# Computer Algebra using Maple
# Part V: Extensions; some special topics.
# I/O and file handling


## Winfried Auzinger, Dirk Praetorius (SS 2018)


```
> restart:
```

## 1 Polynomial equations and systems

For polynomial equations up to order 4, **solve** finds the exact solution.

```
> p:=x^3+x^2+1;
```

$$p := x^3 + x^2 + 1$$

```
> solve(p,x);
```

$$-\frac{(116+12\sqrt{93})^{1/3}}{6} - \frac{2}{3(116+12\sqrt{93})^{1/3}} - \frac{1}{3}, \frac{(116+12\sqrt{93})^{1/3}}{12} + \frac{1}{3(116+12\sqrt{93})^{1/3}} - \frac{1}{3} + \frac{I\sqrt{3}\left(-\frac{(116+12\sqrt{93})^{1/3}}{6} + \frac{2}{3(116+12\sqrt{93})^{1/3}}\right)}{2}, \frac{(116+12\sqrt{93})^{1/3}}{12} + \frac{1}{3(116+12\sqrt{93})^{1/3}} - \frac{1}{3} - \frac{I\sqrt{3}\left(-\frac{(116+12\sqrt{93})^{1/3}}{6} + \frac{2}{3(116+12\sqrt{93})^{1/3}}\right)}{2}$$

```
> evalf([%]);
```

$$[-1.465571232,\ 0.2327856159 - 0.7925519930\,I,\ 0.2327856159 + 0.7925519930\,I]$$

```
> p:=x^4+x+1;
```

$$p := x^4 + x + 1$$

```
> sol:=solve(p,x);
```

$$sol := RootOf(\_Z^4 + \_Z + 1, index = 1), RootOf(\_Z^4 + \_Z + 1, index = 2), RootOf(\_Z^4 + \_Z + 1, index = 3), RootOf(\_Z^4 + \_Z + 1, index = 4)$$

In more complicated cases, you may get a **RootOf** object representing the solution in an implicit way. **allvalues** tries to evaluate explicitly:

```
> allvalues([sol]);
```

$$\left[\frac{\sqrt{6}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}}{12}\right.$$

$$-\frac{1}{12}\left(-\left(6\left(\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\left(108+12\,I\sqrt{687}\right)^{2/3}+12\sqrt{6}\left(108\right.\right.\right.\right.$$

$$\left.\left.\left.\left.+12\,I\sqrt{687}\right)^{1/3}+48\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\right)\right)\Big/\left(\left(108+12\,I\sqrt{687}\right)^{1/}\right.$$

$$\left.\left.{}^{3}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\right)\right)^{1/2},-\frac{\sqrt{6}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}}{12}$$

$$-\frac{1}{12}\left(\sqrt{6}\right.$$

$$\left(\left(-\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\left(108+12\,I\sqrt{687}\right)^{2/3}+12\sqrt{6}\left(108+12\,I\sqrt{687}\right)^{1/3}\right.\right.$$

$$\left.\left.-48\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\right)\Big/\left(\left(108+12\,I\sqrt{687}\right)^{1/}\right.$$

$$\left.\left.{}^{3}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\right)\right)^{1/2}\Bigg),\ -\frac{\sqrt{6}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}}{12}$$

$$+\frac{1}{12}\left(\sqrt{6}\right.$$

$$\left(\left(-\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\left(108+12\,I\sqrt{687}\right)^{2/3}+12\sqrt{6}\left(108+12\,I\sqrt{687}\right)^{1/3}\right.\right.$$

$$-48\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\Bigg)\Bigg/\Bigg(\left(108+12\,I\sqrt{687}\right)^{1/}$$

$$3\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\Bigg)\Bigg)\Bigg)^{1/2}\Bigg),\frac{\sqrt{6}\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}}{12}$$

$$+\frac{1}{12}\Bigg(-\Bigg(6\Bigg(\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\left(108+12\,I\sqrt{687}\right)^{2/3}+12\sqrt{6}\,(108$$

