# Lesson 3: Solving Equations; Floating-point Computation

```
> restart;
```

## A hard equation

Last time we were looking at this equation.

```
> eq := x * sin(x) = Pi/2;
```

$$eq := x \sin(x) = \frac{1}{2}\pi \tag{1.1}$$

Maple didn't know the solutions.

```
> solve(eq,x,AllSolutions);
```

$$RootOf\left(2\_Z \sin(\_Z) - \pi\right) \tag{1.2}$$

Maple doesn't know the solutions, even though there are two that are easy to guess. It gave us a numerical value

```
> evalf(%);
```

$$-1.570796327 \tag{1.3}$$

and this turned out to be

```
> identify(%);
```

$$-\frac{1}{2}\pi \tag{1.4}$$

The **identify** command is sometimes able to guess a formula that produces a decimal number. It isn't foolproof though. We can verify that this really is a solution by plugging it in to the equation, using **eval**.

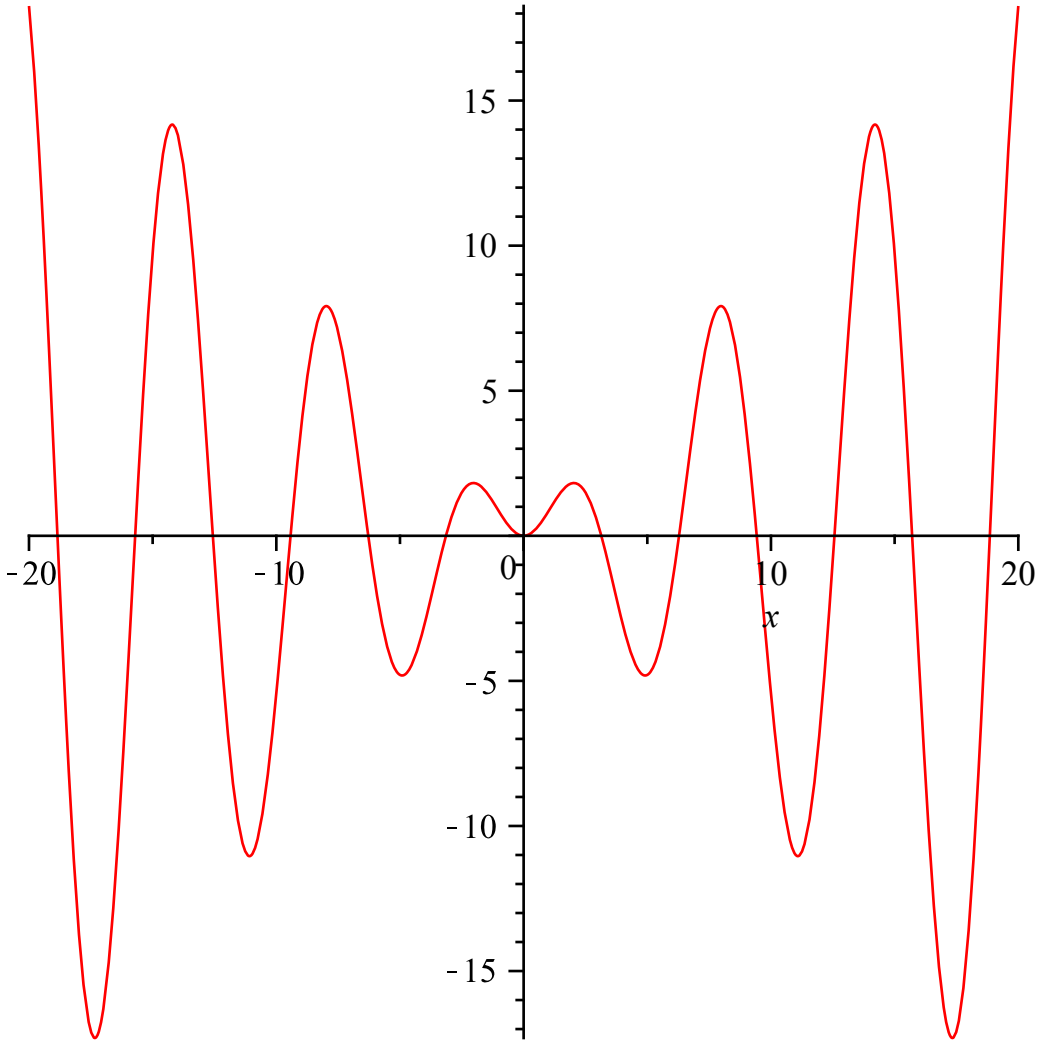
```
> eval(eq, x = -1/2*Pi);
```

$$\frac{1}{2}\pi = \frac{1}{2}\pi \tag{1.5}$$

## Plotting

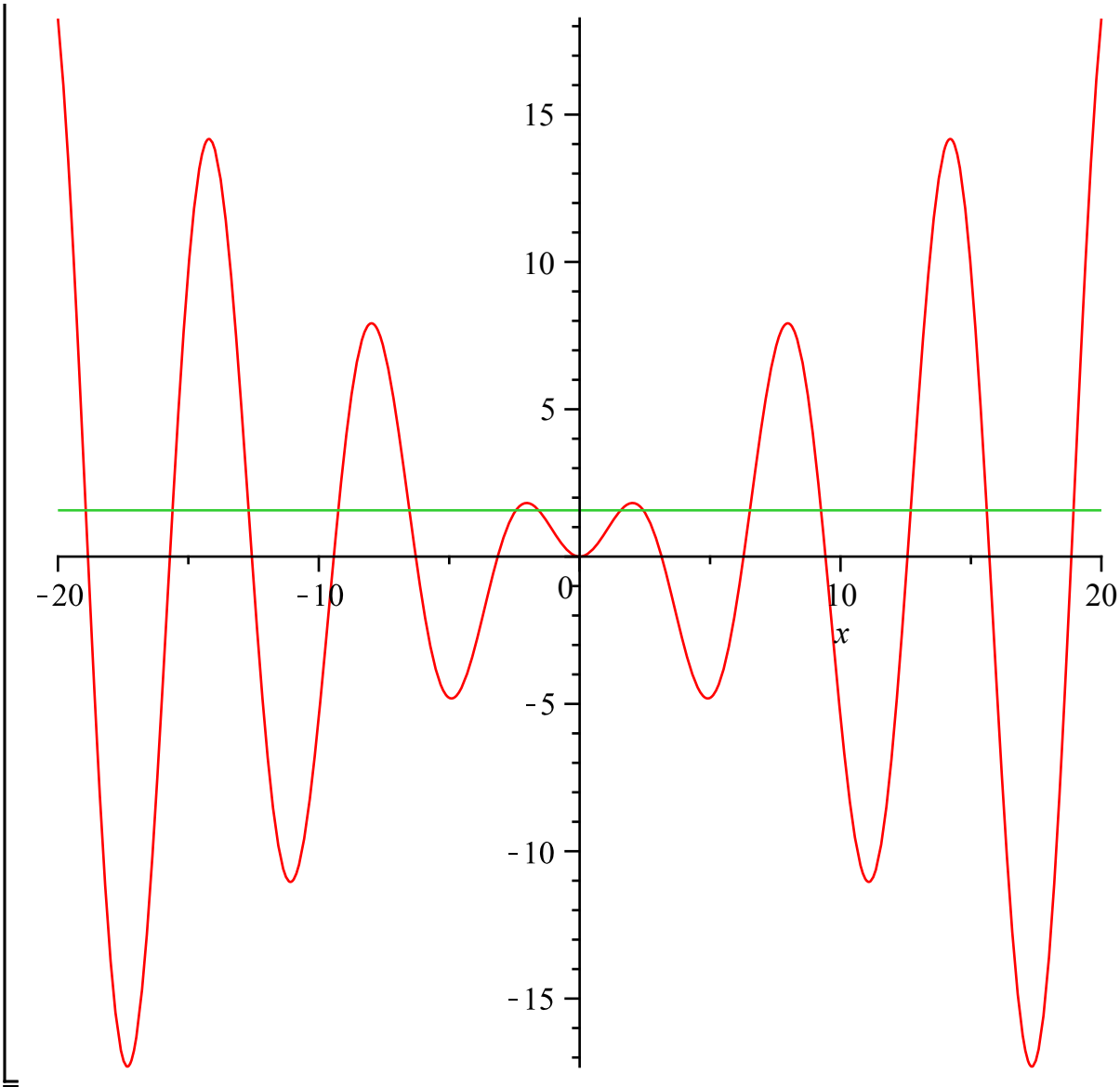
Are there other solutions? It's almost always a good idea to draw a graph.

```
> plot(x*sin(x), x=-20..20);
```




It would be handy to see the line $y = \frac{\pi}{2}$ too. Maple can plot several things in the same graph. You put the different items to plot within square brackets.

```
> plot([x*sin(x), Pi/2], x = -20 .. 20);
```




It looks like there are infinitely many solutions.

