The package evilmath

2023-10-31 v0

Warning!
This package is *evil*. The only non-evil thing about it is its name, which doesn't hide its mischievousness.

Loading
Load with `\usepackage{evilmath}`. There is only the option `show`, which is described later.

Usage
Once you load the package, you have the macro `\evilmath` at your disposal. In order to show what it does, consider the code

```
\evilmath
\[
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = \frac{10 \cdot 11}{2} = 55.
\]
```

and the corresponding output

$$1 + 2 + 3 - 4 + 5 + 6 + 7 + 8 + 9 + 10 = \frac{10 \cdot 11}{2} = 55.$$

Do you notice something? A sign has been swapped! Even there is definitely a " + 4" in the source code, we see a " − 4" in the output.

That's the evil: `\evilmath` makes the characters "+" and "-" math active (i.e. sets their `\mathcode` to `"8000`). Every time they are encountered in a math formula, they change with a certain probability. The probability is by default 10%, but you can change it in two ways:

1. with the optional parameter to `\evilmath`: with `\evilmath[3]` on average one sign out of three will change, with `\evilmath[5]` one out of five, and so on;
2. the default can be changed globally by using `\evilmathprobability{`⟨*number*⟩`}`.

Internally, random numbers are generated by `\int_rand:n`. The seed changes at every compilation and so will the resulting PDF. You can set the seed globally by using `\evilmathseed`. For example, this documentation has `\evilmathseed{826349}` in the preamble, so that I can type it without going crazy.[1]

Showing the wickedness
The package loads the `color` package, because it allows you to show which signs have been changed. You can reveal them either by loading the package with the option `show`, or by means of the macro `\showevilmath` (while `\hideevilmath` hides again the changed signs). The code

[1] Which is, in fact, the main reason for `evilmath` to exist…

```
\showevilmath
\[
1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10
= \evilmath[3] 1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10
\]
```

yields

$$1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10 = 1 + 2 - 3 - 4 + 5 + 6 - 7 + 8 - 9 - 10$$

and as you can see no sign on the left-hand side and three on the right-hand side have been changed.

Clearly you can expect the number of changed signs to mirror the set probability only over a larger number of signs. Let us say, a line with 45 signs

```
\evilmathprobability{3}
\[
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\]
```

$$+-+$$

with 12 changed signs, while

```
\evilmathprobability{15}
\[
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\]
```

$$+-+$$

with two changed signs. I'd say that's reasonable. Clearly, with `\evilmathprobability{1}` you'll change *all* signs

```
\evilmathprobability{1}
\[
\sin(a+b) = \sin(a) \cos(b) + \cos(a) \sin(b)
\]
```

$$\sin(a - b) = \sin(a)\cos(b) - \cos(a)\sin(b)$$

which is funnily still correct (though not what you typed).

Can you make it even more wicked?
If you *really* hate your co-author(s), then change the name of this package into something harmless, issue `\evilmath` in the package code itself, and tell your collaborator(s) it's needed to type better matrices, or something.

Can you make it less wicked?
Yes. With `\goodmath` everything goes back to normal. Note that evil is local but good is global: while `\evilmath` uses local assignments, `\goodmath` makes things good again *globally*.