$$+12\,I\sqrt{687}\,)^{1/3}+48\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\Bigg)\Bigg)\Bigg/\Bigg(\left(108+12\,I\sqrt{687}\right)^{1/}$$

$$3\sqrt{\frac{\left(108+12\,I\sqrt{687}\right)^{2/3}+48}{\left(108+12\,I\sqrt{687}\right)^{1/3}}}\Bigg)\Bigg)\Bigg)^{1/2}\Bigg]$$

```
> evalf([%]);
```

$$[[0.7271360849+0.9340992892\,I,\ -0.7271360851+0.4300142886\,I,\ -0.7271360847-0.4300142888\,I,\ 0.7271360849-0.9340992890\,I]]$$

For higher degree, you cannot expect to obtain a solution in general.

```
> p:=x^5+x^3+1;
```

$$p := x^5+x^3+1$$

```
> sol:=solve(p,x);
```

$$sol := RootOf\left(\_Z^5+\_Z^3+1,\ index=1\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=2\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=3\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=4\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=5\right)$$

```
> allvalues([sol]);
```

$$\left[RootOf\left(\_Z^5+\_Z^3+1,\ index=1\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=2\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=3\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=4\right),\ RootOf\left(\_Z^5+\_Z^3+1,\ index=5\right)\right]$$

Applying **evalf** to RootOf objects usually finds numerical values:

```
> evalf([sol]);
```

$$[0.6366631068 + 0.6647015651\,I,\ -0.2178532194 + 1.166951246\,I,\ -0.8376197748,\ -0.2178532194 - 1.166951246\,I,\ 0.6366631068 - 0.6647015651\,I]$$

Systems of polynomial equations are even more complicated.
The algorithm implemented in solve tries to reduce the problem to
a single polynomial equation by special elimination techniques.
Success is not guaranteed.

```
> p[1]:=x^2+x*y+y^2;
  p[2]:=x^3-x*y^2+1;
```

$$p_1 := x^2 + x\,y + y^2$$

$$p_2 := x^3 - x\,y^2 + 1$$

```
> solve([p[1],p[2]],[x,y]);
```

$$\left[\left[x = RootOf\left(3\_Z^6 + 3\_Z^3 + 1\right),\ y = 3\,RootOf\left(3\_Z^6 + 3\_Z^3 + 1\right)^4 + RootOf\left(3\_Z^6 + 3\_Z^3 + 1\right)\right]\right]$$

```
> sol:=allvalues(%);
```

$$sol := \left[\left[x = \frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{6},\ y = \frac{\left(-108 + 36\,I\sqrt{3}\right)^{4/3}}{432} + \frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{6}\right]\right],$$

$$\left[\left[x = -\frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12},\ y = 3\left(-\frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12}\right)^4 - \frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12}\right]\right],$$

$$\left[\left[x = -\frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12},\ y = 3\left(-\frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12}\right)^4 - \frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} + \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12}\right]\right],$$

$$\left[\left[x = -\frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12},\ y = 3\left(-\frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12}\right)^4 - \frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{12}\right]\right],$$

$$\left[\left[x = -\frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12},\ y = 3\left(-\frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12}\right)^4 - \frac{\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12} - \frac{I\sqrt{3}\left(-108 + 36\,I\sqrt{3}\right)^{1/3}}{12}\right]\right],$$

$$\left[\left[x = \frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{6},\ y = \frac{\left(-108 - 36\,I\sqrt{3}\right)^{4/3}}{432} + \frac{\left(-108 - 36\,I\sqrt{3}\right)^{1/3}}{6}\right]\right]$$

```
> numelems([sol]);
```

$$6$$

```
> evalf([sol]);
```

$$[[x = 0.5352384294 + 0.6378723211\,I,\ y = -0.8200328476 + 0.1445939185\,I]],\ [[x = 0.2847944198 + 0.7824662375\,I,\ y = 0.5352384292 - 0.6378723210\,I]],\ [[x = -0.8200328492 + 0.1445939165\,I,\ y$$

$= 0.2847944198 - 0.7824662380\,I]], [[x = -0.8200328492 - 0.1445939165\,I, y = 0.2847944198 + 0.7824662380\,I]], [[x = 0.2847944198 - 0.7824662375\,I, y = 0.5352384292 + 0.6378723210\,I]], [[x = 0.5352384294 - 0.6378723211\,I, y = -0.8200328476 - 0.1445939185\,I]]]$

Application: Find an approximation to the solution of a differential equation y'(t) = f(y(t))
starting from y(0) which only uses evaluation of f but no differentiations.
Think of a short interval [0,t] (in practice, this is repeated iteratively over several intervals).