Click inside the plot, then right-click, choose "Probe Info" and "Cursor position" or "Nearest point on line", and you can see the coordinates of the "target" cursor. This lets you locate any point on the plot with better precision than you would have guessing "by eye", especially for points not on the axes.

You can also resize the plot by grabbing and dragging the little boxes on the border of the plot window.

By the way, notice that you don't plot an **equation**; what I plotted were the left and right sides of the equation.

```
> plot(eq, x=-20..20);
```

```
Error, invalid input: plot expects its 1st argument, p, to be
of type {set, array, list, rtable, algebraic, procedure, And
(`module`, appliable)}, but received x*sin(x) = (1/2)*Pi
```

I could have extracted the left or right side of the equation with the commands **lhs** and **rhs**.

```
> lhs(eq);
```

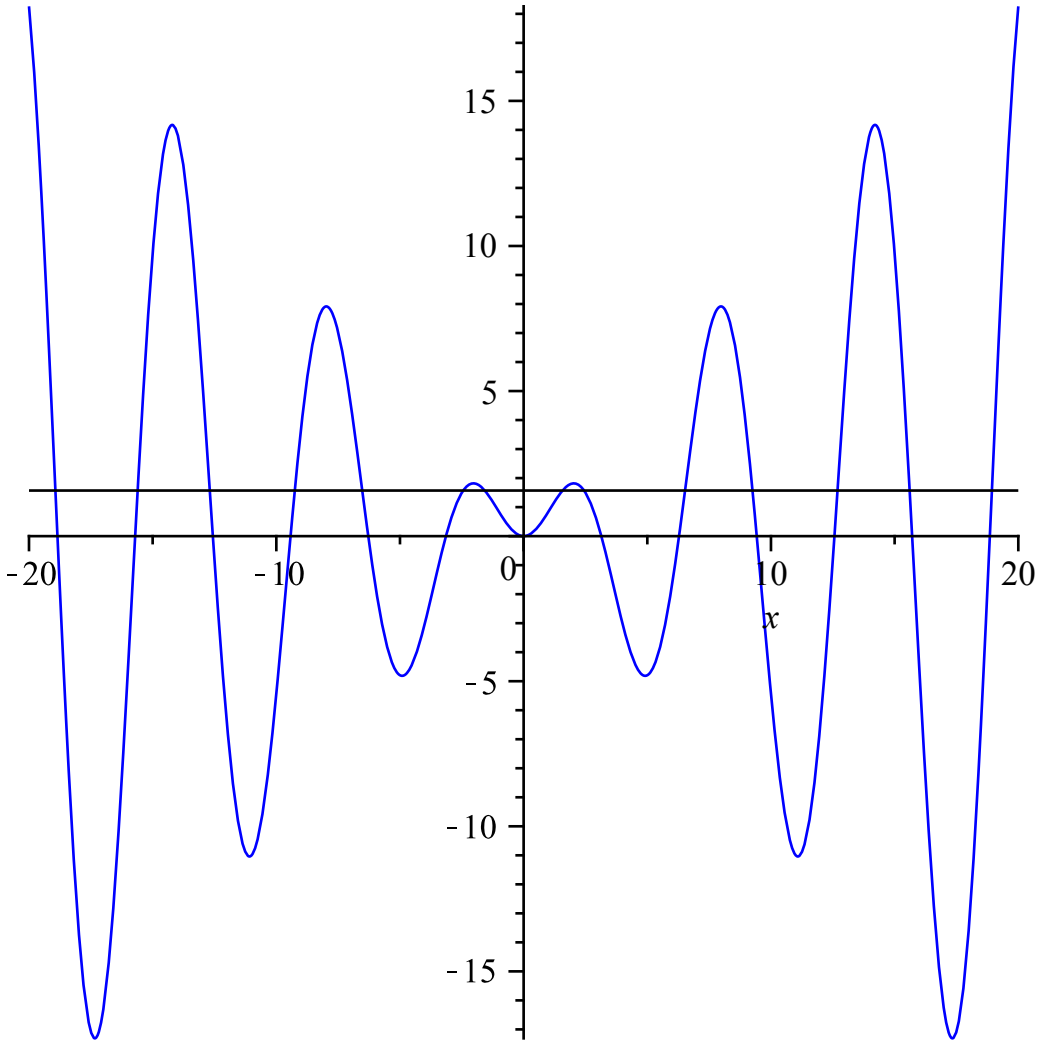
$$x \sin(x) \qquad \textbf{(2.1)}$$

```
> rhs(eq);
```

$$\frac{1}{2}\,\pi \qquad \textbf{(2.2)}$$

So I could have said something like this. While I'm at it, I'll change the colours.

```
> plot([lhs(eq), rhs(eq)], x = -20 .. 20, colour=[blue,black]);
```




