```
> restart;
```

Numeric solutions of ODEs in Maple

The purpose of this worksheet is to introduce Maple's **dsolve/numeric** command. There are many examples of differential equations that Maple cannot solve analytically, it these cases a default call to dsolve returns a null (blank) result:

```
> ode := diff(y(x),x,x) + y(x)^2 = x^2;
  dsolve(ode);
```

$$ode := \frac{d^2}{dx^2} y(x) + y(x)^2 = x^2 \quad (1)$$

If this happens, one can obtain a solution numerically by specifying initial conditions and providing the option "numeric":

```
> ICs := y(0) = 0, D(y)(0)= 1/2;
  sol := dsolve({ode,ICs},numeric);
```

$$ICs := y(0) = 0, \mathrm{D}(y)(0) = \frac{1}{2}$$

$$sol := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc} \quad (2)$$

The output of dsolve is by default a Maple procedure of a single argument. If we call this procedure with argument *x*, we obtain information about the solution at that value of the independent variable:

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

$$\left[x = 1., y(x) = 0.560986197489666, \frac{d}{dx} y(x) = 0.739332218315567\right] \quad (3)$$

That is, we get a list giving us a numeric approximation to the value of the unknown and its first derivative at our choice of *x*. If the equation we wanted to solve was higher order (say n^{th} order), we would have more elements in the list corresponding to all the derivatives of *y* up to $(n-1)^{th}$ order. The default behaviour of dsolve/numeric is to return a procedure which itself returns a list; however, we can instead have it return a list of procedures but using the optional command "output = listprocedure".

```
> sol := dsolve({ode,ICs},numeric,output=listprocedure);
```

$$sol := \left[x = \mathbf{proc}(x) \ \ldots \ \mathbf{end\ proc}, y(x) = \mathbf{proc}(x) \ \ldots \ \mathbf{end\ proc}, \frac{d}{dx} y(x) = \mathbf{proc}(x) \quad (4)$$

...

end proc]

Here, we get a list of three equations. The RHS of each equation is a procedure that calculates what is represented on the LHS. For example, the RHS of the second element is a procedure that calculates $y(x)$. We can isolate this particular procedure as follows:

```
> y_sol := sol[2];
  y_sol := rhs(y_sol);
```

$$y\_sol := y(x) = \mathbf{proc}(x) \ \ldots \ \mathbf{end\ proc}$$

$$y\_sol := \mathbf{proc}(x) \ \ldots \ \mathbf{end\ proc} \quad (5)$$

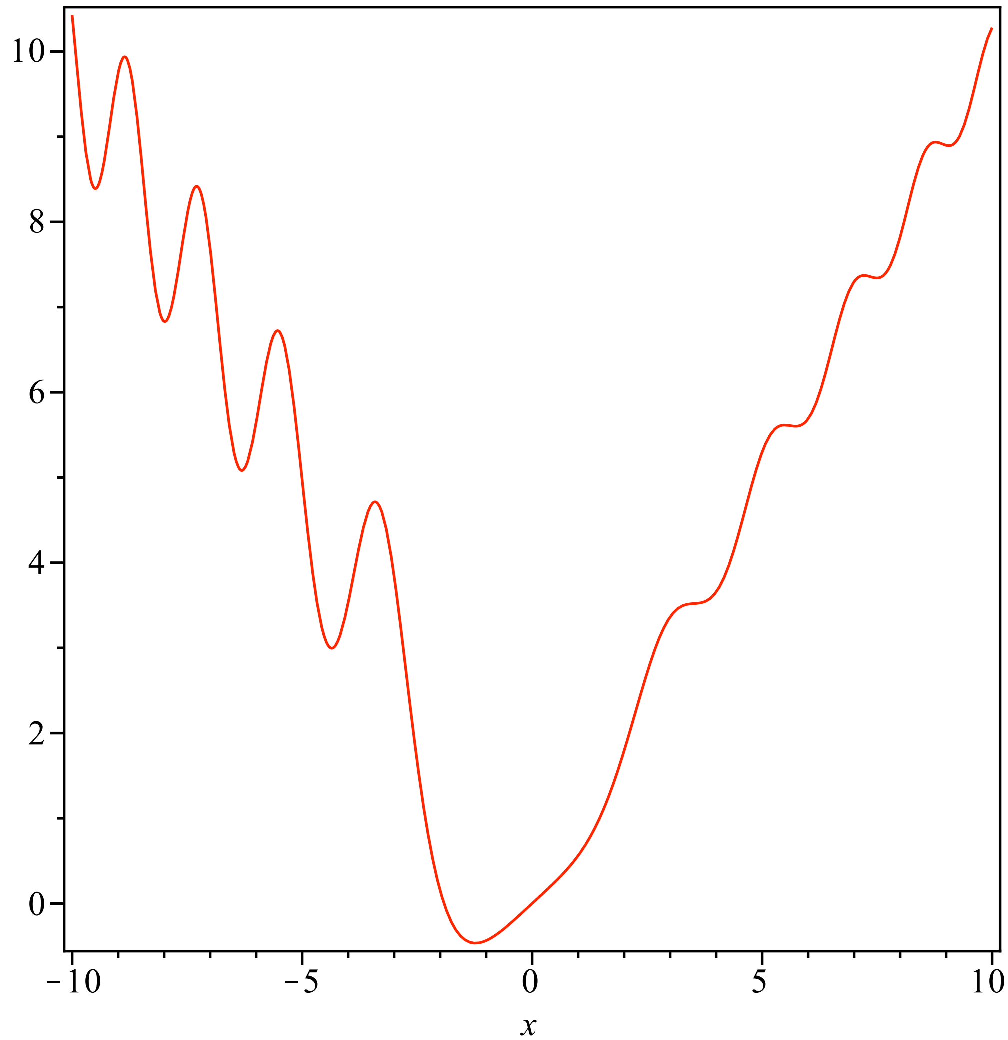
Now, **y_sol** is a procedure that just returns the numeric solution for *y* as a function of *x*:

```
> y_sol(2);
```

