

MLC Lab Visit - Lab 03 - Maple

Mth 355 (a.k.a. Mth 399) Jan 22, 2003 Maple 7
Bent E. Petersen

There are 4 problems below. Problem solutions are due Jan 29, 2003. Email your solutions to me as Maple worksheet attachments. Your worksheet must execute correctly for full credit.

In this worksheet we investigate a few more features of Maple.

- Introduction

If you are viewing the MWS version of this document you will note all the Maple output has been removed. You will have to execute each command (by pressing Enter when the cursor is on the command line) to see the output. You can of course also use the menu selections **Edit/Execute/Worksheet** to execute the entire worksheet, but you should execute the commands one at a time and take the opportunity to experiment along the way! Change some things to see how things work.

The static PDF version of this document shows all of the Maple output. It is useful to look at when Maple is not available.

[>

- Equations

[> **restart;**

In Maple any expression, including an equation, may be assigned to a label. Thus we can take a simple equation such as $A=B$ and assign it to a label for ease in dealing with it. Thus

[> **eqn1:=A=B;**

eqn1 := A = B

Here eqn1 is the label (or name) of the equation $A=B$. Sometimes we wish to extract the left or right (hand) side of an equation.

[> **lhs(eqn1);**

A

[> **rhs(eqn1);**

B

There are devious, sometimes useful, ways to achieve the same thing. For example,

```
> subs(eqn1, A);
```

B

Here we have taken the expression A and substituted A=B in it. The result is of course B. Note eqn1 is not altered in any way - it simply provides the specification for what to substitute for what. Note also the subs() approach has the advantage that, unlike rhs(), it correctly handles expressions containing several equations.

Another approach is to use the assign() function. It assigns the right side of each equality in a list to the left side, that is the left side becomes a label for the right side. Thus

```
> assign(eqn1);
```

```
> B:=6; A;
```

$B := 6$

6

In case we want to use A and B again let's unassign them:

```
> unassign('A', 'B');
```

The single quotes here prevent Maple from evaluating A and B. Otherwise we would be trying to unassign 6, which would produce an error.

```
>
```

- Functions and Differentiation

```
> restart;
```

A function in Maple is defined by the "arrow" notation:

```
> f:=x->x^2+2*x-3;
```

$f := x \rightarrow x^2 + 2x - 3$

A suitable expression may be converted to a function by using unapply(). Thus

```
> expr:=x^2+2*x-3;
```

$expr := x^2 + 2x - 3$

```
> g:=unapply(expr, x);
```

$$g := x \rightarrow x^2 + 2x - 3$$

Yes there is a related `apply()` function too!

```
> apply(g,x);
```

$$x^2 + 2x - 3$$

It's usually easier just to use `g(x)` to apply `g` to `x`.

Functions are evaluated in the usual familiar way, whereas expressions are evaluated by using the substitute command `subs()`. Thus

```
> f(t); f(6);
```

$$t^2 + 2t - 3$$
$$45$$

```
> subs(x=6,expr);
```

$$45$$

One thing to be careful about is the `x` in the arrow notation above is local to the function we are defining, that is, it is just a dummy variable. Thus

```
> h:=x->expr;
```

$$h := x \rightarrow expr$$

does not define the same function as `f` above, but instead defines the constant function whose value is `expr`,

```
> h(t); h(6);
```

$$x^2 + 2x - 3$$
$$x^2 + 2x - 3$$

It is best not to be too devious when defining functions. Errors like this can be hard to find!

Maple works well with expressions, but is not quite as competent with functions. If one plans to use Maple to simplify complicated expressions, it is best to leave them as expressions, even if habit makes functions feel more natural to work with.

```
> is(f=g);
```

FAIL

```
> is(f(x)=g(x));
```

true

Note FAIL does not mean false. It means Maple does not know the answer.

The differentiation operator for functions is D(). The differentiation operator for expressions is diff(). Thus

```
> D(f);
```

$$x \rightarrow 2x + 2$$

```
> diff(f(x), x);
```

$$2x + 2$$

Note that diff() is very easy to use for partial derivatives and for higher order derivatives. Thus

```
> diff(arctan(y/x), x, y): simplify(%);
```

$$-\frac{x^2 - y^2}{(x^2 + y^2)^2}$$

The D operator on the other hand applies only to functions of one variable.

Note the use of Maple ditto operator % above. Be very careful! This operator refers to the previously evaluated expression (in time), not the previous expression on the worksheet. It makes a difference since one can move around and evaluate expressions in numerous places on the worksheet, so previous in time need not be the same as previous in location. The two are the same of course if you restrict % to refer to an expression on the same line, as above. That is the safest way to use the % operator.

