A Course of ODE with a CAS

To Miguel de Guzmán (In Memoriam)

Alfonsa GARCÍA
Universidad Politécnica de Madrid.
Departamento de Matemática Aplicada. E.U. Informática.
Cr. Valencia Km.7, 28031 Madrid (Spain)
e-mail: garcial@eui.upm.es

Francisco GARCÍA
Universidad Politécnica de Madrid.
Departamento de Matemática Aplicada. E.U. Informática.
Cr. Valencia Km.7, 28031 Madrid (Spain)
e-mail: gmazario@eui.upm.es

Gerardo RODRÍGUEZ
Universidad de Salamanca.
Departamento de Matemática Aplicada. E.P.S. de Zamora.
Avda. Requejo 33, 49022 Zamora (Spain)
e-mail: gerardo@usal.es

Agustín DE LA VILLA
Universidad Pontificia Comillas.
Departamento de Matemática Aplicada y Computación. ETSI (ICAI).
Alberto Aguilera, 23. 28015 Madrid (Spain)
e-mail: avilla@upco.es
and
Universidad Politécnica de Madrid
Departamento de Matemática Aplicada. E.U.I.T.I
Ronda de Valencia, 3. 28012 Madrid (Spain)

KEY WORDS: Computer Algebra Systems, Mathematical Education, Ordinary Differential Equations, Numerical Methods.

ABSTRACT

In this lecture we present a proposal for a basic course of Ordinary Differential Equations (ODE). In that proposal, using a Computer Algebra System (CAS), we try to show, in a general and integrated way, exact and approximate methods to solve ODE.

The systematic utilization of all possibilities of CAS (the CAS used, like Maple, Mathematica, etc. is not important) allows a deep change in the contents and in the methodology to be used in the lectures regarding traditional teaching where two different ODE courses are proposed. The first one is devoted to exact resolution; it includes a wide number of methods to solve ODE, in an exact way. The second one deals with numerical methods.

In our integrated course we include new contents which is not possible to handle with a traditional way of teaching. In order to enhance our thesis, we will propose several examples (including as annex documents) concerning numerical integration methods, Picard theorem, some deduction of algorithms, graphical capacities, qualitative theory, etc.

1. Introduction

Traditionally, regardless of the university degree being studied (engineering, mathematics, basic science, etc.) syllabus often contains two differentiated undergraduate courses on ordinary differential equations.

The first one is a general course including the following topics:

- Basic terminology.
- Different methods for the exact integration of ODEs.
- Modelling problems.

These modelling problems are more intensely directed towards interest areas for students: the modelling of physical problems, mechanical problems, problems in electricity, control problems, etc.

The methodology employed to impart such course also tends to be fairly traditional and the problems explored in the different fields are usually "black/white-board" problems. Many university manuals follow this scheme.

After taking one of such courses many students think that all ODEs have exact solution. It is very difficult for them to understand that the number of ODEs in which closed form solution is possible is very small.

The second one is a course dealing with numerical methods to integrate ODEs, clearly focused on the problems of the computation and programming of numerical methods. The practical lessons involve programming of different numerical methods in an appropriate language (like C) and use numerical program libraries for solving problems involving differential equations.

The disparity of focuses and the methodology to be used is such that, in Spain, both subjects are usually assigned to different Departments.

The advent of Computer Algebra Systems has in many cases failed to lead to any reflection as regards the organisation of the subjects or any important methodological changes in the teaching of ODE.

2. Our Proposal

Below we offer a proposal for the integration of both points of view in just one course including exact and numerical method for solving ODEs. The main goals of the course will be the following:

- To know basic terminology.
- To use a few exact methods to solve specific ODE, mainly linear ODE.
- To interpret the information as much as possible only regarding the ODE (direction field, qualitative analysis, isocline lines, asymptotic behaviour, etc.).
- To be able to implement some numerical methods and to interpret the obtained results.

The use of the different capabilities of CASs available today allows this alternative teaching, currently in use in our respective Universities.

It is also pertinent to point out that the methodological change proposed is not related to any specific CAS. Accordingly, we present examples using both Maple and Mathematica in an attempt to show that, regardless of the CAS selected. The important thing is the new teaching's focus proposed.

2.1. Using the CAS to introduce concepts and theoretical results

The CAS's graphical power allows us to visualize the isocline curves associated with a differential equation and to represent some solutions easily, even though the students still do not know how to obtain them in an explicit way, which is very illustrative of what is represented by the ODE $y' = f(x, y)$. If we connect each isocline with another one by segments whose slope is the value of the isocline, we are showing the Euler method that will be seen in numerical methods. In Example 1 (file ODE1.mws) we perform a graphic treatment of different situations that may arise in the qualitative analysis of differential equations.

The CAS calculating power also allows to highlight important theoretical results. For example, it is possible to get the sequence of functions convergent to the solution of a Cauchy problem given by the proof of Picard's existence and unicity theorem, see Example 2 (file ODE2.mws). This method cannot be approached in a practical way in traditional teaching and it is one of the advantages of our alternative.

2.2. Our course includes a few exact methods

One of the most striking proposed changes with respect to conventional teaching involves the presentation of the different exact methods of integration. In view of the strength of symbolic computation of CAS, we believe that it is not necessary to offer an exhaustive list of exact integration methods; instead, we shall focus mainly on the algorithmic process involved in the integration of linear differential equations, very important in practical applications, see Example 3 (file ODE3.mws). The CAS can solve most of the classical differential equations with no trouble, even identifying the type of equation and then integrating it. Accordingly, attempts should be made to foster the use of the solvers of differential equation of each CAS (**dsolve** command) to obtain the solution, even though it is obtained blindly.

