Plot[Evaluate[y[x] /. solution]], {x,xmin, xmax}] plot the solution to a differential equation

11.2.1 Numeric Solutions of Systems of Differential Equations for One Dependent Variable

11.2.1.1 Example 1

```
Clear[y, x]; xmin = 0.; xmax = 6;
s = NDSolve[ {y''[x] == - y[x], y[0] == 0, y'[0] == 1},
                                                y, {x, xmin, xmax}] // Flatten
```

```
{y → InterpolatingFunction[ Domain: {{0., 6.}}
                            Output: scalar ]}
```

```
y[2.31]  /. s
y'[2.31]  /. s
y''[2.31]  /. s
```

```
0.739005

-0.6737

-0.739005
```

```
np = 5; xk := k (xmax - xmin) / np;
ts = Table[{k, N[xk], y[xk] /. s, y'[xk] /. s }, {k, 0, np}] // Chop;

TableForm["k     x      y(x)          y'(x)   "]
TableForm[ts]
```

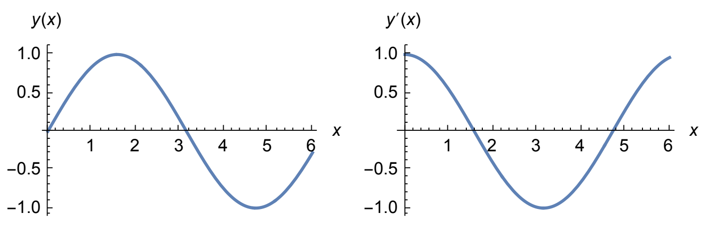
```
k     x      y(x)          y'(x)

0     0      0             1.
1     1.2    0.932039      0.362358
2     2.4    0.675463      -0.737394
3     3.6    -0.44252      -0.896758
4     4.8    -0.996165     0.0874989
5     6.     -0.279416     0.96017
```

```
py = Plot[Evaluate[y[x] /.s], {x, xmin, xmax}, AxesLabel → {x, y[x]}];

dpy = Plot[Evaluate[y′[x] /.s], {x, xmin, xmax}, AxesLabel → {x, y′[x]}];
```

```
Show[GraphicsRow[{py, dpy}]]
```



11.2.1.2 The function defined by the numeric solution of the differential equation

It is possible to use the numeric solution of a differential equation for defining a function, which may be used for getting values of the solution or plots. As an example we take a nonlinear differential equation with initial data:

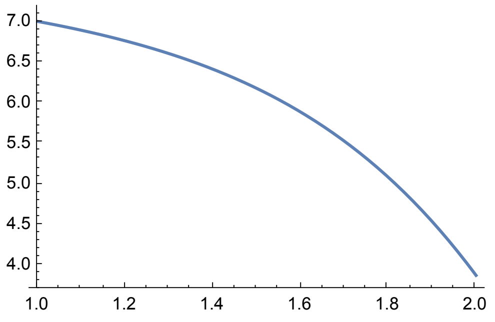
```
Clear[sol];
sol[t_] =
 First[x[t] /. NDSolve[{x''[t] == √x[t] x'[t] + 1, x[1] == 7, x'[1] == -1}, {x[t]},
    {t, 1, 2}, MaxSteps → 10^4]]
```

```
InterpolatingFunction[ Domain: {{1., 2.}}
                       Output: scalar ][t]
```

```
sol[1.5]
```

```
6.17497
```

```
Plot[sol[t], {t, 1, 2}, ImageSize → 250]
```



11.2.1.3 Motion of mass point in gravity field with and without Newtonian friction.

```
Clear[x, y, z, r, v, b]
r[t_] = {x[t], y[t]};
v[t_] = D[r[t], t];
b[t_] = D[v[t], t];
m = 1; g = 10; a = .3;
sys = Thread[ m b[t] == {0, -m g} ]
```

$\{x''[t] == 0,\ y''[t] == -10\}$

```
sysa = m b[t] ==
{0, -m g} - a v[t] Sqrt[ v[t] . v[t] ] // Thread
```

$\left\{x''[t] == -0.3\, x'[t] \sqrt{x'[t]^2 + y'[t]^2},\ y''[t] == -10 - 0.3\, y'[t] \sqrt{x'[t]^2 + y'[t]^2}\right\}$

