fsolve

Use `fsolve` to have Maple use numerical approximation techniques (as opposed to algebraic methods) to find a decimal approximation to a solution of an equation or system of equations.

The syntax of `fsolve` is the standard Maple syntax:

`fsolve(what,how)` ;

where "what" stands for the equation (or system of equations) to be solved and "how" refers to the variable(s) being solved for.

To read more about setting up equations for solution, see the description of the `solve` command.

Using `fsolve` to solve a single equation: Because `fsolve` uses numerical techniques rather than algebraic ones, it is required that the number of equations be precisely the same as the number of variables being solved for. So, when fsolving one equation, it is crucial that there be exactly one unspecified variable in the equation.

There are two ways to use the `fsolve` command. The first is precisely like the `solve` command:

```
> fsolve(x^2+5*x=17,x);
```

-7.321825380, 2.321825380

This demonstrates that `fsolve`, like `solve`, knows how many roots to expect of a polynomial and will attempt to find them all (even if some are complex). When solving a transcendental equation, `fsolve` is usually content to find one solution.

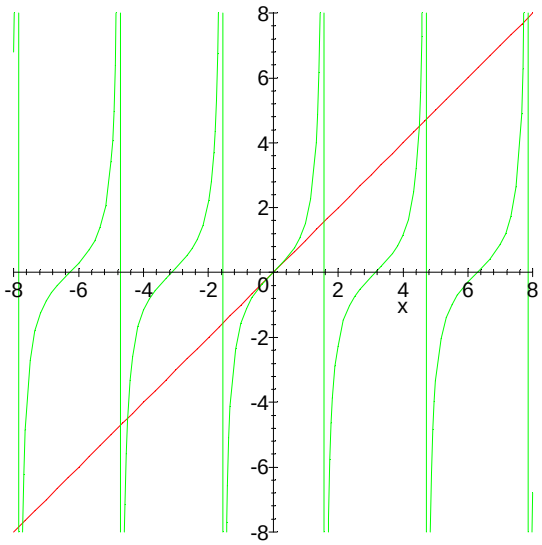
The second way to use `fsolve` is especially important when an equation has many solutions, and you want to pick out a specific one. In this version of `fsolve`, it is possible to specify a domain (interval) in which the solution should be found. For example, consider the equation

```
> eqn:=x=tan(x);
```

$$eqn := x = \tan(x)$$

As we can see from plotting both sides, this equation has many solutions:

```
> plot({x,tan(x)},x=-8..8,-8..8);
```



Just using fsolve on this equation will find one solution:

```
> fsolve(eqn,x);
```

4.493409458

Now, suppose we want to find the solution between 6 and 8. Then we enter:

```
> fsolve(eqn,x,x=6..8);
```

7.725251837

```
> tan(");
```

7.725251841

So we have found the solution to about 7 decimal places.

What can go wrong? Aside from syntax errors, there are a few things that can go wrong when using `fsolve`. First, as with solve, it is possible that the "variable" in the equation to be solved has already been given a value (perhaps one that was forgotten in the course of the Maple session). This results in the following response:

```
> x:=3: fsolve(x^2=4,x);
Error, (in fsolve) invalid arguments
```

The other things that can go wrong involve Maple's seeming inability to find a solution. This can result from one of two situations: first, there might be no solution -- second, the numerical procedure being used by Maple might need a little assistance from the user. For example:

```
> x:='x':fsolve(sin(x)=exp(x^2),x);
```

$$\text{fsolve}(\sin(x) = \mathbf{e}^{(x^2)}, x)$$

This "non-response" from Maple indicates that it cannot find a solution. But that is because this equation has no solutions. Sometimes, `fsolve` chooses its initial approximation poorly and subsequently is unable to find a solution even if it exists. In this case Maple returns a message to this effect, and suggests choosing a "different" starting interval. In this case, using the second version of `fsolve` with specified domain will remedy the problem. (Of course, to find the appropriate domain, the most reasonable thing to do is plot the two sides of the equation and look for the intersection point!).