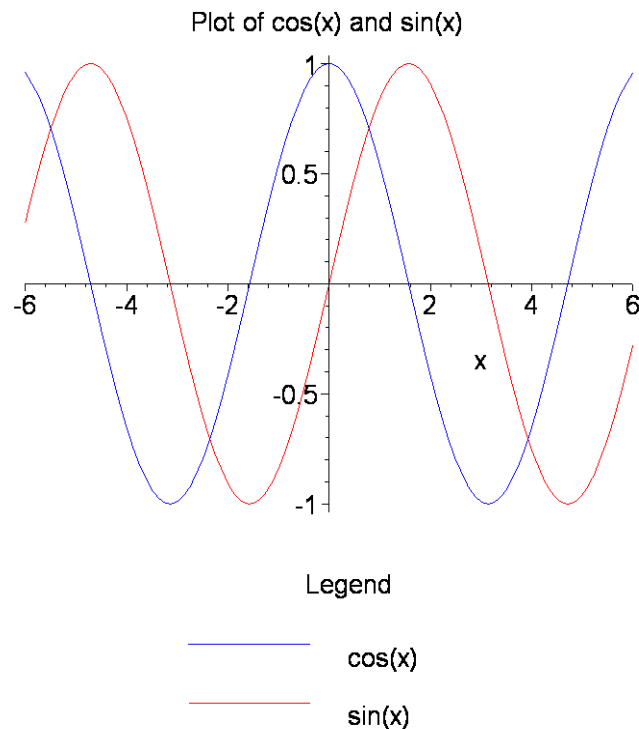
```
>
```

Plotting Multiple Functions

```
> restart;
```

There are several ways to plot multiple functions in one plot, but the simplest is to plot a list of functions. Note we can control the colors and add a title and a legend. Here is a simple example:

```
> plot([cos(x), sin(x)], x=-6..6, color=[blue, red],  
      legend=["cos(x)", "sin(x)"], title="Plot of cos(x) and sin(x)");
```



>

— Ordinary Differential Equations (Symbolic)

> `restart;`

We use the `dsolve()` command to solve differential equations. It returns an expression (which may be empty) which contains equations giving the solution.

There are numerous numerical versions of `dsolve()` which a procedure for approximating a solution. We will discuss this topic in the next section.

Here's an example:

> `ode1:=diff(y(x),x)=x*sec(y(x));`

$$ode1 := \frac{\partial}{\partial x} y(x) = x \sec(y(x))$$

> `init1:=y(0)=2;`

$$init1 := y(0) = 2$$

> `soln1:=dsolve({ode1,init1},y(x));`

$$soln1 := y(x) = \arcsin\left(\frac{1}{2}x^2 + \sin(2)\right)$$

If one wants the solution expressed as a function y, that can be achieved as follows:

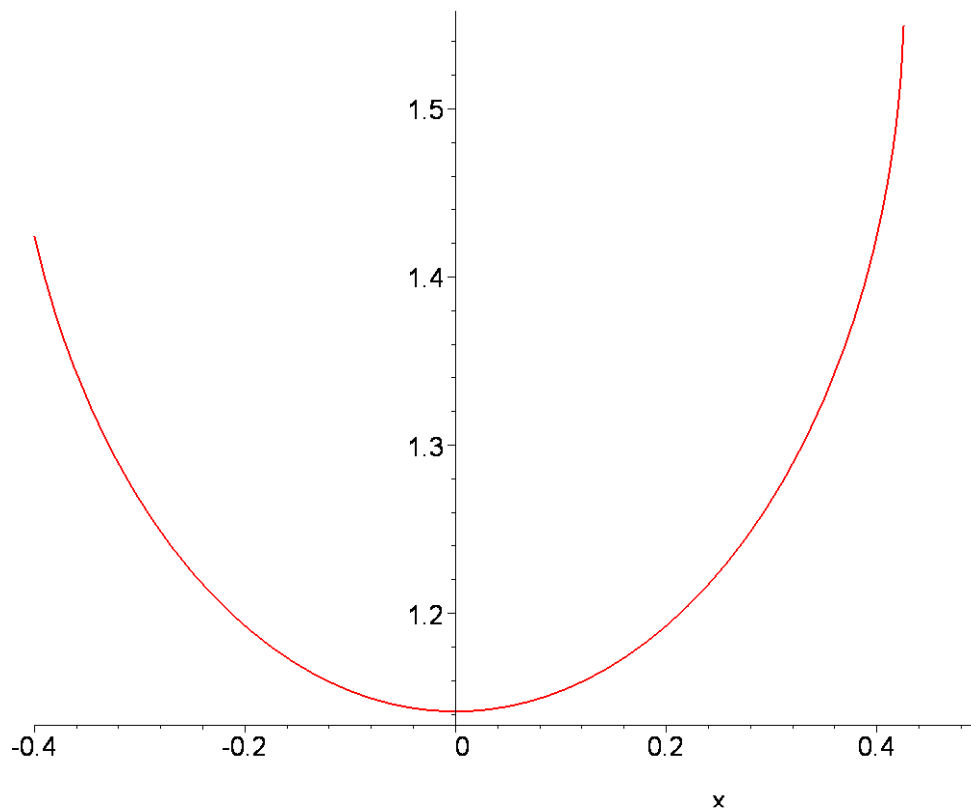
```
> y:=unapply(rhs(soln1),x);
```

$$y := x \rightarrow \arcsin\left(\frac{1}{2}x^2 + \sin(2)\right)$$

The disadvantage here is we have now assigned a value to y. We will not be able to use y again as a "variable" in a differential equation without first unassigning it.

If you just want to plot the solution, or manipulate it in some other way, it is not necessary to convert it to a function - you just extract the appropriate expression and plot it. Thus

```
> plot(rhs(soln1),x=-2/5..1/2,color=red,thickness=2);
```



Let's clean up:

```
> unassign('y');
```

The conversion to a function above can be done more simply by using the assign() function. First we assign y(x) as a label for the solution expression.

```
> assign(soln1);
```

```
> y:=unapply(y(x),x);
```

$$y := x \rightarrow \arcsin\left(\frac{1}{2}x^2 + \sin(2)\right)$$

Let's clean up again:

```
> unassign('y');
```

Here's another example:

```
> ode2:=diff(y(x),x,x)-6*diff(y(x),x)+5*y(x)=exp(5*x)-exp(-5*x);
```

$$ode2 := \left(\frac{\partial^2}{\partial x^2} y(x)\right) - 6 \left(\frac{\partial}{\partial x} y(x)\right) + 5 y(x) = e^{(5.x)} - e^{(-5.x)}$$

```
> soln2:=dsolve(ode2);
```

$$soln2 := y(x) = e^{(5.x)} _C2 + e^x _C1 + \frac{1}{240} e^{(-5.x)} (60x - 15) e^{(10.x)} - \frac{1}{60} e^{(-5.x)}$$

Notice the arbitrary constants here. You may prefer a more familiar expression:

```
> subs(_C1=c[1],_C2=c[2],soln2);
```

$$y(x) = e^{(5.x)} c_2 + e^x c_1 + \frac{1}{240} e^{(-5.x)} (60x - 15) e^{(10.x)} - \frac{1}{60} e^{(-5.x)}$$

You can of course also solve an initial value problem:

```
> init2:=y(0)=3,D(y)(0)=-2;
```

$$init2 := y(0) = 3, D(y)(0) = -2$$

```
> solni2:=dsolve({ode2,init2},y(x));
```

$$solni2 := y(x) = \frac{209}{48} e^x - \frac{51}{40} e^{(5.x)} + \frac{1}{240} e^{(-5.x)} (60x - 15) e^{(10.x)} - \frac{1}{60} e^{(-5.x)}$$

System of differential equations

Let's look at a simple system

```
> ode3a:=diff(y(t),t)=3*x(t)-2*y(t);
```

$$ode3a := \frac{\partial}{\partial t} y(t) = 3 x(t) - 2 y(t)$$

```
> ode3b:=diff(x(t),t)=2*x(t)+3*y(t);
```

$$\text{ode3b} := \frac{\partial}{\partial t} x(t) = 2 x(t) + 3 y(t)$$

> **init3:=x(0)=3,y(0)=-1;**

$$\text{init3} := x(0) = 3, y(0) = -1$$

> **soln3:=dsolve({ode3a,ode3b,init3},{x(t),y(t)});**

$$\text{soln3} := \left\{ x(t) = \frac{1}{3} \left(-\frac{1}{2} + \frac{11}{26} \sqrt{13} \right) \sqrt{13} e^{(\sqrt{13} t)} - \frac{1}{3} \left(-\frac{1}{2} - \frac{11}{26} \sqrt{13} \right) \sqrt{13} e^{(-\sqrt{13} t)} \right. \\ \left. + \frac{2}{3} \left(-\frac{1}{2} + \frac{11}{26} \sqrt{13} \right) e^{(\sqrt{13} t)} + \frac{2}{3} \left(-\frac{1}{2} - \frac{11}{26} \sqrt{13} \right) e^{(-\sqrt{13} t)}, \right. \\ \left. y(t) = \left(-\frac{1}{2} + \frac{11}{26} \sqrt{13} \right) e^{(\sqrt{13} t)} + \left(-\frac{1}{2} - \frac{11}{26} \sqrt{13} \right) e^{(-\sqrt{13} t)} \right\}$$