$$1.79279104009982 \quad (6)$$

This procedure can readily be used to obtain a plot of the numeric solution of the ODE:

```
> plot(y_sol(x),x=-10..10,axes=boxed);
```



This is not the only way to obtain a plot of the numeric solution of an ODE, look at **`?plots/odeplot`** for an alternate method (I recommend the one given here, howeever). The **`dsolve/numeric`** command can also be used to solve systems of ODEs such as the following [this is a predator-prey model from mathematical biology where $X(T)$ and $Y(T)$ represent the populations of humans and fish, respectively]
:

```
> ODE1 := diff(X(T), T) = -X(T)*(-1+Y(T));
  ODE2 := diff(Y(T), T) = Y(T)*(-1+X(T))*alpha;
```

$$ODE1 := \frac{d}{dT} X(T) = -X(T)\,(-1+Y(T))$$

$$ODE2 := \frac{d}{dT} Y(T) = Y(T)\,(-1+X(T))\,\alpha \qquad (7)$$

Here is the code to generate procedures **`X_sol`** and **`Y_sol`** giving the numeric solution for the two unknowns for a particular choice of initial data and α.

```
> alpha := 1;
  X0 := 2;
  Y0 := 1;
  ans := dsolve([ODE1,ODE2,X(0)=X0,Y(0)=Y0],numeric,output=
  listprocedure);
```

```
X_sol := rhs(ans[2]);
Y_sol := rhs(ans[3]);
```

$$\alpha := 1$$

$$X0 := 2$$

$$Y0 := 1$$

$ans := [T = \mathbf{proc}(T)\ \ldots\ \mathbf{end\ proc}, X(T) = \mathbf{proc}(T)\ \ldots\ \mathbf{end\ proc}, Y(T) = \mathbf{proc}(T)$

...