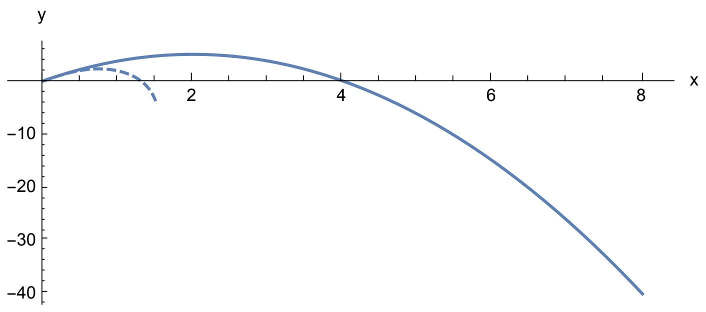
```
anf = {x[0] == 0, y[0] == 0, x'[0] == 2, y'[0] == 10};

sol = NDSolve[ Join[sys, anf], {x, y}, {t, 0, 4}] // Flatten ;

sola = NDSolve[ Join[sysa, anf], {x, y}, {t, 0, 2}] // Flatten ;
```

```
p = ParametricPlot[Evaluate[{x[t], y[t]} /.sol], {t, 0, 4}];
pa = ParametricPlot[Evaluate[{x[t], y[t]} /.sola],
    {t, 0, 2}, PlotStyle → Dashing[{0.01`}]];
```

```
Show[p, pa, AxesLabel → {"x", "y"}, AspectRatio →  0.4]
```



We assume the plane y = 0 to be the ground. So the trajectories should end there. We could find the appropriate value of the independent variable t by trial and error. A more systematic approach uses the command **`FindRoot[]`** to find these values of t. Note that one must use this command with two starting values.

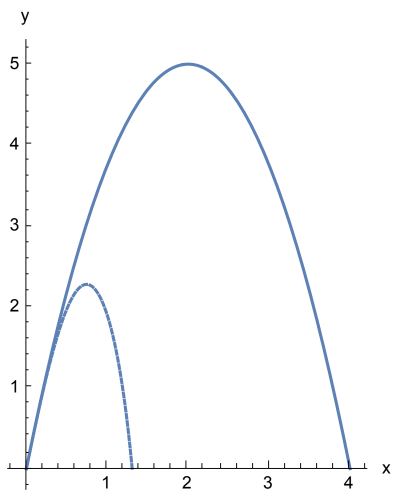
```
te  = t/.FindRoot[ Evaluate[y[t] /. sol ]  == 0., {t,2,4}]
ta  = t/.FindRoot[ Evaluate[y[t] /. sola]  == 0., {t,1,2}]
```

```
2.
```

```
1.35376
```

```
p = ParametricPlot[Evaluate[{x[t], y[t]} /.sol], {t, 0, te}];
pa = ParametricPlot[Evaluate[{x[t], y[t]} /.sola],
    {t, 0, ta}, PlotStyle → Dashing[{0.01`}]];
```

```
Show[p, pa, AxesLabel → {"x", "y"}, AspectRatio → Automatic, ImageSize → 200]
```



11.2.2 Numeric Solutions of Systems of Differential Equations for Several Dependent Variables.

```
sy  = { - (y'[t]) + x''[t]  == 0,  x'[t] + y''[t]
  == 0,  z''[t] == 0,  x[0] == 0, y[0] == 0,
 z[0]  == 0, x'[0]  == 2, y'[0]  == 0, z'[0]  == 1/3 }
```

$$\left\{-y'[t] + x''[t] == 0,\ x'[t] + y''[t] == 0,\ z''[t] == 0,\ x[0] == 0,\ y[0] == 0,\ z[0] == 0,\ x'[0] == 2,\ y'[0] == 0,\ z'[0] == \frac{1}{3}\right\}$$

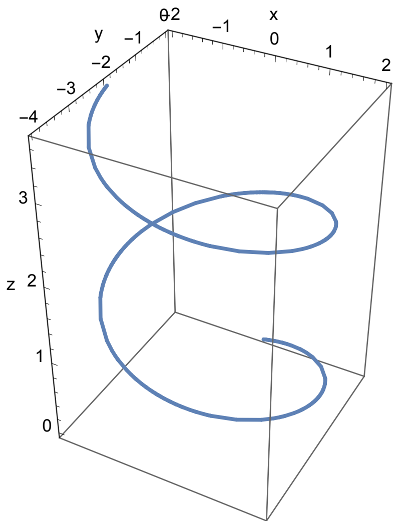
```
tmax = 11;
sosy = NDSolve[sy, {x, y, z}, {t, 0, tmax}]
```

```
{{x → InterpolatingFunction[ Domain: {{0., 11.}}
                             Output: scalar ],

  y → InterpolatingFunction[ Domain: {{0., 11.}}
                             Output: scalar ],

  z → InterpolatingFunction[ Domain: {{0., 11.}}
                             Output: scalar ]}}
```

```
ParametricPlot3D[Evaluate[{x[t], y[t], z[t]} /.sosy], {t, 0, tmax},
 BoxRatios → {1, 1, 1.5`}, AxesLabel → {"x", "y", "z"}, ImageSize → 200]
```