First we compute the Taylor polynomial of y(t) up to degree 2 about t=0,
and express it using derivatives of f (chain rule).

```
> tay_y := convert(taylor(y(t),t=0,3),polynom);
```

$$tay\_y := y(0) + \mathrm{D}(y)(0)\,t + \frac{\mathrm{D}^{(2)}(y)(0)\,t^2}{2}$$

Here, due to y'=f(y) we have y'(0)=f(y(0)) and y''(0) = (d/dt)f(t,y(t)) at t=0.

```
> d1y := f(y(0));
```

$$d1y := f(y(0))$$

```
> d2y := diff(f(y(t)),t);
```

$$d2y := \mathrm{D}(f)(y(t))\left(\frac{\mathrm{d}}{\mathrm{d}t}\,y(t)\right)$$

```
> d2y := subs(diff(y(t),t)=f(y(t)),d2y); # use differential equation!
```

$$d2y := \mathrm{D}(f)(y(t))\,f(y(t))$$

```
> d2y := subs(t=0,d2y);
```

$$d2y := \mathrm{D}(f)(y(0))\,f(y(0))$$

This gives the following Taylor expansion:

```
> tay_y := subs(D(y)(0)=d1y,(D@@2)(y)(0)=d2y,tay_y);
```

$$tay\_y := y(0) + f(y(0))\,t + \frac{\mathrm{D}(f)(y(0))\,f(y(0))\,t^2}{2}$$

The derivative of D(f) (or even higher derivatives) is not available in many practical
situations. For **approximating this,** we use the following ansatz:

```
> appr := y(0) + a[1]*t*f(y(0)) + a[2]*t*f(y(0)+a[3]*t*f(y(0)));
```

$$appr := y(0) + a_1\,t\,f(y(0)) + a_2\,t\,f\left(y(0) + a_3\,t\,f(y(0))\right)$$

Now we try to determine optimal parameters a[1], a[2] and a[3].
Taylor expansion about t=0 gives

```
> tay_appr:=convert(taylor(appr,t=0,3),polynom);
```

$$tay\_appr := y(0) + \left(a_1 f(y(0)) + a_2 f(y(0))\right)t + a_2\,\mathrm{D}(f)(y(0))\,a_3 f(y(0))\,t^2$$

Now we consider the difference between these two Taylor expanions,
and **collect,** i.e., arrange with respect to powers of t:

```
> difference:=collect(tay_appr-tay_y,t);
```

$$difference := \left(a_2\,\mathrm{D}(f)\,(y(0))\,a_3 f(y(0)) - \frac{\mathrm{D}(f)\,(y(0))\,f(y(0))}{2}\right) t^2 + \left(a_1 f(y(0)) + a_2 f(y(0)) - f(y(0))\right) t$$

Extract coefficients using **coeff:**

```
> coe[1],coe[2]:=coeff(difference,t,1),coeff(difference,t,2);
```

$$coe_1, coe_2 := a_1 f(y(0)) + a_2 f(y(0)) - f(y(0)),\ a_2\,\mathrm{D}(f)\,(y(0))\,a_3 f(y(0)) - \frac{\mathrm{D}(f)\,(y(0))\,f(y(0))}{2}$$

```
> coe[1]:=simplify(coe[1]/f(y(0)));
```

$$coe_1 := a_1 + a_2 - 1$$

```
> coe[2]:=simplify(coe[2]/(D(f)(y(0))*f(y(0))));
```

$$coe_2 := a_2 a_3 - \frac{1}{2}$$

```
> solve([coe[1],coe[2]],[a[1],a[2],a[3]]);
```

$$\left[\left[a_1 = \frac{-1 + 2\,a_3}{2\,a_3},\ a_2 = \frac{1}{2\,a_3},\ a_3 = a_3\right]\right]$$

This is the general set of solutions (1 free parameter). This case is very simple; requiring higher approximation orders gives rather complicated systems of polynomial equations in the parameters a[i]. (--> so-called Runge-Kutta methods.)