# fsolve

In addition to **solve** which finds exact solutions (when possible), Maple has **fsolve** which finds a numerical approximation to a solution of an equation, even those that "solve" can't handle. Let's see how it handles our equation:

```
> fsolve(eq);
```

$$-1.570796327 \qquad \textbf{(3.1)}$$

That's the same solution we got before (probably not a coincidence: **evalf** used **fsolve** to get a value for the **RootOf**). If we want a different solution, we can specify an interval we want **fsolve** to look in. For example:

```
> fsolve(eq, x = 5 .. 10);
```

$$6.526260523 \qquad \textbf{(3.2)}$$

Is there a formula for that one?

```
> identify(%);
```

$$6.526260523 \tag{3.3}$$

Probably not.

If **fsolve** can't find a solution in the interval (usually because there isn't any), it returns unevaluated. For example, it looks like there won't be any solution between $x = 5$ and $x = 6$.

```
> fsolve(eq, x = 5 .. 6);
```

$$fsolve\left(x \sin(x) = \frac{1}{2}\,\pi, x, 5..6\right) \tag{3.4}$$

Usually **fsolve** returns one solution (if it can find one). The exception is for polynomials (or equations where both sides are polynomials in the variable), where **fsolve** returns all the real solutions (or all solutions in the specified interval).

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

$$p := x^5 + x^3 - 3\,x + 1 \tag{3.5}$$

```
> fsolve(p = 0, x);
```

$$-1.231063087, 0.3492680180, 1. \tag{3.6}$$

```
> fsolve(p, x = 0 .. 1);
```

$$0.3492680180, 1. \tag{3.7}$$

If you do want complex solutions, you can add the option **complex**.

```
> fsolve(p, x, complex);
```

$$-1.23106308721416, -0.0591024653986264 - 1.52389151122041\,I, \\ -0.0591024653986264 + 1.52389151122041\,I, 0.349268018011410, 1. \tag{3.8}$$

Here's another difficult equation to solve.

```
> eq := sin(x)^2 = exp(-x)*cos(x);
```

$$eq := \sin(x)^2 = e^{-x}\cos(x) \tag{3.9}$$

By the way, note that **exp** is the exponential function.