Finally , `fsolve` will return an error message if there is a different number of equations than unknowns:

```
> fsolve(a*x=1,x);
Error, (in fsolve) should use exactly all the indeterminates
```

Of course, the `solve` command is able to handle this equation easily.

```
> solve(a*x=1,x);
```

$$\frac{1}{a}$$

Using `fsolve` to solve systems of equations: To be consistent with its "what","how" syntax, `fsolve` requires that a system of equations be enclosed in braces { } and that the list of variables being solved for also be so enclosed. For example:

```
> fsolve({2*x+y=17,x^2-y^2=20},{x,y});
```

$$\{y = -15.75516397, x = 16.37758198\}$$

It is important to remember that the number of equations must be the same as the number of unknowns, and that no other (unspecified) variables are allowed in the equations:

```
> fsolve({a*x+y=13,b*x-y=20},{x,y});
Error, (in fsolve) should use exactly all the indeterminates
```

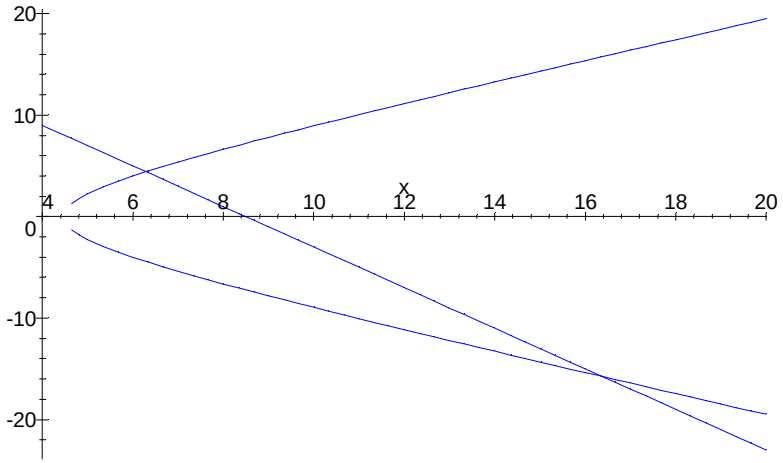
Finally, we note that it is possible (and often advisable) to give `fsolve` a domain to search in -- this is done by giving an interval for each variable separately, thereby providing fsolve with a rectangular box-like region in which to find a solution. For example, let's consider the system solved above:

```
> eqn1:=2*x+y=17: eqn2:=x^2-y^2=20:
```

We give the equations names mostly to remind you that this is possible. Experimenting around with several plots ultimately resulted in the following one, which shows that our "`fsolve`" statement above returned only one of the two

intersection points:

```
> plot({17-2*x,sqrt(x^2-20),-sqrt(x^2-20)},x=4..20,color=blue);
```



We can force `fsolve` to find the leftmost one as follows:

```
> fsolve({eqn1,eqn2},{x,y},{x=4..8,y=0..10});
```

$$\{y = 4.421830634,\ x = 6.289084683\}$$

The syntax here is important! First comes the set of equations to solve (enclosed in braces), then the set of variables to solve for (enclosed in braces) and then the list of ranges for the variables (enclosed in braces). Only the third of these (the list of variable ranges) is optional when solving systems of equations. The other two must be present.

As usual with `solve` and `fsolve`, we can substitute this solution (just as it is!) back into the equations to make sure it is correct. First, we give it a name:

```
> s:=";
```

$$s := \{y = 4.421830634,\ x = 6.289084683\}$$

```
> subs(s,eqn1),subs(s,eqn2);
```

$$17.00000000 = 17,\ 19.99999999 = 20$$

So it seems to work.

One final note -- when you give intervals for the variables, it is necessary to give ranges for all of the variables. `Fsolve` will return nothing if intervals are specified for some but not all variables.