# 2 Differential equations

Maple can exactly (or numerically) solve differential equations and systems of differential equations (see Part II). The basic solver is **dsolve**. Here are two more examples:

**Example:** A nonlinear first order equation

```
> de := D(y)(t) = t*y(t)^2;
```

$$de := \mathrm{D}(y)(t) = t\,y(t)^2$$

```
> dsolve(de,y(t));
```

$$y(t) = \frac{2}{-t^2 + 2\,\_C1}$$

Or with a given initial values:

```
> ic := y(0)=1;
```

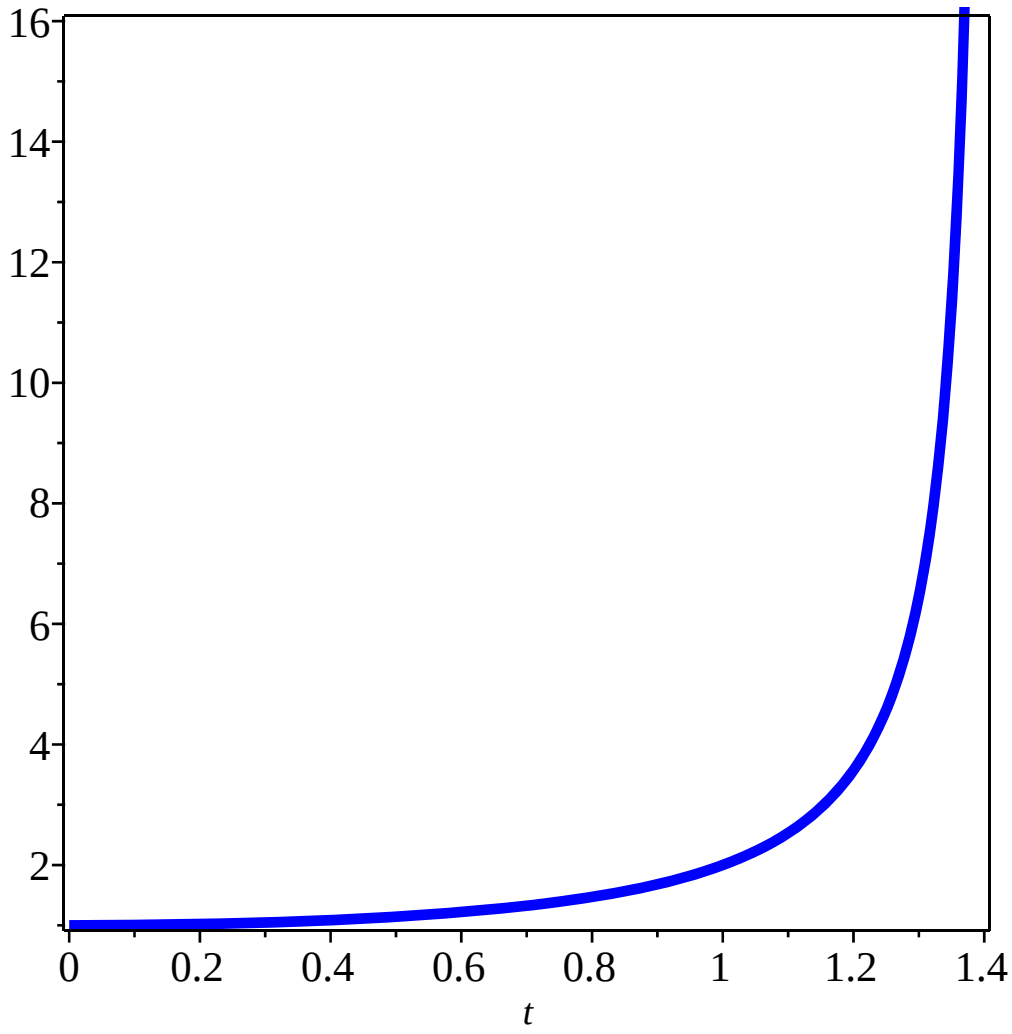
$$ic := y(0) = 1$$

```
> dsolve([de,ic],y(t)); # solution blows up to infinity at t=sqrt(2)
```