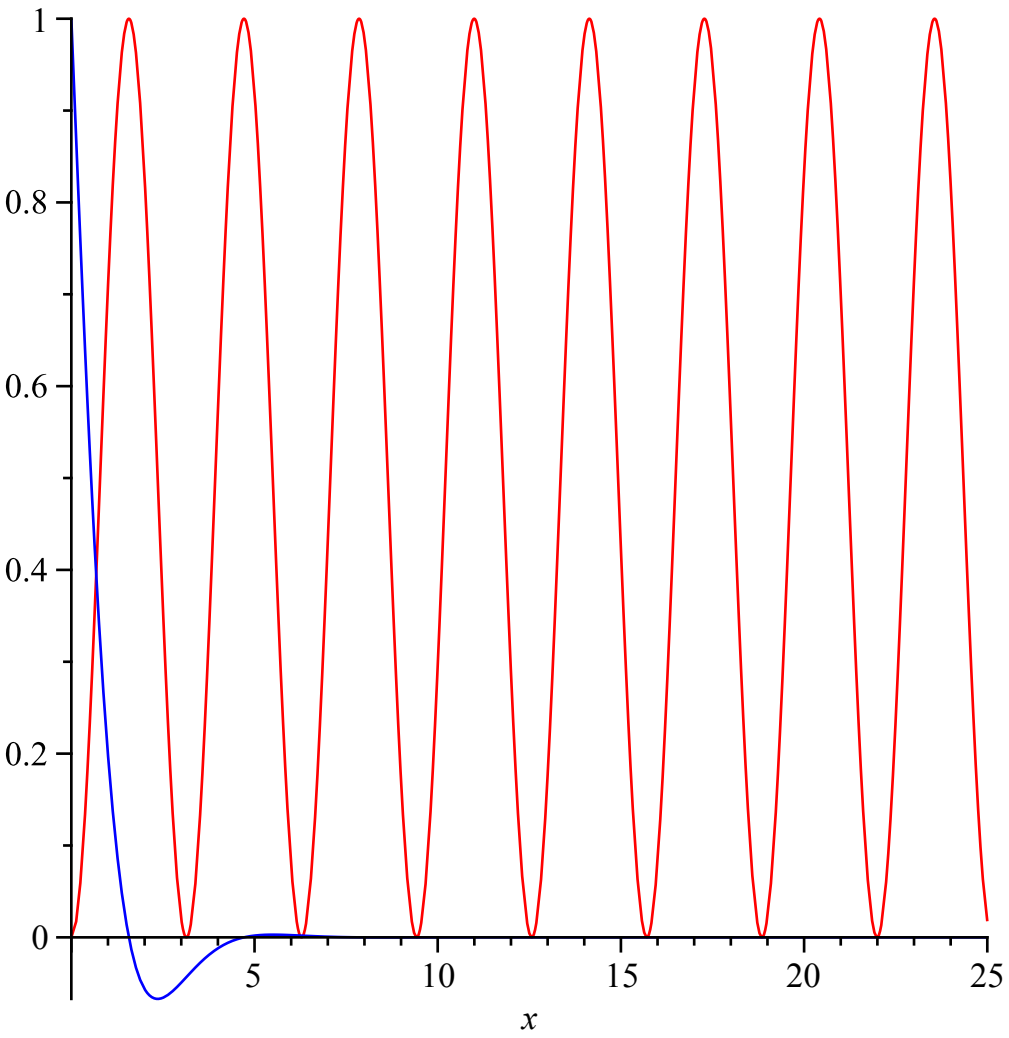
In Maple input, it's **exp(-x)**, not **e^(-x)**; to Maple, **e** is nothing special, just another name. Maple shows exp(-x) as $e^{-x}$ in output, though if you look closely, you'll see that the e is in a