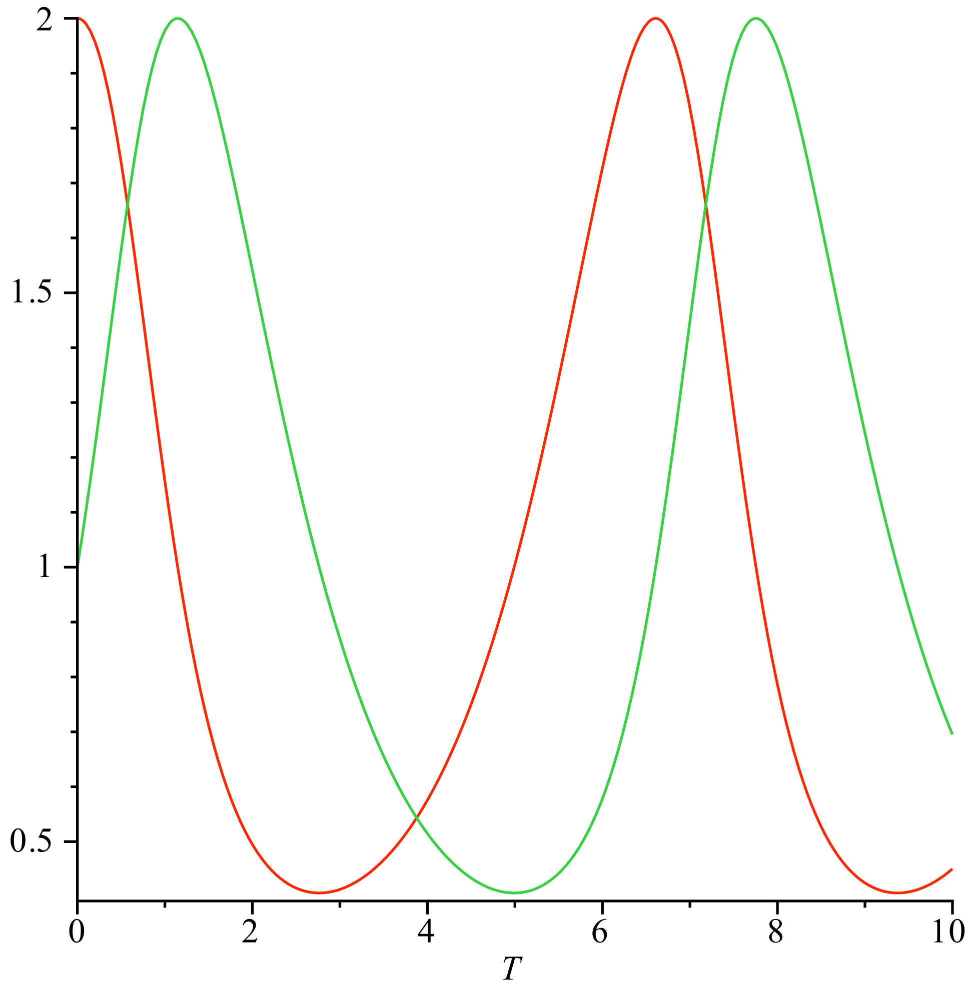
end proc]

$$X\_sol := \mathbf{proc}(T)\ \ldots\ \mathbf{end\ proc}$$

$$Y\_sol := \mathbf{proc}(T)\ \ldots\ \mathbf{end\ proc} \quad \textbf{(8)}$$

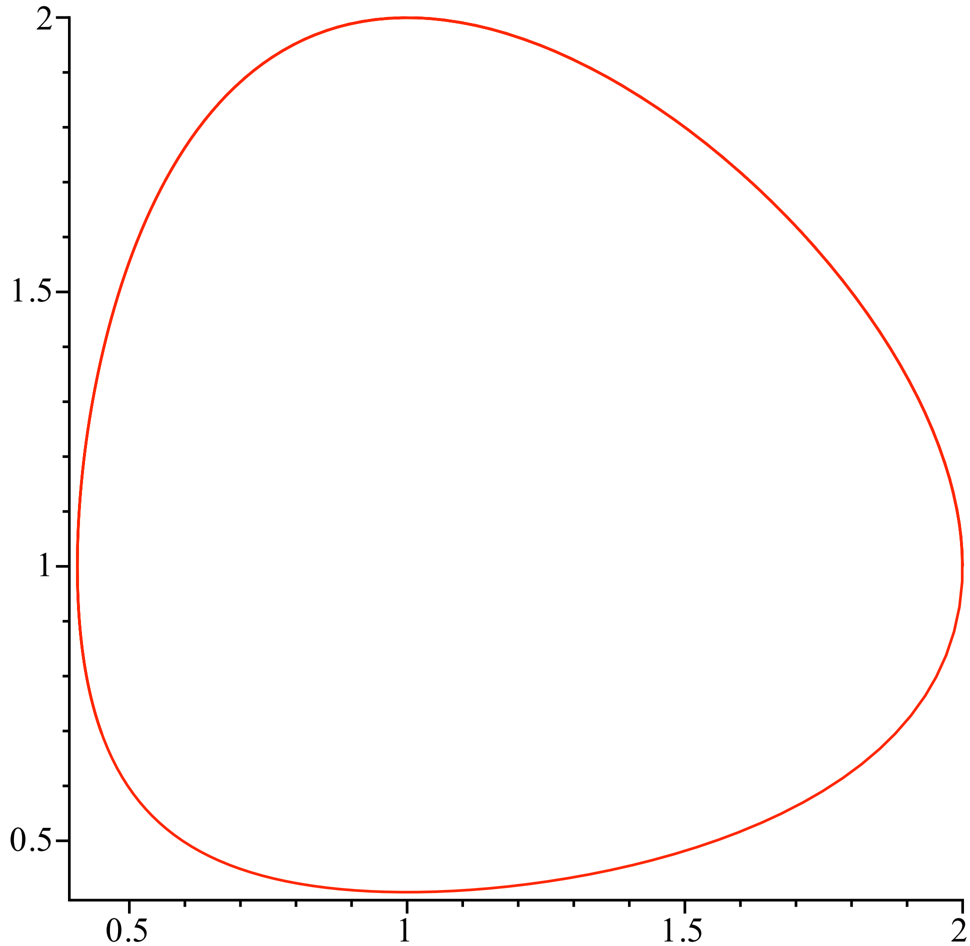
Here is a plot of the two solutions:

```
> plot([X_sol(T),Y_sol(T)],T=0..10);
```