$$y(t) = -\frac{2}{t^2 - 2}$$

```
> assign(%);
```

```
> plot(y(t),t=0..1.4,thickness=4,axes=boxed,color=blue);
```




**Example:** Describing a 3D spiral via a system of 3 differential equations

```
> A:=Matrix([[0.2,2,3],
             [-2,0.2,1],
             [-3,-1,0.2]]); # 0.2*identity + antisymmetric
```

$$A := \begin{bmatrix} 0.2 & 2 & 3 \\ -2 & 0.2 & 1 \\ -3 & -1 & 0.2 \end{bmatrix}$$

```
> de:=seq(D(y[i])(t)=add(A[i,j]*y[j](t),j=1..3),i=1..3);
```

$$de := \mathrm{D}(y_1)(t) = 0.2\,y_1(t) + 2\,y_2(t) + 3\,y_3(t),\ \mathrm{D}(y_2)(t) = -2\,y_1(t) + 0.2\,y_2(t) + y_3(t),\ \mathrm{D}(y_3)(t) = -3\,y_1(t) - y_2(t) + 0.2\,y_3(t)$$

```
> var:=seq(y[i](t),i=1..3);
```

$$var := y_1(t), y_2(t), y_3(t)$$

```
> ic:=y[1](0)=1,y[2](0)=1,y[3](0)=0;
```

$$ic := y_1(0) = 1, y_2(0) = 1, y_3(0) = 0$$

```
> dsolve([de,ic],[var]);
```

$$\left\{ y_1(t) = -\frac{e^{\frac{t}{5}}\left(-\frac{80\cos\left(\sqrt{14}\,t\right)}{7} - \frac{10\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} + \frac{10}{7}\right)}{10}, y_2(t) = -\frac{e^{\frac{t}{5}}\left(-\frac{40\cos\left(\sqrt{14}\,t\right)}{7} + \frac{10\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} - \frac{30}{7}\right)}{10}, y_3(t) = e^{\frac{t}{5}}\left(\frac{2\cos\left(\sqrt{14}\,t\right)}{7} - \frac{2\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} - \frac{2}{7}\right) \right\}$$

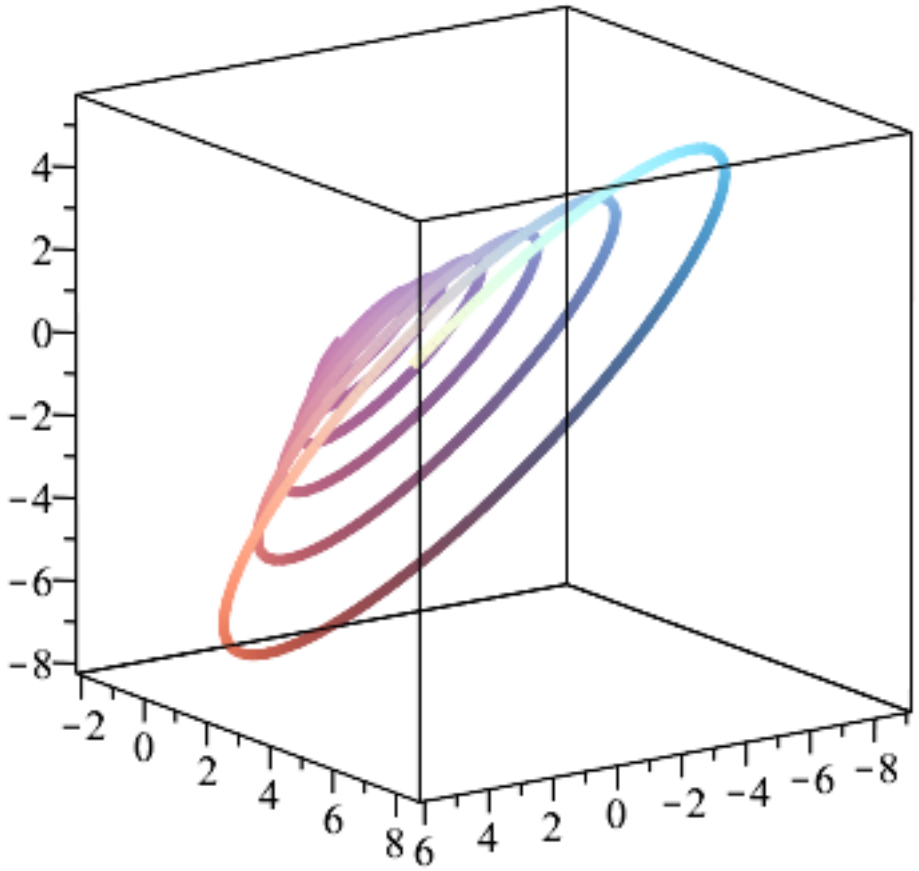
```
> map(assign,%); # 'map' is explained in more detail below
```