different font than what it would use for the name e.

```
> e^(-x);
```

$$e^{-x} \tag{3.10}$$

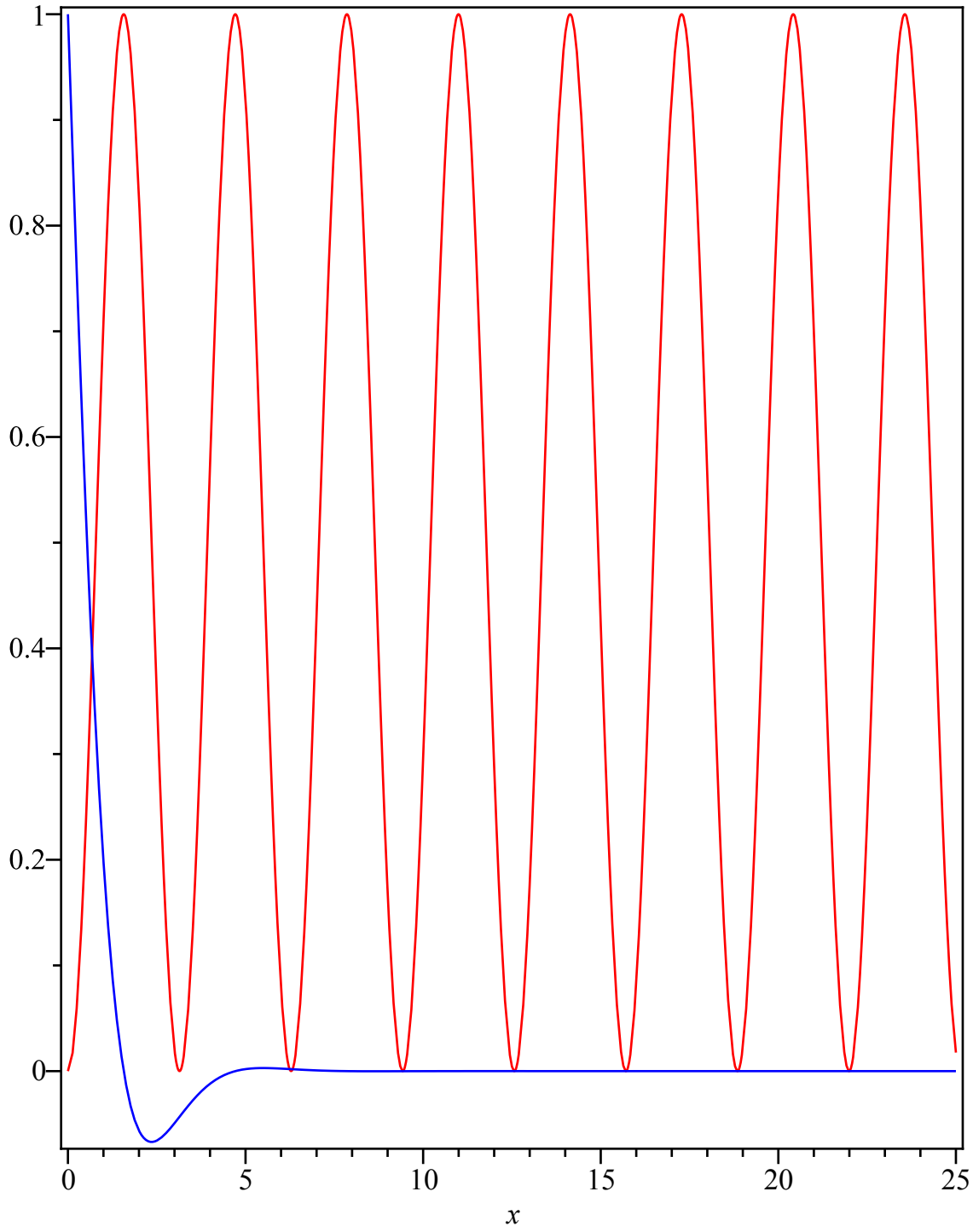
Let's say I'm interested in positive solutions.

```
> plot([lhs(eq),rhs(eq)], x = 0 .. 25, colour=[red,blue]);
```