2.3. Modelling Problems

From the point of view of modelling, systematic use of a CAS allows the computer to be used as a true experimental laboratory in mathematics, such that by modifying the initial conditions and the different terms of the differential equations under study it is possible to visualize the different types of behaviour of the physical systems being modelled. For example, when analyzing a mass-spring system, it is easy to present –in an experimental way– the successive types of systems that may arise: damped systems, over-damped systems, resonance, etc. So they can simulate all possible situations. The students are guided through the different cases. In Example 4 (file ODE4.nb) we present an approach to study of mass-spring mechanisms.

It is also possible to model the nature of the trajectories of conservative systems, just depending on the energy value. Thus, in Example 5 (file ODE5.nb), we analyze the stability of non-linear hard and soft springs.

2.4. Numerical Methods

The programming language of a CAS as Mathematica or Maple is very suitable to implement easy numerical procedures for solving ODEs. So the students can do it. Thus, using the program libraries of the CAS itself or programming the different algorithms, it is possible to reconstruct the different numerical methods, both single-step and multi-step methods. The main advantages provided by a CAS over traditional programming language stem from the possibility of including symbolic calculations in the programs and it is also possible to summarize these in two aspects: On one hand, the CAS allows us to carry out in a simple way valid procedures for Taylor methods of any degree, giving the specific equation as the input parameter, and thereby avoiding the technical disadvantage of deriving symbolically in each individual ODE, see Example 6 (file ODE6.mws). On the other hand, the application of the concept on which multi-step methods are based by integration of suitable interpolation polynomials permits the direct deduction of the different formulas, thereby extending students' understanding of the classic multi-step methods.

Bibliography

[1] Martha L. Abell and James P. Braselton, *Differential Equations with Mathematica*, Elsevier Academic Press, Burlington, MA, 2004.

[2] Robert L. Borelli and Courtney S. Coleman, *Ecuaciones diferenciales: una perspectiva de modelación*, Oxford University Press, Mexico, 2002.

[3] Charles H. Edwards and David E. Penney, *Differential Equations and Boundary value Problems: computing and modeling*, Englewood Cliffs, New Jersey, 1996.

[4] Alfonsa García López, Francisco García Mazarío, Gerardo Rodríguez, Agustín de la Villa y otros, *Ecuaciones diferenciales ordinarias*, Clagsa, Madrid, preprint.

[5] Miguel de Guzmán, *Ecuaciones diferenciales ordinarias: teoría de estabilidad y control*, Alhambra, Madrid, 1980.

[6] Darren Redfern and Edgar Chandler, *Maple ODE Lab Book*, Springer-Verlag, New York, 1996.

[7] Stephen Wolfram, *The Mathematica Book*, 5^{th} *ed.*, Wolfram media, Champaign, IL, 2003.

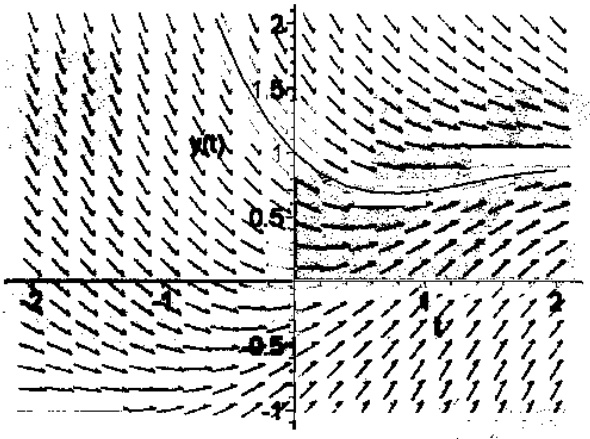
[8] Dennis G. Zill, *Ecuaciones diferenciales con aplicaciones de modelado,* 7^{a} *ed.*, Thomson, Mexico, 2002.

EXAMPLE 1: Introducing ODE and ODES

```
> with(DEtools):
```

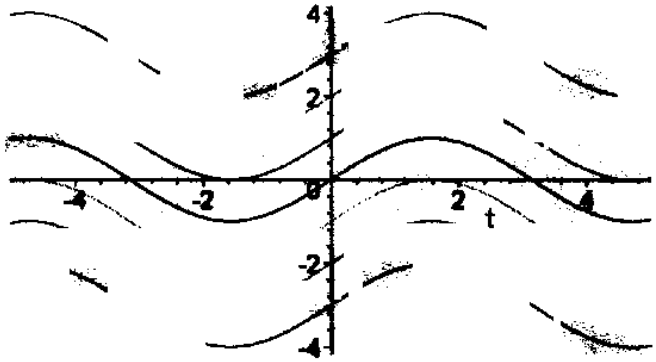
Given a ODE, we can plot its direction field which consists of a grid of arrows that are tangent to solution curves. Also we can overplot a particular solution.

```
> DEplot(diff(y(t),t)=sin(t)-y(t),
  y(t),t=-2..2,y=-1..2,[[y(0)=1]], linecolor=black);
```



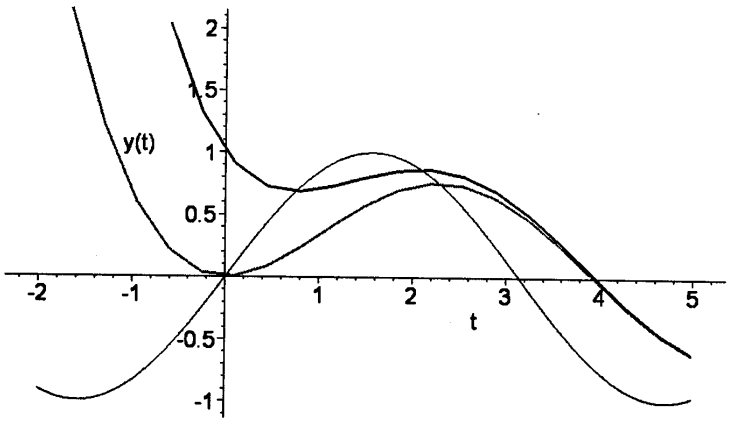
The isoclines of $y' = \sin(t) - y$ are $\sin(t) - y = c$.

```
> plot([seq(sin(t)+c,c=-3..3)],t=-5..5);
```