$$\varnothing$$

```
> y[1](t),y[2](t),y[3](t);
```

$$-\frac{e^{\frac{t}{5}}\left(-\frac{80\cos\left(\sqrt{14}\,t\right)}{7} - \frac{10\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} + \frac{10}{7}\right)}{10}, -\frac{e^{\frac{t}{5}}\left(-\frac{40\cos\left(\sqrt{14}\,t\right)}{7} + \frac{10\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} - \frac{30}{7}\right)}{10}, e^{\frac{t}{5}}\left(\frac{2\cos\left(\sqrt{14}\,t\right)}{7} - \frac{2\sqrt{14}\,\sin\left(\sqrt{14}\,t\right)}{7} - \frac{2}{7}\right)$$

```
> plots[spacecurve]([y[1](t),y[2](t),y[3](t)],t=0..10,thickness=4,axes=
  boxed,numpoints=1000);
```




A numerical solution (based on an adaptive Runge-Kutta method):
With option **numeric** activated, **dsolve** returns a procedure:

```
> y:='y':
  sol:=dsolve([de,ic],[var],numeric);
```

$$sol := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$

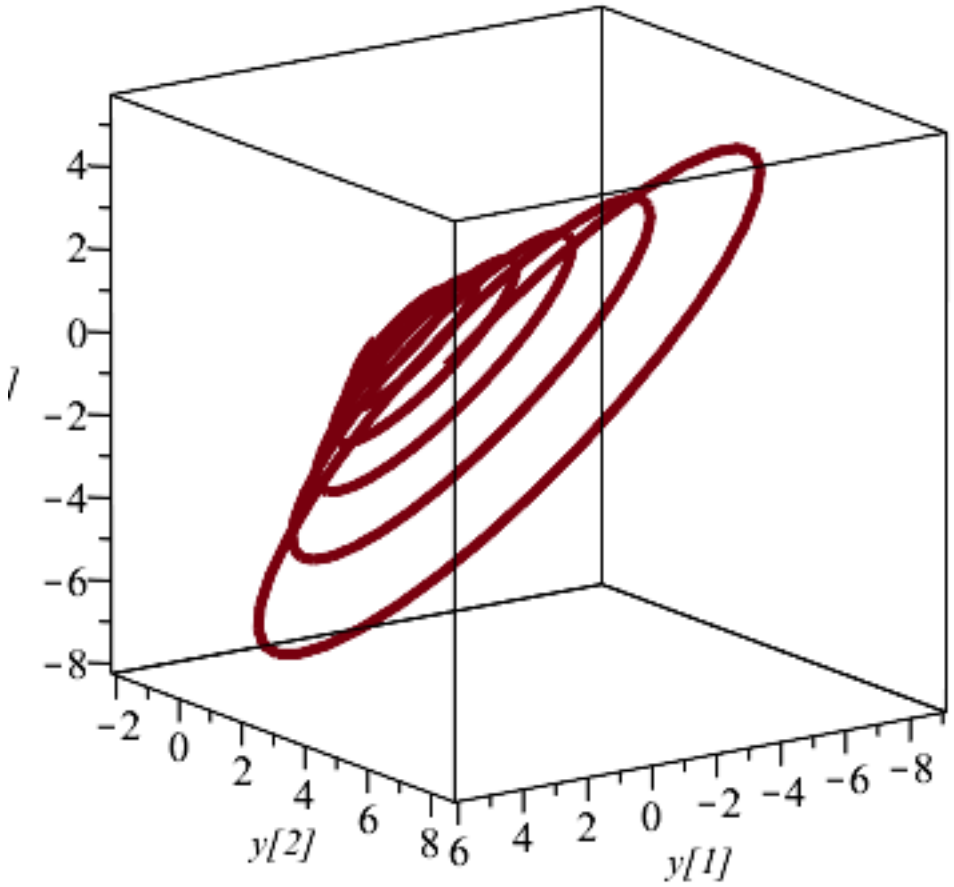
Evaluate solution at t=1:

```
> sol(1);
```