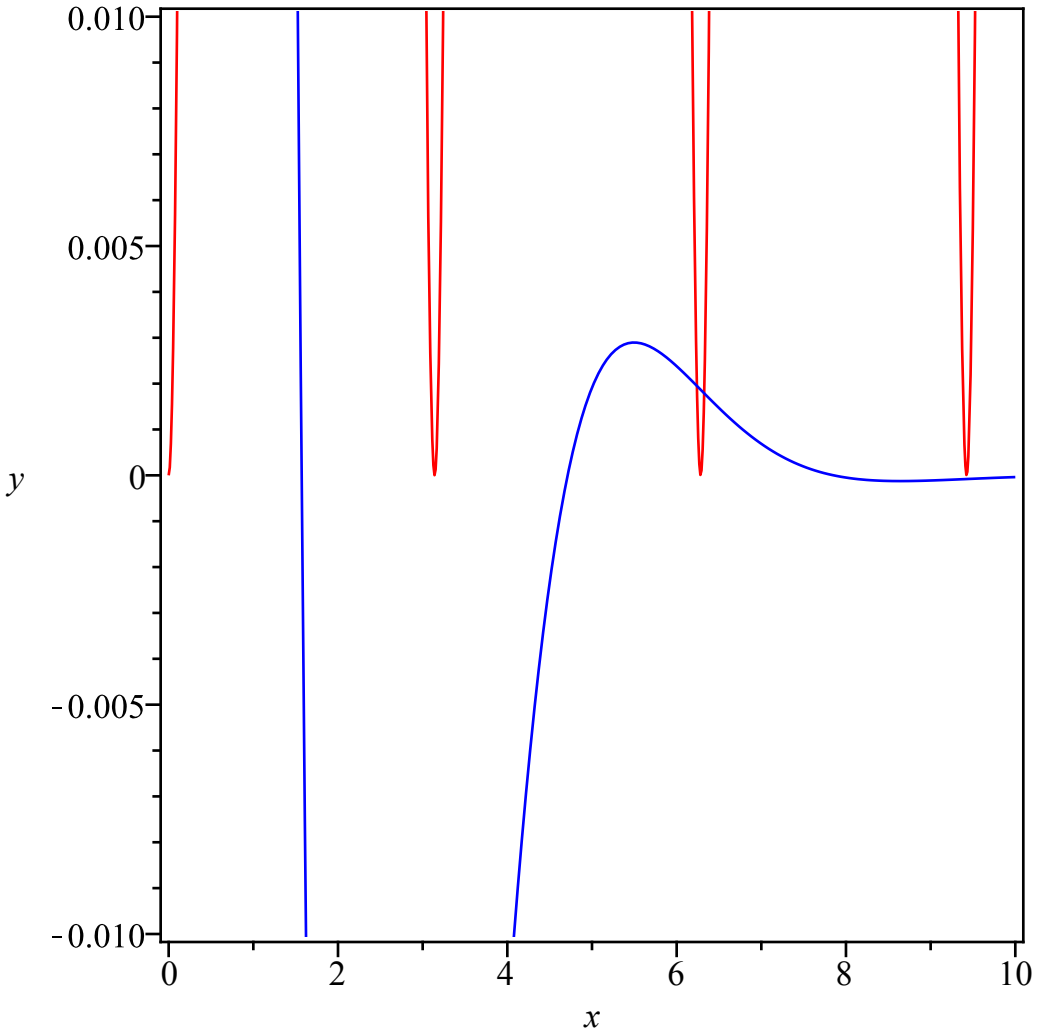

It's a bit hard to see the blue curve because part of it is hidden by the x axis. We can get that out of the way.

```
> plot([lhs(eq),rhs(eq)], x = 0 .. 25, colour=[red,blue], axes
  = box);
```




There's certainly one solution near x = .63, maybe some near x = 6.3, x = 9.4, etc, but it's hard to tell. We're interested in what happens near y = 0: most of the graph is wasted from that point of view. Notice that Maple automatically chooses the y interval to accommodate the curves it's plotting. But we can tell plot what y interval we want, if we don't like the one Maple chose.

```
> plot([lhs(eq),rhs(eq)], x = 0 .. 10, y = -0.01 .. 0.01,
  colour=[red,blue], axes = box);
```




It looks like there are solutions near 6.3, but maybe not near 9.4.

```
> fsolve(eq, x = 6.1 .. 6.5);
```

$$6.325488468 \tag{3.11}$$

```
> fsolve(eq, x = 9.2 .. 9.6);
```

$$fsolve\left(\sin(x)^2 = e^{-x}\cos(x), x, 9.2..9.6\right) \tag{3.12}$$

Actually a bit of thought and analysis explains why this is so. When $x$ is more than 5 or so, the $e^{-x}$ makes the right side very close to 0. The left side, being the square of something, is always $\geq 0$, and = 0 only when $x$ is a multiple of $\pi$. If $x$ is near an even multiple of $\pi$, $\cos(x) > 0$; the right side is greater than the left side when $x$ is exactly an even multiple of $\pi$, so they should be equal at some points on either side of those multiples of $\pi$. On the other hand, if $x$ is near an odd multiple of $\pi$, $\cos(x) < 0$ so there's no chance of a solution.

# Floating point computation and roundoff error

As we saw in Lesson 1, Maple will write decimals or floating-point numbers with a certain number of significant digits, specified by the variable "Digits". Here are some floating-point numbers, with the default Digits = 10.

```
> evalf(Pi), evalf(exp(20)), evalf(exp(-7)), evalf(exp(-20));
```

$$3.141592654,\ 4.851651954\ 10^{8},\ 0.0009118819656,\ 2.061153622\ 10^{-9} \qquad \textbf{(4.1)}$$

These are not exact values, they are approximations. To see more clearly what's going on, I'll use Digits = 3.

Caution: Don't use small values of Digits if you really want to calculate something. We'll see why.

```
> Digits:= 3:
  evalf(Pi), evalf(exp(20)), evalf(exp(-7)), evalf(exp(-20));
```

$$3.14,\ 4.85\ 10^{8},\ 0.000912,\ 2.06\ 10^{-9} \qquad \textbf{(4.2)}$$

The first and third could be represented as $3.14\ 10^{0}$ and $9.12\ 10^{-4}$, but Maple doesn't display numbers using "scientific notation" if the exponent is small.

**Roundoff error** is the error that occurs in a calculation due to the fact that you're only using a finite number of digits. That's not "error" as in "mistake", it's just that the results of arithmetic with a finite number of digits can't be the same as what they would be in exact arithmetic, with arbitrarily many digits. The computer does the best it can with what is available: the result of each addition, multiplication, division etc. is as close as possible to the exact result of that operation on the same numbers, given that you only have Digits digits available. For example, to divide 1 by 3, the exact result is 1/3 = .333333..., but with Digits = 3 this must be rounded to just .333. And then if you multiply the result by 3, you get .999.

```
> 1.0/3.0;
```

$$0.333 \qquad \textbf{(4.3)}$$

```
> % * 3.0;
```

$$0.999 \qquad \textbf{(4.4)}$$

Many implementations of floating point actually store and calculate numbers in base 2 (converting to base 10 for output), but Maple actually uses base 10. Also, many calculators store more digits than they display, but Maple doesn't: with Maple floating point, what you see is what you get. This makes it useful for investigating the effects of roundoff error.