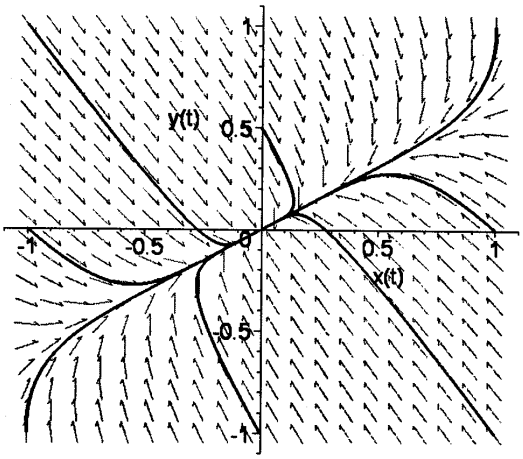
A plot of two solutions and an isocline curve, concretely the nullcline. We can see that the slope of solutions in the intersecting points with nullcline are, obviously cero.

```
> g1:=DEplot(diff(y(t),t)=sin(t)-y(t),
  y(t),t=-2..5,y=-1..2,[ [y(0)=0], [y(0)=1]],
  linecolor=[green,blue], arrows='NONE'):
> g2:=plot(sin(t), t=-2..5, color=black):
> plots[display](g1,g2);
```



Given a first order system of differential equations we will plot solution curves in the phase plane, provided the system is determined to be autonomous.

```
> DEtools[phaseportrait]([D(x)(t)=-x(t)+y(t),D(y)(t)=x(t)-2*y(t)],[x
  (t),y(t)],t=0..10,
  [[x(0)=0,y(0)=1/2],[x(0)=1,y(0)=0],[x(0)=1,y(0)=1],[x(0)=-1,y(0)=-
  1],[x(0)=1,y(0)=-1],[x(0)=-1,y(0)=1],[x(0)=-1,y(0)=-0],[x(0)=0,y(0
  )=-1]],stepsize=.05, \
  scene=[x(t),y(t)],linecolour=black);
```



Students can introduce new initial conditions for its experimentation.

```
[ >
```

Maple Worksheet ODE2.MWS

EXAMPLE 2: The Picard's iterations

The proof of Picard's Theorem provides a sequence of functions which converges to the solution of the Cauchy problem:

$y' = f(t, y), y(t0) = y0.$

This sequence is:

$$y_n(t) = y0 + \int_{t0}^{t} f(x, y_{n-1}(x))\, dx$$

Students can program with **Maple** the generation of this sequence:

```
> Picard:=proc(f,t0,y0,n,t1)
    local sequence, j,YN;
    YN:=y0;
    print(y0);
    for j to n do
      YN:=y0+int(f(t,YN),t=t0..c);#the most important instruction

    if (has(YN,int)) then
       print (YN-y0);
       RETURN (`Maple can't continue`);
      fi;
      YN:=sort(subs(c=t,YN));
         print(YN);
      sequence[j]:=YN;
    od;
    plot({seq(sequence[j],j=0..n)},
  t=t0..t1,title=`Picard Iterations`):
   end:
```

Example:

```
> f:=(t,y)->1+y^2:
> Picard(f,0,0,4,2);
```

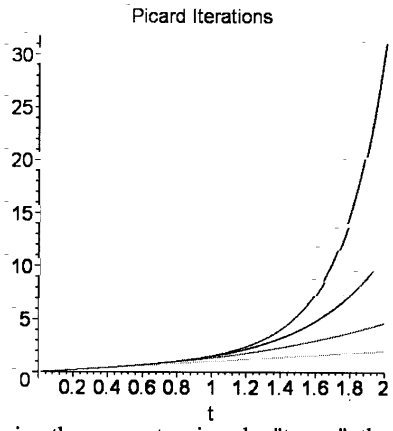
$$0$$

$$t$$

$$\frac{1}{3}t^3 + t$$

$$\frac{1}{63}t^7 + \frac{2}{15}t^5 + \frac{1}{3}t^3 + t$$

$$\frac{1}{59535}t^{15} + \frac{4}{12285}t^{13} + \frac{134}{51975}t^{11} + \frac{38}{2835}t^9 + \frac{17}{315}t^7 + \frac{2}{15}t^5 + \frac{1}{3}t^3 + t$$



Students can experiment by changing the parameters ir order "to see" the convergence of the sequence of functions

But in the following example, the procedure fails, because Maple is not able to integrate the functions involved in the procedure:

```
> f:=(x,y)->-(x-y+3)/(3*x+y+1);
```

$$f := (x, y) \to -\frac{x - y + 3}{3x + y + 1}$$

```
> Picard(f,0,1,4,2);
```

$$1$$

$$-\frac{t}{3} - \frac{4}{9}\ln(3t + 2) + 1 + \frac{4}{9}\ln(2)$$

$$\int_0^{\infty} \frac{\frac{4t}{3} + \frac{4}{9}\ln(3t + 2) + 2 - \frac{4}{9}\ln(2)}{\frac{8t}{3} - \frac{4}{9}\ln(3t + 2) + 2 + \frac{4}{9}\ln(2)}\, dt$$

Maple can't continue

In this case, it is not possible to do a graph for the convergent sequence.