> **soln3:=simplify(soln3);**

$$\text{soln3} := \left\{ x(t) = \frac{3}{26} \sqrt{13} e^{(\sqrt{13} t)} + \frac{3}{2} e^{(\sqrt{13} t)} - \frac{3}{26} \sqrt{13} e^{(-\sqrt{13} t)} + \frac{3}{2} e^{(-\sqrt{13} t)}, \right. \\ \left. y(t) = -\frac{1}{2} e^{(\sqrt{13} t)} + \frac{11}{26} \sqrt{13} e^{(\sqrt{13} t)} - \frac{1}{2} e^{(-\sqrt{13} t)} - \frac{11}{26} \sqrt{13} e^{(-\sqrt{13} t)} \right\}$$

The easiest way to extract the individual pieces is to use assign(). Also assign() makes no assumptions about the order of the solutions.

> **assign(soln3);**

Note this assigns x(t) and y(t) as labels for the appropriate expressions (not functions!).

> **x(t);**

$$\frac{3}{26} \sqrt{13} e^{(\sqrt{13} t)} + \frac{3}{2} e^{(\sqrt{13} t)} - \frac{3}{26} \sqrt{13} e^{(-\sqrt{13} t)} + \frac{3}{2} e^{(-\sqrt{13} t)}$$

> **y(t);**

$$-\frac{1}{2} e^{(\sqrt{13} t)} + \frac{11}{26} \sqrt{13} e^{(\sqrt{13} t)} - \frac{1}{2} e^{(-\sqrt{13} t)} - \frac{11}{26} \sqrt{13} e^{(-\sqrt{13} t)}$$

It is important to realise that x and y remain unassigned. The labels x(t) and y(t) here are atomic labels, not function evaluations. If you want x and y to be functions corresponding to the solutions you have to say so explicitly:

> **x:=unapply(x(t),t); y:=unapply(y(t),t);**

$$x := t \rightarrow \frac{3}{26} \sqrt{13} e^{(\sqrt{13} t)} + \frac{3}{2} e^{(\sqrt{13} t)} - \frac{3}{26} \sqrt{13} e^{(-\sqrt{13} t)} + \frac{3}{2} e^{(-\sqrt{13} t)}$$

$$y := t \rightarrow -\frac{1}{2} e^{(\sqrt{13} t)} + \frac{11}{26} \sqrt{13} e^{(\sqrt{13} t)} - \frac{1}{2} e^{(-\sqrt{13} t)} - \frac{11}{26} \sqrt{13} e^{(-\sqrt{13} t)}$$

Let's clean up before going on to another example:

```
> unassign('x', 'y');
```

We do not need to unassign $x(t)$ and $y(t)$. Those labels were clobbered when we defined the functions x and y , since, after defining the functions, $x(t)$ became the function x evaluated at t and $y(t)$ became the function y evaluated at t .

Symbolic Initial Values

As you might expect we can solve (some) initial value problems with symbolic initial values:

```
> ode4:=diff(y(x), x$2)+4*y(x)=sec(x);
```

$$ode4 := \left(\frac{\partial^2}{\partial x^2} y(x) \right) + 4 y(x) = \sec(x)$$

Note we used $x\$2$ rather than x,x to indicate the second derivative. It's not a big deal here, but for higher order derivatives the $x\$n$ notation has obvious advantages.

```
> init4:=y(0)=A,D(y)(0)=B;
```

$$init4 := y(0) = A, D(y)(0) = B$$

```
> soln4:=dsolve({ode4,init4},y(x));
```

$$soln4 := y(x) = \frac{1}{2} \sin(2 x) B + \cos(2 x) (-1 + A) + \sin(2 x) \sin(x)$$

$$-\frac{1}{2} \sin(2 x) \ln\left(\frac{1 + \sin(x)}{\cos(x)}\right) + \cos(x) \cos(2 x)$$

```
>
```