A phase portrait of the system is an implicitly defined curve where the horizontal coordinate is $X(T)$ and the vertical coordinate is $Y(T)$:

```
> plot([X_sol(T),Y_sol(T),T=0..10]);
```



The above numeric solution was for a particular choice of initial data. Lets generate a series of ten solution curves for a number of choices of initial data (N.B. now we choose α = 10):

```
> alpha := 10;
  for i from 1 to 10 do:
     X0 := 1+i/10;
     Y0 := 1;
     ans := dsolve([ODE1,ODE2,X(0)=X0,Y(0)=Y0],numeric,output=
  listprocedure);
     X_sol[i] := rhs(ans[2]);
     Y_sol[i] := rhs(ans[3]);
     print(i);
  od:
```

$$\alpha := 10$$

1

2

3

4

5

6

7

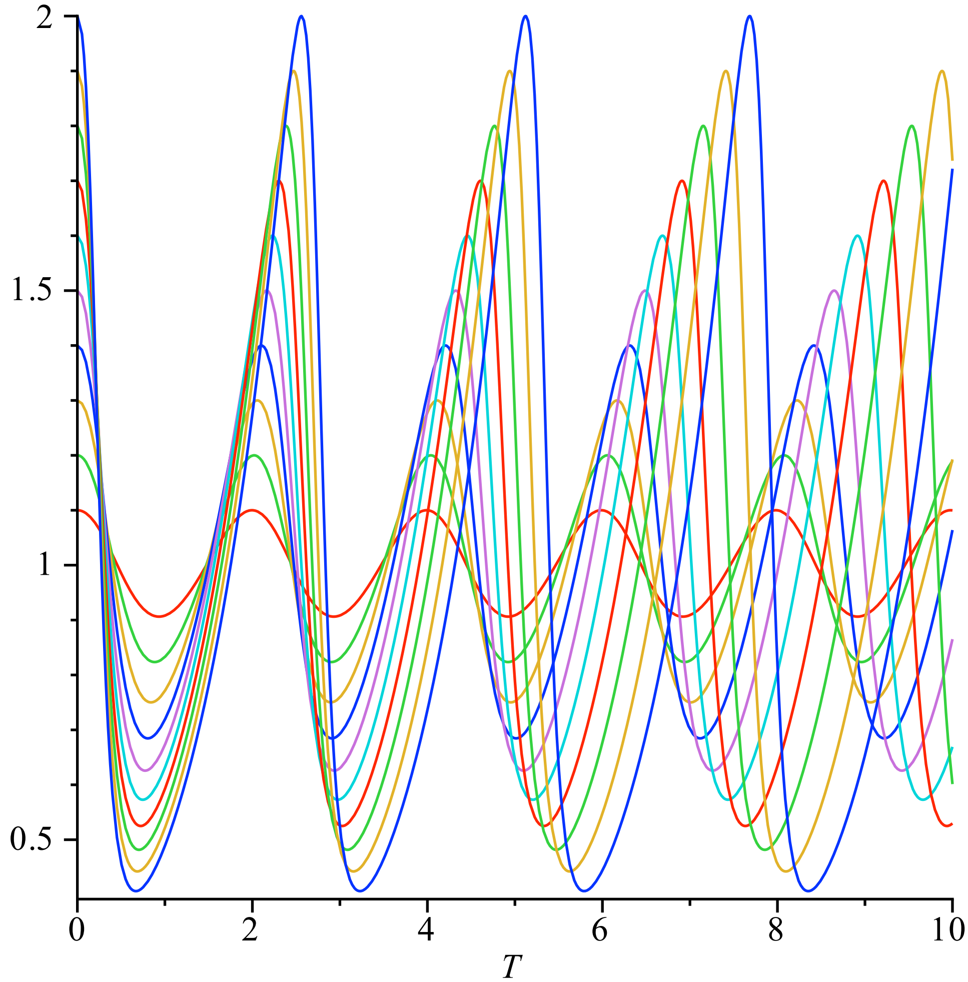
8

9

10 **(9)**

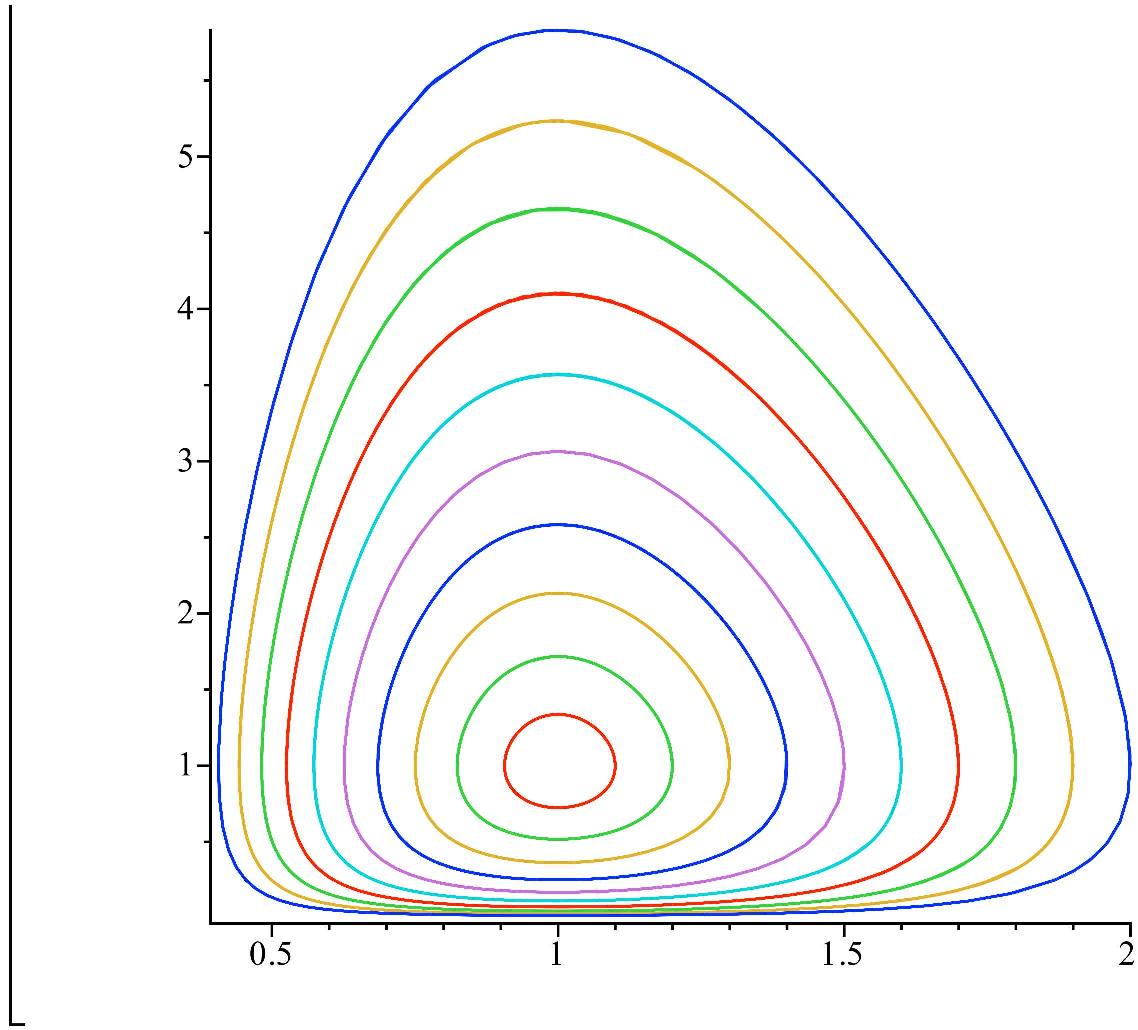
Here is a plot of the X solutions as a function of T:

```
> plot([seq(X_sol[i](T),i=1..10)],T=0..10);
```



Here is a plot of the phase portrait of each of the solutions:

```
> plot([seq([X_sol[i](T),Y_sol[i](T),T=0..10],i=1..10)]);
```


5
4
3
2
1
0.5
1
1.5
2