The teacher solves the problem. Discrete versus continuous.

It is possible to solve this problem by doing a discrete version of the procedure.

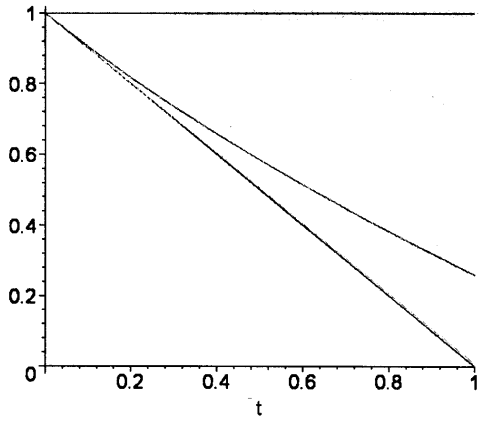
Functions in the Picard's sequence can be discretized and the integrals calculated by the trapezoidal rule.

The following procedure is too technical for students. But they can use it without seeing the body

of the procedure.

```
> PicarDis:=proc(f::procedure,t0::constant,y0::constant,nit::posi
nt,P::posint,h::positive)
  local i,j,n,m,t,tAux,sequence;
   t:=array(1..P+1,1..2);
   tAux:=array(1..P+1,1..2);
   t[1,1]:=t0; t[1,2]:=y0;
   tAux[1,1]:=t0; tAux[1,2]:=y0;
   for n from 2 to P+1 do
      tAux[n,1]:=tAux[n-1,1]+h;
      t[n,1]:=t[n-1,1]+h; tAux[n,2]:=y0;
   od;
   sequence[0]:=convert(tAux,listlist);
   for j to nit do
   for n from 2 to P+1 do
        t[n,2]:=evalf(y0+h/2*(f(tAux[1,1],tAux[1,2])+
f(tAux[n,1],tAux[n,2])+
2*sum(f(tAux[i,1],tAux[i,2]),i=2..n-1)));
   od;
    sequence[j]:=convert(t,listlist);
    tAux:=t;
  od;
  print(plot({seq(sequence[i],i=0..nit)},t=t0..t0+h*P)):
   eval(t);
end:
```

```
> PicarDis(f,0,1,7,10,0.1);
```



$$\begin{bmatrix} 0 & 1 \\ 0.1 & 0.9000000000 \\ 0.2 & 0.8000000001 \\ 0.3 & 0.7000000002 \\ 0.4 & 0.6000000002 \\ 0.5 & 0.5000000002 \\ 0.6 & 0.4000000000 \\ 0.7 & 0.3000000000 \\ 0.8 & 0.2000000005 \\ 0.9 & 0.1000000000 \\ 1.0 & 0.5\ 10^{-9} \end{bmatrix}$$

```
>
```

Maple Worksheet ODE3.MWS

EXAMPLE 3: We give a few exact methods for solving ODE

1. The CAS can be used to make clear the algorithms

Separable equations: $y' - t^2 y = 0$,

```
> ode:=diff(y(t),t)-t^2*y(t);
```

$$ode := \left(\frac{d}{dt}y(t)\right) - t^2 y(t)$$

```
> dsolve(ode, [separable],useInt);
```

$$\int t^2\,dt - \int^{y(t)} \frac{1}{\_a}\,d\_a + \_C1 = 0$$

```
> value(%);
```

$$\frac{t^3}{3} - \ln(y(t)) + \_C1 = 0$$

First order linear equations:

```
> dsolve(ode, [linear],useInt);
```

$$y(t) = \_C1\, e^{\left(\int t^2\,dt\right)}$$

```
> value(%);
```

$$y(t) = \_C1\, e^{\left(\frac{t^3}{3}\right)}$$

2. Easy algorithms can be programmed by students

A **Maple** procedure (made by the students) to verify if $P(t,y)\,dt + Q(t,y)\,dy = 0$ is an exact equation and find its solution:

$$\int P(t,y)\,dt + \int Q(t,y) - \left(\frac{\partial}{\partial y}\left(\int P(t,y)\,dt\right)\right)dy + k = 0.$$

```
> Exact:= proc(P,Q)
  if simplify(diff(P,y)-diff(Q,t))=0 then
    print(`The equation is exact. The general solution is:`);
    simplify(int(P,t)+int(Q-diff(int(P,t),y),y))+k=0;
  else
    print(`The equation`);
  print(diff(y(t),t)=-P/Q);
  print(`is not exact`);
  fi
  end:
```

Example: $((\sin(t\,y) + t\,y\cos(t\,y))\,dt + t^2\cos(t\,y))\,dy = 0.$

```
> Exact(sin(t*y)+t*y*cos(t*y), t^2*cos(t*y));
```

The equation is exact. The general solution is:

$$\sin(t\,y)\,t + k = 0$$

```
>
```

EXAMPLE 4

•Mass-spring systems

We analyse the mass-spring systems for different values of spring constant k, with k>0, and the damping constant n, n>0. Then we solve the equation mx"[t]+ nx'[t]+kx[t]=f[t]. We study the unforced equation (f(t)= 0), and forced equation with f(t)= F*Sin(w t), periodical force and we only analyse this case. We try to analyse different cases according different values of parameters m, n and k. Then we define the equation with input command.
First we define the parameters of equation m x"[t]+ n x'[t]+k x[t]= F Sin[w t]

```
Clear[m, n, k]
m = Input[m]; n = Input[n]; k = Input[k]; Print[{m, n, k}]
```

```
{1, 0, 4}
```