[-] Ordinary Differential Equations (Numeric)

```
> restart;
```

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

Maple has a number of numeric method built in that can be used to solve differential equations that Maple can not solve symbolically.

```
> ode1:=diff(y(x),x)=x^2+x*y(x)+y(x)*tan(y(x));
```

$$ode1 := \frac{\partial}{\partial x} y(x) = x^2 + x y(x) + y(x) \tan(y(x))$$

```
> init1:=y(0)=1;
```

$$init1 := y(0) = 1$$

```
> soln1:=dsolve({ode1,init1},y(x));
```

soln1 :=

The empty solution indicates Maple could not find a solution.

```
> soln1n:=dsolve({ode1,init1},y(x),numeric);
```

soln1n := proc(rkf45_x) ... end proc

This looks mysterious but is a procedure which will compute the numeric approximations to $y(x)$ for specified x . Thus:

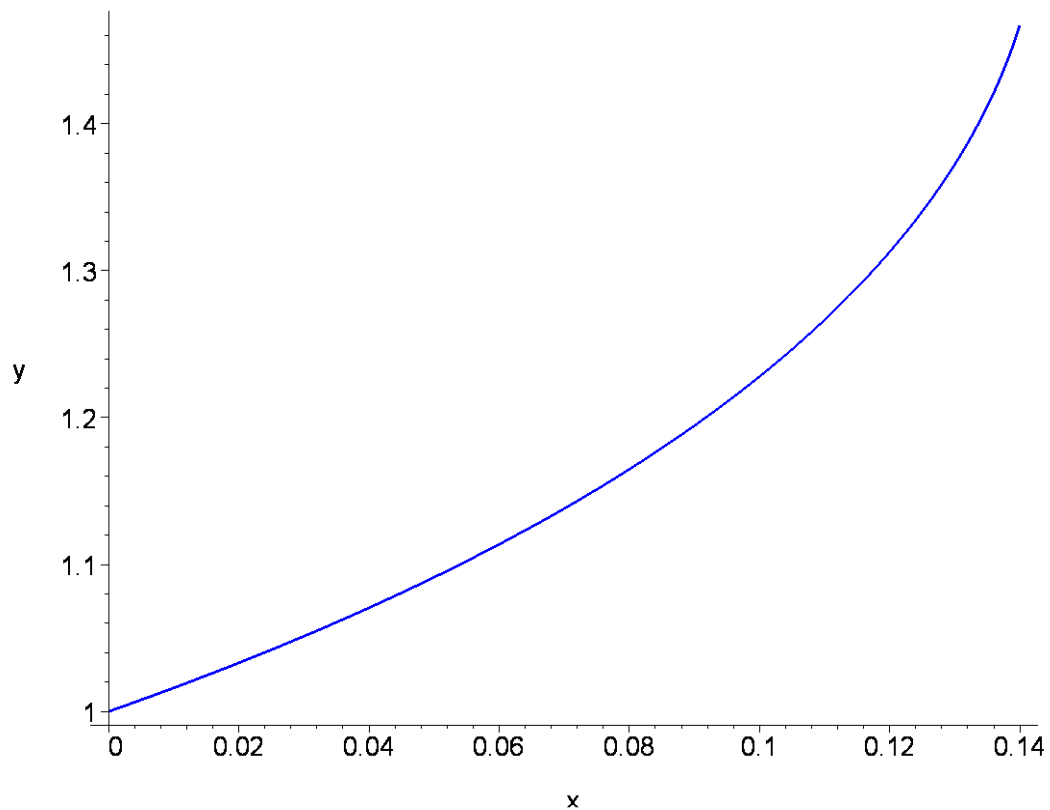
```
> soln1n(0.12); soln1n(0.13);
```

[x = .12, y(x) = 1.31272859522880259]

[x = .13, y(x) = 1.37226447298656406]

Of course we could use these expressions to plot the approximate solution, but fortunately the Maple procedure `odeplot()` understands the data returned by `dsolve/numeric` and is more efficient. Note though that the procedure `odeplot()` is defined in the `plots` package, and so it can be used in the form `odeplot()` only after executing the command `with(plots)` (as we did above). However, it can always be accessed through the long form of the command by using `plots[odeplot]()` (in which case the `plots` package does not have to be loaded explicitly).

```
> odeplot(soln1n,[x,y(x)],0..0.14,thickness=3,color=blue);
```



Note Maple knows quite a few numeric procedures for approximating the solution to an initial value problem. Try the following help request:

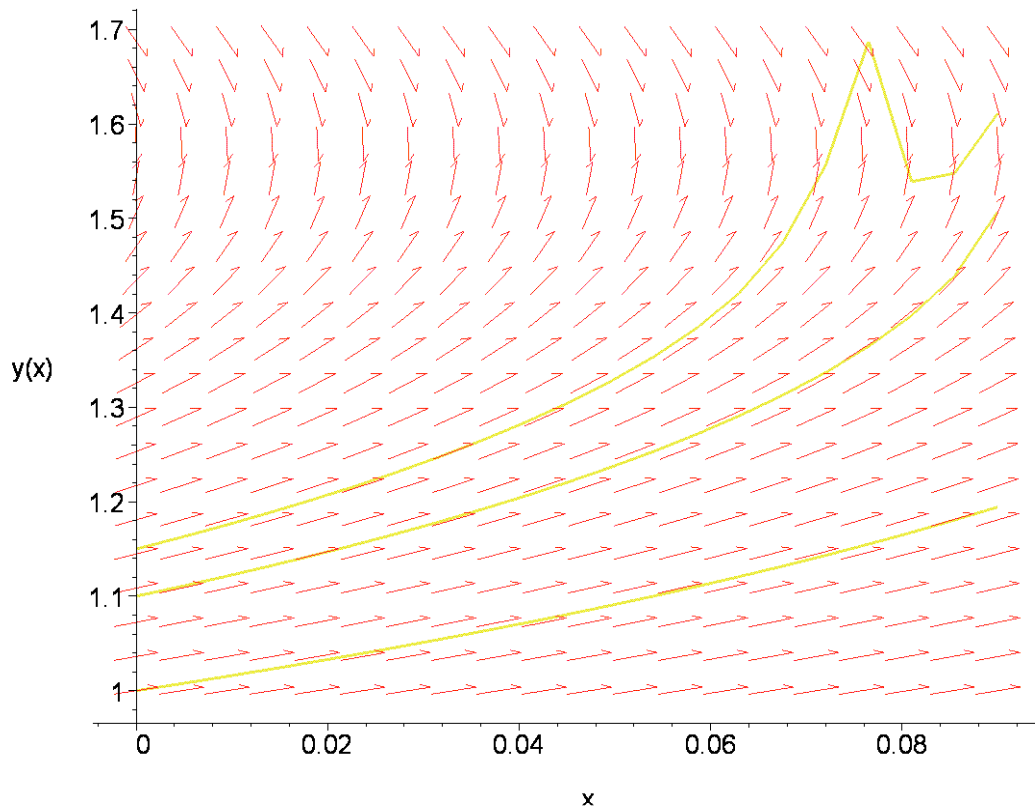
```
> # ?dsolve/numeric
```

to find out what is available.

Note Maple's DEtools package contains many useful tools for studying differential equations. In particular `DEplot()` can be used to plot solution curves (with a direction field in the case of first order differential equations:

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

```
> DEplot(ode1,[y(x)],x=0..0.09,[ [y(0)=1],[y(0)=1.1],[y(0)=1.15] ]);
```



>

- Prime Numbers

```
> restart;
```

The `isprime()` procedure checks if a number is prime:

```
> isprime(104729); isprime(1299709); isprime(3297703);
      true
      true
      false
```

The Maple primality test is very quick, but it is "probabilistic" and could possibly return an incorrect answer. On the other hand no incorrect response has ever been observed.

The `ithprime()` procedure returns the *n*th prime:

```
> ithprime(1); ithprime(2); ithprime(3); ithprime(100);
      2
      3
      5
      541
```

A prime pair is a an ordered pair $[p,p+2]$ such that p and $p+2$ are both prime.

```
> ispprime:=proc(n)
>   isprime(n) and isprime(n+2);
> end;
```

Here are the first few primes p such that $[p,p+2]$ is a prime pair:

```
> ispprime(3); ispprime(5); ispprime(11); ispprime(17);
ispprime(29);
```

true

true

true

true

true

Here is a prime which is the not the first component in a prime pair:

```
> isprime(31); ispprime(31);
```

true

false

It has been conjectured that there is an infinite number of prime pairs.

In the late 1700's and early 1800's Gauss conjectured that the number of primes in an interval of length dx starting at x is about $1/\log(x)$ (natural logarithm). It follows that the number of primes less than or equal to x ought to be about

```
> Int(1/log(t),t=0..x);
```

$$\int_0^x \frac{1}{\ln(t)} dt$$

This integral (which is only interesting for x greater than 2) is divergent and should be taken in the principal value sense. Alternately we can change the lower limit to about 1.4 without changing the value of integral. This makes it convergent since we avoid the singular point at $t = 1$.