11.2.3 Options for NDSolve

```
?? NDSolve
```

NDSolve[*eqns*, *u*, {*x*, x_{min}, x_{max}}] finds a numerical solution to the ordinary differential equations *eqns* for the function *u* with the independent variable *x* in the range x_{min} to x_{max}.
NDSolve[*eqns*, *u*, {*x*, x_{min}, x_{max}}, {*y*, y_{min}, y_{max}}] solves the partial differential equations *eqns* over a rectangular region.
NDSolve[*eqns*, *u*, {*x*, *y*} ∈ Ω] solves the partial differential equations *eqns* over the region Ω.
NDSolve[*eqns*, *u*, {*t*, t_{min}, t_{max}}, {*x*, *y*} ∈ Ω] solves the time-dependent partial differential equations *eqns* over the region Ω.
NDSolve[*eqns*, {u_1, u_2, …}, …] solves for the functions u_i. ≫

```
Attributes[NDSolve] = {Protected}

Options[NDSolve] =
 {AccuracyGoal → Automatic, Compiled → Automatic, DependentVariables → Automatic,
  DiscreteVariables → {}, EvaluationMonitor → None, InterpolationOrder → Automatic,
  MaxStepFraction → 1/10, MaxSteps → Automatic, MaxStepSize → Automatic,
  Method → Automatic, NormFunction → Automatic, PrecisionGoal → Automatic,
  StartingStepSize → Automatic, StepMonitor → None, WorkingPrecision → MachinePrecision}
```

MaxSteps is an option to NDSolve that determines the maximum number of steps to take.

```
sy = { -(y'[t]) + x''[t] == 0,  x'[t] + y''[t]
   == 0,  z''[t] == 0,  x[0] == 0, y[0] == 0,
   z[0] == 0, x'[0] == 1, y'[0] == 0, z'[0] == 1/3 };

NDSolve[ sy, {x,y,z}, {t,0,2000}]
```

```
{{x → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ],

  y → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ],

  z → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ]}}
```

NDSolve::mxst : "Maximum number of 10000 steps reached at the point t == 1419.9526343299897`. >>

So one must increase the number of steps. Thereafter everything works properly:

```
NDSolve[ sy, {x,y,z}, {t,0,2000}, MaxSteps -> 20000]
```

```
{{x → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ],

  y → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ],

  z → InterpolatingFunction[ ⊞  Domain: {{0., 2.00×10^3}}
                                Output: scalar ]}}
```

```
??AccuracyGoal
```

AccuracyGoal is an option for various numerical operations which
specifies how many effective digits of accuracy should be sought in the final result. ≫

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

```
??PrecisionGoal
```

PrecisionGoal is an option for various numerical operations which
specifies how many effective digits of precision should be sought in the final result. ≫

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

```
??WorkingPrecision
```

WorkingPrecision is an option for various numerical operations that
specifies how many digits of precision should be maintained in internal computations. ≫

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

```
??StartingStepSize
```

StartingStepSize is an option to NDSolve and related
functions that specifies the initial step size to use in trying to generate results. ≫

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

The last four options may be very important for achieving sufficient accuracy of the solutions obtained by numeric integration of differntial equations. This is discussed in the next section.

```
?? Method
```

Method is an option for various algorithm−intensive
functions that specifies what internal methods they should use. ≫

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

Various methods of solution can be selected. Find details via Help/Find Selected/Functions/NDSolve . See also **11.2.8**

11.2.4 Checking the Accuracy of Numeric Solutions of Differential Equations.

A method to check the accuracy of numeric solutions of differential equations is to run **`NDSolve[]`** at least twice with different values assigned to the options **AccuracyGoal, PrecisionGoal** and **Working-Precision** (s. 11.2.3). Another method for such a check applicable to differential equations describing a conservative system is to compare the initial and final values of total energy. The latter check, however, is a rather weak one. A third check is to run the
system in a second run in the revers direction and to check whether the final data of the latter run agree with the initial date of the first run. All this is shown for the nonlinear system discussed in more detail in 11.2.6 .

Equations of Motion

```
Clear[t, r, x, y, vx, vy, sys, en, tmax, force]
r[t_] = {x[t], y[t]}; v[t_] = {vx[t], vy[t]};
force = {-x[t] - 2 x[t] y[t], -y[t] + y[t]^2 - x[t]^2} ;
sys = D[Join[r[t], v[t]], t]  == Join[v[t], force ] // Thread
```

$\{x'[t] == vx[t],\ y'[t] == vy[t],\ vx'[t] == -x[t] - 2\,x[t]\,y[t],\ vy'[t] == -x[t]^2 - y[t] + y[t]^2\}$

```
en = (vx[t]^2 + vy[t]^2) / 2 + (x[t]^2 + y[t]^2) / 2 +
     x[t]^2 y[t] - y[t]^3 / 3;
```

```
ad = {x[0], y[0], vx[0], vy[0]};
```