We give now the frequency of forced term and we advise of the possible resonance.

```
Clear[w]
w = Input[w]; Print[w]; If[n == 0, Print["Possible resonance"]]
```

```
2
```

```
Possible resonance
```

If the above answer is possible resonance we try to see if the phenomenon appears.

$$\text{If}\left[w == \sqrt{\frac{k}{m}}\,,\ \text{Print}["\text{RESONANCE}"],\ \text{Print}["\text{NO RESONANCE}"]\right]$$

```
RESONANCE
```

We define the equation. Previously we define the amplitude of the wave.

```
Clear[f]
f = Input[f]; Print[f];
```

```
5
```

```
ecua = m x''[t] + n x'[t] + k x[t] == f * Sin[w t]
```

```
4 x[t] + x''[t] == 5 Sin[2 t]
```

UNFORCED MOVEMENT

We observe the nature of the roots of the characteristic polynomial to analyse the natural response or free response of the system.
We show the different possibilities.

Supercritical damping

```
If[n^2 - 4 k m > 0, Print["Supercritical damping"], Print[" Another case"]]
```

```
 Another case
```

If the answer is Supercritical damping we find the roots and the general solution of the homogeneous equation in exponential form.

```
caract = Solve[m a^2 + n a + k == 0, a]
```

```
solgenh[t_, c1_, c2_] = c1 Exp[caract[[1, 1, 2]] t] + c2 Exp[caract[[2, 1, 2]] t]
```

We jump to the Forced damping.

Sub critical damping

```
If[n^2 - 4 k m < 0, Print["Sub critical damping"], Print[" Another case"]]
```

```
Sub critical damping
```

If the answer is Sub critical damping we use Root command to find the roots. After we obtain the general solution of the homogeneous equation.

```
caract = Roots[m a^2 + n a + k == 0, a]
```

```
a == 2 i || a == -2 i
```

In easier form

```
prim = caract[[1, 2]]
```

```
2 i
```

```
seg = caract[[2, 2]]
```

```
-2 i
```

Analysing again the resonance. ONLY will be possible with imaginary roots of the characteristic polynomial.

```
If[Re[prim] == 0, Print["POSIBLE RESONANCE"], Print["NO RESONANCE"]]
```

```
POSIBLE RESONANCE
```

Analysing the real situation.

```
If[Im[prim] == w, Print["RESONANCE DANGEREOUS CASE"], Print["NO RESONANCE"]]
```

```
RESONANCE DANGEREOUS CASE
```

The general solution.

```
solgenh[t_, c1_, c2_] = c1 Exp[Re[prim] t] Cos[Im[prim] t] +
  c2 Exp[Re[prim] t] Sin[Im[prim] t]
```

```
c1 Cos[2 t] + c2 Sin[2 t]
```

We jump to the Forced damping.

•Critical damping

```
If[n^2 - 4 k m == 0, Print["Critical damping"], Print[" Another case"]]
```

```
 Another case
```

If the answer is Critical damping we use Root command to find the roots. We will obtain the general solution of the homogeneous equation.

```
caract = Roots[m a^2 + n a + k == 0, a]
```

```
prim = caract[[1, 2]]
```

```
seg = caract[[2, 2]]
```

```
solgenh[t_, c1_, c2_] = c1 Exp[prim t] + c2 t Exp[prim t]
```

•FORCED DAMPING

We found a particular solution of non-homogeneous equation, forced response. We suppose null initial conditions (speed and displacement 0 at t=0). We apply the linearity principle to find the general solutions of the non homogeneous equation.
Finally we plot graphs of some solutions for different values of constants c1 y c2.

•Supercritical damping

```
If[n^2 - 4 k m > 0, Print["Supercritical damping"], Print[" Another case"]]
```

```
 Another case
```

If the answer is Supercritical damping we solve the equation with null initial conditions.

```
solecua = DSolve[{ecua, x[0] == 0, x'[0] == 0}, x[t], t]
```

Easier form.

```
solpar[t_] = Simplify[x[t] /. solecua[[1]]]
```

The general solution of equation

```
solfin[t_, c1_, c2_] = solgenh[t, c1, c2] + solpar[t]
```

We plot graphs of some solutions. In this case the solutions decay exponentially for every values of c1 and c2.

```
grafic = Table[solfin[t, c1, c2], {c1, -3, 3, 3}, {c2, -3, 3, 3}]
```

We must be careful with the values of the constants c1 and c2 and the chosen interval for the independent variable. A bad election implies a bad understanding of the graph. Please check each case with several values of constants and several intervals.

```
Plot[Evaluate[grafic], {t, 0, 3}]
```

Remark: Experiment with big values of t, time. Some times only one graph is appearing. Analyse the reason. All transient solutions, corresponding to natural response tend to 0. Only the steady-state response appears.

•Sub critical damping

```
If[n^2 - 4 k m < 0, Print["Sub critical damping"], Print[" Another case"]]
```

```
Sub critical damping
```

If the answer is Sub critical damping we solve the equation with null initial conditions.

```
solecua = DSolve[{ecua, x[0] == 0, x'[0] == 0}, x[t], t]
```

$$\left\{\left\{x[t] \to -\frac{5}{16}\,(4\,t\,\text{Cos}[2\,t] - \text{Sin}[2\,t] + \text{Cos}[4\,t]\,\text{Sin}[2\,t] - \text{Cos}[2\,t]\,\text{Sin}[4\,t])\right\}\right\}$$

Easier form.

```
solpar[t_] = Simplify[x[t] /. solecua[[1]]]
```

$$-\frac{5}{4}\,(t\,\text{Cos}[2\,t]-\text{Cos}[t]\,\text{Sin}[t])$$

The general solution of equation

```
solfin[t_, c1_, c2_] = solgenh[t, c1, c2] + solpar[t]
```