This integral is called the logarithmic integral and is implemented in Maple as $\text{Li}()$. Maple can not compute the integral exactly so $\text{Li}()$ will return unevaluated unless the argument is a floating point number.

```
> Li(100); evalf(%); Li(100.0);
```

```
Li(100)  
30.12614158  
30.12614158
```

The lower limit referred to above is found by solving $\text{Li}(x) = 0$

```
> Digits:=40: fsolve(Li(x)=0,x=1..infinity, fulldigits);  
Digits:=10:
```

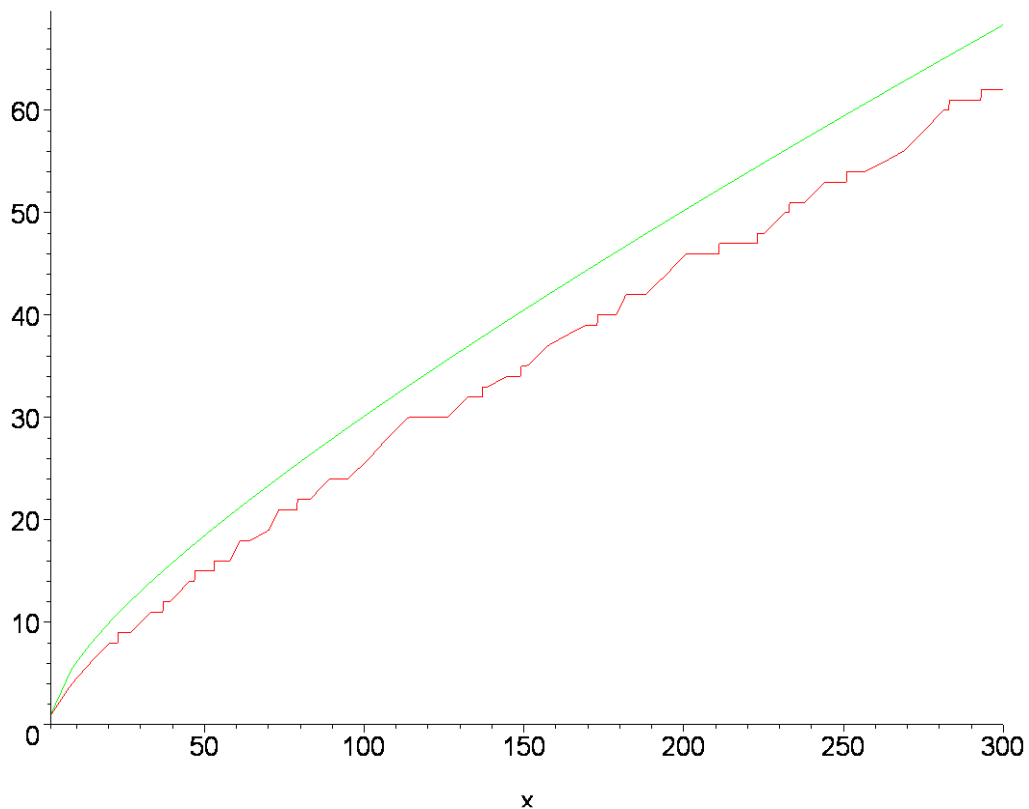
```
1.451369234883381050283968485892027449493
```

Maple is able to compute "exactly" the number of primes less than or equal to x . This function is implemented in the numtheory package, so we can test Gauss's conjecture.

```
> with(numtheory):
```

```
Warning, the protected name order has been redefined and unprotected
```

```
> plot([ pi(floor(x)), Li(x) ], x=2..300 );
```