$$[t = 1., y_1(t) = -1.695183820, y_2(t) = 0.3161166973, y_3(t) = 0.1003641977]$$

**odeplot** calls the solution procedure:

```
> plots[odeplot](sol,[y[1](t),y[2](t),y[3](t)],t=0..10,thickness=4);
```




# 3 Some further useful commands

```
> restart;
```

## 3.1 map

A scalar function can usually not be applied to a data structure like Vector or Matrix (this depends on the function at hand):

```
> v:=Vector([0,1,I]);
```

$$v := \begin{bmatrix} 0 \\ 1 \\ I \end{bmatrix}$$

```
> v^2; exp(v); # does not work
Error, (in rtable/Power) exponentiation operation not defined for Vectors
Error, invalid input: exp expects its 1st argument, x, to be of type
algebraic, but received Vector(3, {(1) = 0, (2) = 1, (3) = I})
> Re(v);  # however, this works
```

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Workaround: The command **map** applies a function to all elements of a data structure.

**Examples:**

```
> map(t->t^2,v);
```

$$\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

```
> map(exp,v);
```

$$\begin{bmatrix} 1 \\ e \\ e^{I} \end{bmatrix}$$

See also: **? map2**

### 3.2 collect, coeff

Functions for ordering polynomial expressions and extracting coefficients.

```
> p:=mul((x-i),i=1..5);
```

$$p := (x-1)\,(x-2)\,(x-3)\,(x-4)\,(x-5)$$

```
> p:=collect(p,x);
```

$$p := x^5 - 15\,x^4 + 85\,x^3 - 225\,x^2 + 274\,x - 120$$

```
> seq(coeff(p,x,i),i=0..degree(p));
```

$$-120,\ 274,\ -225,\ 85,\ -15,\ 1$$

### 3.3 subs, more generally

**subs** to some extent can replace subexpressions by others. Examples:

```
> subs(0=2,Vector([0,1]))
```

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

```
> taylor(f(x(t)),t=0,2);
```

$$f(x(0)) + \mathrm{D}(f)\,(x(0))\,\mathrm{D}(x)(0)\,t + \mathrm{O}(t^2)$$

```
> subs(x(0)=1,D(x)(0)=2,%);
```

$$f(1) + 2\,\mathrm{D}(f)\,(1)\,t + \mathrm{O}(t^2)$$

# 4 Interactive input

Use **readline(terminal)** to enter data interactively.
(readline can also read from an external data file, see help.)

A call to readline produces a string. This can be converted using **sscanf**.
Note that sscanf produces a list.