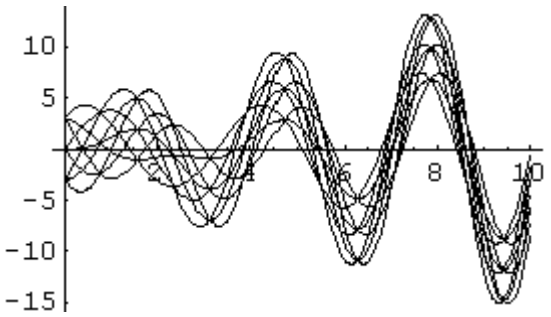
$$c1\,\text{Cos}[2\,t]-\frac{5}{4}\,(t\,\text{Cos}[2\,t]-\text{Cos}[t]\,\text{Sin}[t])+c2\,\text{Sin}[2\,t]$$

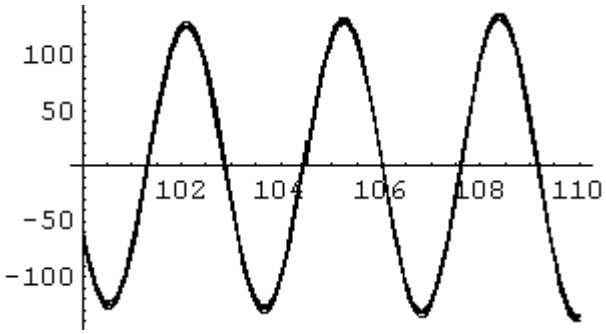
We plot graphs of some solutions. In this case for f(t)= F sin(wt) the solutions decay exponentially, for every values of c1 and c2 but with change of sign, due to the sine and cosine functions. In the resonance phenomenon the solutions are going to infinity, due to the solution tsin(wt) or t cos(wt) or both.

```
grafic = Table[solfin[t, c1, c2], {c1, -3, 3, 3}, {c2, -3, 3, 3}]
```

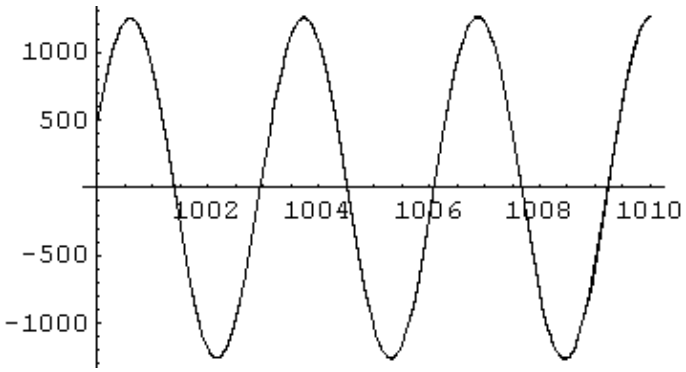
We have to be careful with the values of constants c1 and c2 and the chosen interval for independent variable.
A bad election implies a bad understanding of the graph. Please check in all the cases with several values of constants and several intervals, for small and big values of time.

```
Plot[Evaluate[grafic], {t, 0, 10}]; Plot[Evaluate[grafic], {t, 100, 110}];
```





```
Plot[Evaluate[grafic], {t, 1000, 1010}];
```



Remark: Experiment with small and big values of t, time. Some times only one graph is appearing. Analyse the reason. (Solution of the homogeneous equation are periodic) and for some values of the constant only steady-state response are represented.

•Critical damping

```
If[n^2 - 4 k m == 0, Print["Critical damping"], Print[" Another case"]]
```

```
 Another case
```

If the answer is Critical damping we solve the equation with null initial conditions.

```
solecua = DSolve[{ecua, x[0] == 0, x'[0] == 0}, x[t], t]
```

Easier form

```
solpar[t_] = Simplify[x[t] /. solecua[[1]]]
```

General solution:

```
solfin[t_, c1_, c2_] = solgenh[t, c1, c2] + solpar[t]
```

We graph some solutions. In this case for f[t] = Fsin[wt] the solutions decay exponentially for every values of c1 and c2 but with change of sign, due to sine and cosine functions. in the resonance phenomenon the solutions are going to infinity, due to solution tsin[wt]. Also have to be careful with the values of constants c1 and c2 and the chosen interval for independent variable.
A bad election implies a bad understanding of the graph. Please check in all the cases with several values of constants and several intervals, for small and big values of time.

```
grafic = Table[solfin[t, c1, c2], {c1, -3, 3, 3}, {c2, -3, 3, 3}]
```

```
Plot[Evaluate[grafic], {t, 0, 3}]
```

Remark: Experiment with small and big values of time. Some times only one graph is appearing. Analyse the reason. (Solution of homogeneous equation are periodic) and for some values of the constant only steady-state response is represented.

Converted by *Mathematica* (August 30, 2004)

EXAMPLE 5

•Conservative systems

For special conservative systems like the following x'= y, y´= f(x) we can draw the orbits or trajectories only using the total energy $E = \frac{y^2}{2} - \int_0^x f[x]dx$. These systems can represent different technical and physical situations and we describe some examples.

A spring, with non elastic component

We assume that the associated ODE is $x'' + kx + ax^3 = 0$, which it is equivalent to $x'' = -kx - ax^3$. (The term ax^3 is the non linear component). The behaviour is completely different in the cases: a <0, heavy spring. and a>0, soft spring .
We analyse both of them. We need the package:

Eingabe:
```
<< Graphics`ImplicitPlot`
```

a<0. Heavy spring.
We suppose m=1, being m the mass of the spring. We draw several trajectories.