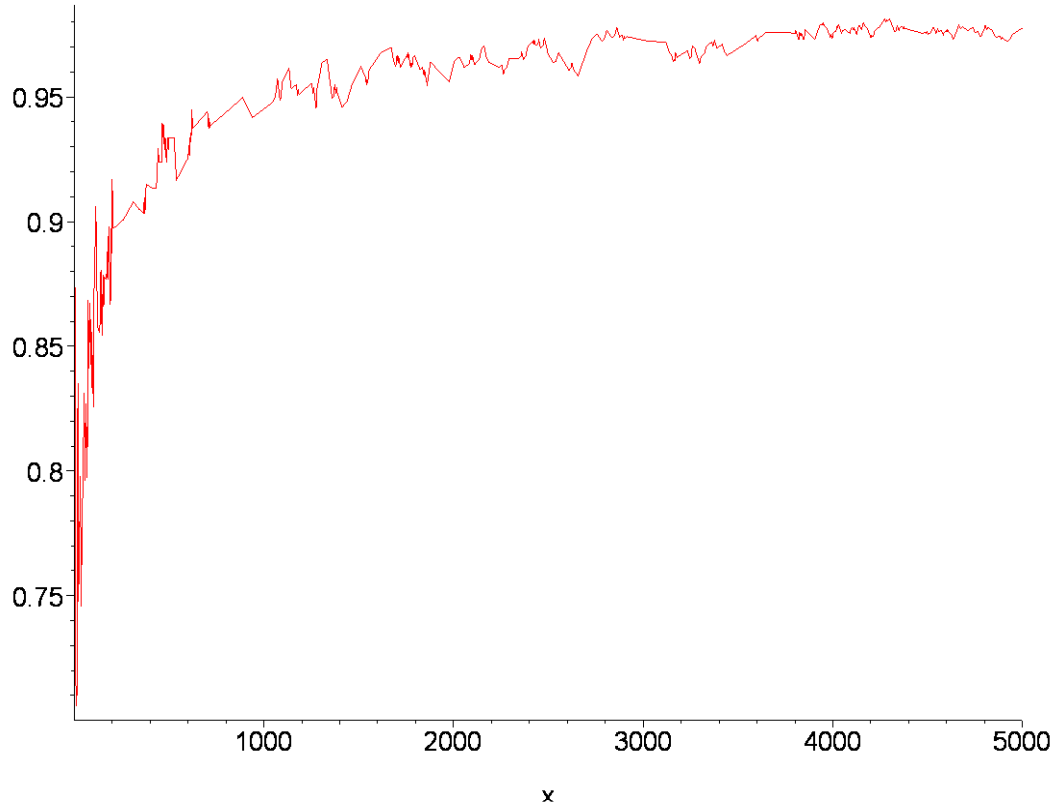


Well, Gauss was certainly onto something!

A stronger result, proved in 1898 by de la Vallée Poussin and Hadamard is the Prime Number

Theorem which says that $\pi(x)/\text{Li}(x)$ has limit 1 as x tends to infinity.

```
> plot(pi(floor(x))/Li(x), x=2..5000);
```



```
>
```

Problems

```
> restart;
```

Here's a few problems to play with:

Problem 1

Write a Maple procedure `pp()` such that `pp(x)` returns a list of all prime pairs $[p, p+2]$ with $p+2$ less than or equal to x .

Note for large x the calculation of `pp(x)` may require a long time and may be difficult to interrupt gracefully. Be sure to save your work before testing your code.

Note you may want to make use of the `ithprime()` procedure and perhaps also Maple's "while" control statement.

As a test `pp(20)` should return `[[3, 5], [5, 7], [11, 13], [17, 19]]`.

Problem 2

Solve **numerically** the initial value problem

```
> ode:=diff(y(x),x,x)+6*x*diff(y(x),x)-4*x^3*y(x)=0;
```

$$ode := \left(\frac{\partial^2}{\partial x^2} y(x) \right) + 6x \left(\frac{\partial}{\partial x} y(x) \right) - 4x^3 y(x) = 0$$

```
> init:=y(0)=0,D(y)(0)=1;
```

$$init := y(0) = 0, D(y)(0) = 1$$

and then plot the solution for $x=-1..2$. Do not load the `plots` package explicitly but instead use the plot procedure `dplot()` defined by

```
> dplot:=plots[odeplot];
```

$$dplot := plots_{odeplot}$$

Problem 3

Solve **symbolically** the initial value problem

```
> ode:=diff(y(x),x,x)+4*diff(y(x),x)+13*y(x)=exp(-2*x)*cos(3*x);
```

$$ode := \left(\frac{\partial^2}{\partial x^2} y(x) \right) + 4 \left(\frac{\partial}{\partial x} y(x) \right) + 13 y(x) = e^{(-2x)} \cos(3x)$$

```
> init:=y(0)=2,D(y)(0)=-1;
```

$$init := y(0) = 2, D(y)(0) = -1$$

Then plot the solution for $x=-0.25..2$.

Problem 4

Solve the initial value problem

```
> ode4:=diff(T(t),t)=-k*(T(t)-A(t));
```

$$ode4 := \frac{\partial}{\partial t} T(t) = -k (T(t) - A(t))$$

```
> A:=t->cos(2*t);
```

$$A := t \rightarrow \cos(2t)$$

```
> k:=1.5;
```

$$k := 1.5$$

```
[ > init4:=T(0)=0;
```

```
                                init4 := T(0) = 0
```

```
[ You can think of this problem as modeling an insulated box heated by an external source. Here  
A(t) is the ambient or external temperature and T(t) is the resulting temperature inside the box.  
Plot both A(t) and T(t) for t=0..10 on a single plot. Note the phase shift in T(t).
```

```
[ >
```

```
[ >
```

```
[ >
```