Notice that floating-point arithmetic does not obey the usual rules of algebra. In algebra you would have **(1/a)* a = 1**. Not in floating-point.

```
> 1 / 3.0 * 3.0;
```

$$0.999 \qquad \textbf{(4.5)}$$

Similarly, in algebra you would have a + b - a = b. Not in floating point.

```
> 9.99 + 6.66 - 9.99;
```

$$6.61 \qquad \textbf{(4.6)}$$

How did that happen? Let's look at it in two steps.

```
> 9.99 + 6.66; % - 9.99;
```

$$16.6$$

$$6.61 \qquad \textbf{(4.7)}$$

The result of the first addition should be 16.65, but since Maple is using Digits = 3 it rounds that to 16.6.

Then it does 16.6 - 9.99 = 6.61.

Often roundoff error only affects the last digit or two of a calculation, but sometimes it makes a big difference.

# Catastrophic cancellation

One case where roundoff error can be severe is if you subtract two numbers that are very close together. This is called "catastrophic cancellation".

```
> 1.24 - 1.23;
```

$$0.01 \tag{4.8}$$

We subtracted two numbers with 3 significant digits, and the result has only one significant digit.

For an example where this kind of thing can occur, think of the definition of derivative:

$$f'(a) = \lim_{h \to 0} \frac{(f(a+h) - f(a))}{h}$$

The numerator requires subtracting two numbers $f(a+h)$ and $f(a)$ that are typically very close together; the smaller $h$ is, the worse the roundoff error will be. So actually trying this numerically is going to run into problems. Let's say you wanted $f'(3)$ where $f(x) = \frac{3}{x}$. The mathematical answer is -1/3. But try it with Digits = 3 and some values of h.

```
> Q:= (3.0/(3.0+h) - 3.0/3.0)/h;
```

$$Q := \frac{\frac{3.0}{3.0+h} - 1.00}{h} \tag{4.9}$$

```
> eval(Q,h=0.1);
```

$$-0.310 \tag{4.10}$$

Not bad: try a smaller h.

```
> eval(Q,h=0.01);
```

$$-0.400 \tag{4.11}$$

```
> eval(Q,h=0.001);
```

$$-1.00 \tag{4.12}$$

```
> eval(Q,h=0.0001);
```

$$-10.0 \tag{4.13}$$

With Digits = 10, things would go better for these values of h.

```
> Digits:= 10:
> eval(Q,h=0.1);
  eval(Q,h=0.01);
  eval(Q,h=0.001);
  eval(Q,h=0.0001);
```

$$-0.3225806440$$
$$-0.3322259200$$
$$-0.3332224000$$
$$-0.3333220000 \tag{4.14}$$

But even then, if we continue to decrease h we eventually run into trouble.

```
> eval(Q,h=0.00001);
```

$$-0.3333400000 \tag{4.15}$$

```
> eval(Q,h=0.0000001);
```

$$-0.3340000000 \quad \textbf{(4.16)}$$

```
> eval(Q,h=0.000000001);
```

$$-0.4000000000 \quad \textbf{(4.17)}$$

```
> Digits:= 3:
```

One good way to think about this is in terms of absolute and relative errors. In approximating a "true value" T by an approximate value A, the **absolute error** is |T-A|. The **relative error** is |T-A|/|T|.
In representing a number in floating point the relative error should be at most 5 * 10^(-Digits).
When two numbers are multiplied or divided, the relative error of the result is at most (approximately) the sum of the two relative errors. This usually doesn't cause any problems.
When two numbers are added or subtracted, the absolute error of the result is at most the sum of the two absolute errors. If the result is close to 0, this means the relative error can be much larger than the relative errors of the original numbers. The absolute error might be small, but if you multiply the result by a large number (as we did in approximating the derivative) it will multiply the absolute error also.

# Maple objects introduced in this lesson

RootOf
I
%
%%
%%%
AllSolutions
about
eval
identify
plot
lhs
rhs
exp
colour
fsolve
..
complex