We assume the ODE associated is $x'' + kx + ax^3 = 0$, with a < 0.

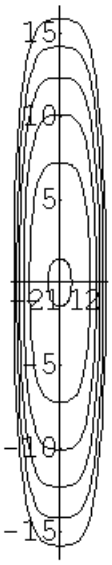
Suppose f(x) = $-2x - 8x^3$

Eingabe:
```
Clear[x, t, f]
```

Eingabe:
```
f[x_] := -2 x - 8 x^3
```

We plot the graphs for different trajectories, using the Table command.

Eingabe:
```
ImplicitPlot[
  Table[y^2 / 2 - Integrate[f[x], {x, 0, x}] == c,
    {c, 1, 126, 25}], {x, -3, 3}];
```



Case a>0. Soft spring.

We suppose the external force is f(x) = -4x+4x^3

Eingabe:

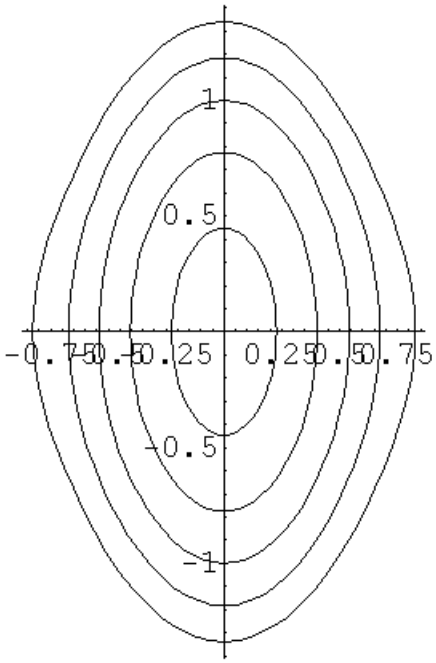
$$f[x\_] := -4x + 4x^3$$

We plot the graphs for some integral curves, corresponding to different values of energy E.

Eingabe:

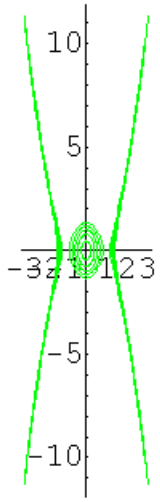
s1 =

$$\text{ImplicitPlot}\left[\text{Table}\left[\frac{y^2}{2} - \int_0^x f[x]\,dx == c,\ \{c, .1, .9, .2\}\right], \{x, -1, 1\}\right];$$



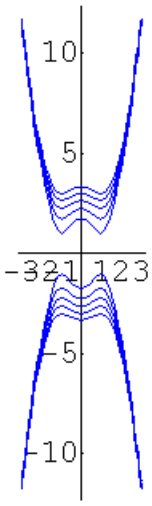
Eingabe:

```
s1 = ImplicitPlot[
   Table[y^2 / 2 - ∫_0^x f[x] dx == c,
    {c, .1, .9, .2}], {x, -3, 3},
   PlotStyle → {RGBColor[0, 1, 0]}];
```



Eingabe:

```
s3 = ImplicitPlot[
   Table[y^2 / 2 - ∫_0^x f[x] dx == c,
    {c, 1.5, 5.5, 1}], {x, -3, 3},
   PlotStyle → {RGBColor[0, 0, 1]}];
```



For different values of the energy E the curves have different shape.
There is a special value of energy, corresponding to energy curves passing through critical unstable points. The critical points are the solutions of f(x)=0, and y=0.

Eingabe:

```
Solve[f[x] == 0, x]
```

Ausgabe:

```
{{x → -1}, {x → 0}, {x → 1}}
```

The unstable points are (-1,0) and (1,0). The energy for this points is:

Eingabe:

$$\left(y\text{^}2/2 - \int_0^x f[x]\,dx\right) /. \{x \text{->} 1, y \to 0\}$$

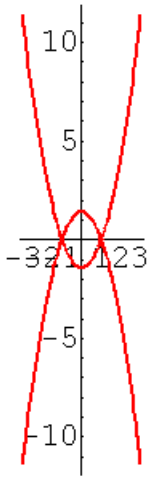
Ausgabe:

```
1
```

We draw the trajectory corresponding to E=1, which it is called the separatrice.

Eingabe:

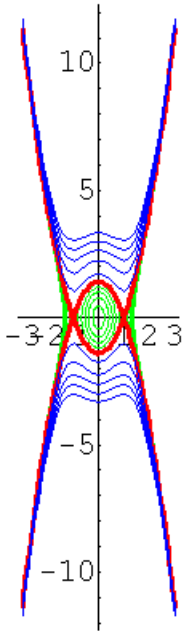
$$s2 = \text{ImplicitPlot}\left[y\text{^}2/2 - \int_0^x f[x]\,dx == 1, \{x, -3, 3\}, \text{PlotStyle} \to \{\text{RGBColor}[1, 0, 0], \text{Thickness}[0.03]\}\right];$$



Now together

Eingabe:

```
Show[s1, s2, s3];
```



The separatrice, red energy, separates the trajectories corresponding to less values of energy, green curves and the curves with more values of energy, blue curves.

Converted by *Mathematica* (August 30, 2004)

Maple Worksheet ODE6.MWS

EXAMPLE 6: Numerical methods

It is easy to do programs for numerical algorithms concerning the approximate solutions of Cauchy problems with Maple and Mathematica. Futhermore symbolic computation allows us to take advantages.