**Example:**

```
> rm:=proc() uses LinearAlgebra:
  description "generate nxn random matrix; n entered by user";
  local A,i,n,s:
  print("Enter dimension n >"):
  s:=readline(terminal): # this produces a string
  n:=sscanf(s,"%a")[1]:
  print(n):
  return RandomMatrix(n,n):
  end proc:
> rm();
```

"Enter dimension n >"

3

$$\begin{bmatrix} 27 & 99 & 92 \\ 8 & 29 & -31 \\ 69 & 44 & 67 \end{bmatrix}$$

# 5 Source code on external files

Text files containg Maple code (procedures, or complete worksheets):
Use file tyle **.mpl**

For generating code, use any text editor, or **save :**

```
> p1 := proc() return NULL end;
```

*p1* := **proc**( ) **return** *NULL* **end proc**

```
> p2 := proc() print("Hello World") end;
```

*p2* := **proc**( ) *print*("Hello World") **end proc**

```
> save p1,p2,"my_code.mpl";
> unassign(p1,p2);
> read "my_code.mpl";
```

*p1* := **proc**( ) **return** *NULL* **end proc**

*p2* := **proc**( ) *print*("Hello World") **end proc**

```
> p2();
```

"Hello World"

**.mpl** files can also be used to run worksheets in 'batch mode' ,
assuming no interactive input is required.

Typical syntax (assuming the maple executable is found on the path):

**maple** *my_code.mpl*

For larger jobs one will start execution in background and redirect the output:

**maple** *my_code.mpl* > *my_output.txt* &

For saving a complete worksheet in .mpl format, use the menu: **File / Export As / ...**

# 6 Flexible argument lists

For a procedure containing no formal arguments,
after calling it, **_passed** will contain arguments actually passed,
and **_npassed** contains the number of arguments passed.

**Example:**

```
> p:=proc()
>   local i;
>   printf("%d ",_npassed);
>   for i from 1 to _npassed do
        printf("%a ",_passed[i])
    end do;
    printf("\n");
> end proc:
> p(); p(a); p(a,b);
0
1 a
2 a b
```

Moreover, optional parameters, keyword parameters, etc., are supported.

See:

```
> ? using_parameters
> ? _passed
```

# 7 Useful packages

```
> ? index[package]
```

Here is just a short list - find out yourself, and check if you require something special.

**combinatorics, combstruct**
**geometry**
**Grid** (for parallelization on a multi-core machine)
**LinearAlgebra**
**Optimization**
**orthopoly**
**Physics**
**plots**
**plottools**
...

and many others.

# 8 Final remarks

-- For the mathematician, programming solutions is the major benefit
you draw from a system implementing computer mathematics.
Instead of performing the same computation again and again (which was trained
in education in early years), write a code for its solution and refine and extend it if
necessary. Do it as genererally as necessary and reasonable.
Coding an algorithm usually also helps in understanding it.

-- Standard methods from analysis and algebra should still be taught and learned in a conventional way,
doing them by hand. Whether computer algebra is used in parallel or later is not essential,
but not form the very beginning, completely replacing handwork.
(Different models are used at different universities.) As soon as you know and understand,
you can resort to computer algebra as you advance.

-- In general, memorizing details of a more complicated formula or an algorithm is usually not very useful,
What is really useful is that you understand what is behind.

-- Use Maple for all kinds of tasks and extend your knowledge. You will benefit.
If you want to solve a problem, try check if a solution methods is available or what needs to be done.

-- Symbolic and numerical algorithms are not completely separated worlds.
Often a symbolic computation serves as a prepocessor to a numerical calculation,
e.g. for generating coefficients of an approximation formula used in the sequel.

-- As you have seen, Maple performs symbolic as well as numerical, or mixed, computations.
The MATLAB system (which you also study in 'Computer Mathematics') is an alternative
with a strong emphasis on numerical compuation and visualization, especially numerical linear algebra,
and it comes with may special packages, called toolboxes. Furthermore, a computer algebra kernel
(**mupad**) has recently been integrated. This enables you to do certain symbolic
manipulations (e.g. differentiation using symbolic variables) directly in MATLAB.

-- **What else** ? (more advanced topics, not covered here:)

--- Interactive debugger

--- Modules (object orientation)

--- Designing packages

--- calling external code

--- Maplets

.... .....

========== end of Part V ==