Taylor methods

The Taylor method of orden k for solving the initial value problem $y' = f(t, y)$, with $y(t0) = y0$, can be represented by:

$$y_{n+1} = y_n + h\, f(t_n, y_n) + \frac{h^2}{2!} f'(t_n, y_n) + \frac{h^3}{3!} f''(t_n, y_n) + \ldots + \frac{h^{(k-1)}}{(k-1)!} f^{(k-1)}(t_n, y_n),$$

f', f'',... are successive derivatives of $f(t, y(t))$.

Classical implementation

The second order Taylor method for the problem

$$y' = \frac{t-y}{t+2y},\ y(0) = 1$$

use the algorithm:

$$y_{n+1} = y_n + h\, f(t_n, y_n) + \frac{h^2}{2!} f'(t_n, y_n).$$

Then we need to compute the derivative of $f(t, y(t)) = \frac{t - y(t)}{t - 2\,y(t)}$

We obtain $\frac{1 - \left(\frac{d}{dt} y(t)\right)}{t + 2\,y(t)} - \frac{(t - y(t))\left(1 + 2\left(\frac{d}{dt} y(t)\right)\right)}{(t + 2\,y(t))^2}$

After, we substitute y'(t) by f(t,y) $\frac{1 - \frac{t-y}{t+2y}}{t+2y} - \frac{(t-y)\left(1 + \frac{2(t-y)}{t+2y}\right)}{(t+2y)^2}$

The second order Taylor formula for this equation is:

$$y_{n+1} = y_n + \frac{h\,(t_n - y_n)}{t_n + 2\,y_n} + \frac{3\,h^2\,(2\,y_n\,t_n + 2\,y_n^2 - t_n^2)}{2\,(t_n + 2\,y_n)^3}.$$

```
> f:=(t,y)->(t-y)/(t+2*y):
```

```
> df:=(t,y)->3*(2*y*t+2*y^2-t^2)/(t+2*y)^3:
> t[0]:=0: y[0]:=1: h:=0.3:
> for n from 0 to 9 do
  t[n+1]:=t[n]+h:
  y[n+1]:=y[n]+h*f(t[n],y[n])+h^2/2*df(t[n],y[n]):
  print('y'(t[n+1])=evalf(y[n+1]))
  od:
```

$$y(0.3) = 0.8837500000$$
$$y(0.6) = 0.8296321763$$
$$y(0.9) = 0.8226954946$$
$$y(1.2) = 0.8483791733$$
$$y(1.5) = 0.8960999623$$
$$y(1.8) = 0.9588642792$$
$$y(2.1) = 1.032127082$$
$$y(2.4) = 1.112901214$$
$$y(2.7) = 1.199182458$$
$$y(3.0) = 1.289594346$$

```
> restart:
```

We have a different algorithm for each order and for each equation.

CAS implementation

We can do a unique program where equation and order of Taylor method are input parameters:

```
> MTaylor:=proc(f::procedure,t0::constant,y0::constant,
  N::posint,h::positive,k::posint)
      local n,_t,_y,pol,H,y,punt;
         _t[1]:=t0;
         for n to N do
           _t[n+1]:=_t[n]+h
          od;
         _y[1]:=y0;
         D(y):=unapply(f(t,y(t)),t);
         pol:=convert(taylor(y(t+H),H,k+1),polynom);
         pol:=unapply(pol,t,y,H);
         for n to N do
           _y[n+1]:=evalf(pol(_t[n],_y[n],h))
         od;
        punt:=[seq([_t[n],_y[n]],n=1..N+1)]:
    array(1..N+1,1..2,punt)
     end proc:
```

We solve, with this procedure, the problem $y' = \frac{t-y}{t+2y}$, $y(0) = 1$:

```
> f:=(t,y)->(t-y)/(t+2*y):
> MTaylor(f,0,1,10,0.3,2);
```

0	1
0.3	0.8837500000
0.6	0.8296321763
0.9	0.8226954946
1.2	0.8483791733
1.5	0.8960999623
1.8	0.9588642792
2.1	1.032127082
2.4	1.112901214
2.7	1.199182458
3.0	1.289594346

```
>
```

Deduction of multi-step formulas.

A multi-step method for solving the initial value problem $y' = f(t, y)$, $y(t0) = y0$, is one whose difference equation for finding the approximation can be represented by

$y_{n+1} = y_n + h\,(a_0\, f(t_n, y_n) + a_1\, f(t_{n-1}, y_{n-1}) + \ldots + a_p\, f(t_{n-p}, y_{n-p}))$.

To begin the deduction of the multi-step methods, note that the solution y(t) of the problem has the property:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt.$$

Since we can not integrate $f(t, y(t))$ without knowing $y(t)$, we instead integrate an interpolating polynomial P, which is determined by some of the previously obtained data points.

The k-step Adams-Bashforth formula is obtained by $y_{n+1} = y_n + \int_{t_n}^{t_n+h} P\, dx$ where P is the interpolating polynomial at $[t_n, y_n]$,....,$[t_{n-k}, y_{n-k}]$.

The classical deduction of these formulas are very tedious, and usually students are lost in the computation. But with a CAS the students can themselves obtain the formulas. The following Maple procedure (made by students) gives the k-step Adams-Basforth formula:

```
> AB:= proc(k::posint)
  local P,t,f,y,n;
  P:=interp([seq(t[n]-j*h,j=0..k-1)],[seq(f[n-j],j=0..k-1)],x):
  y[n+1]=y[n]+simplify(int(P,x=t[n]..t[n]+h))
  end:
> AB(4);
```

$$y_{n+1} = y_n + \frac{1}{24} h\,(55 f_n - 9 f_{n-3} + 37 f_{n-2} - 59 f_{n-1})$$

Obviously the same thing can be done for Adams-Moulton formulas and for other multi-step methods.