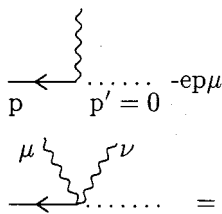


Wiggly Lines

A. G. W. Cameron
Harvard University

During the summer of 1985 the IBM PC Technical Group of the Boston Computer Society carried out a project to test scientific word processing programs. As part of the project the reviewers were given a challenging set of benchmarks which were to be implemented if possible on the various packages. In addition to various standard mathematical constructs which T_EX can handle with ease, the benchmarks contained an electrical circuit diagram, a Feynman diagram (used in high energy physics), and organic chemical structural formulas. I set out to demonstrate that plain T_EX could do all of these, and I succeeded apart from having to make a few minor font substitutions.

Two of the special benchmarks which I mentioned above involved drawing diagonal lines, and the Feynman diagram involved wiggly lines, one vertical and several diagonal. The following shows the essential part of this benchmark.



The method by which I produced slanted lines in the circuit diagram and the chemical structural formulas is a simplification of the method used with the above wiggly lines, so I give here only the technique for doing the wiggly lines. When I started I was unaware of the procedure for drawing diagonal lines due to Amy Hendrickson (TUGboat, Vol. 6, No. 2, p. 83). Nevertheless, my initial trials used a technique similar to hers, laying down overlapping dots. However, since I was developing the method using an Epson FX-80 printer, which produces large and fuzzy dots, I began to worry about the different dot sizes that would occur with other output devices, and also about positional uncertainties in laying down the dots.

Therefore I used square rules. These were made a half point on the side in the above example. For the circuit diagram and the chemical structures I was forced to go to squares one point on the side in order to stay within the available T_EX memory.

The procedure involved using T_EX as a plotting device. Suppose the current position is (x_0, y_0) and that we wish to place a square at the relative coordinates (x, y) . Suppose also that (x, y) is at

the center of the left side of the square that is to be plotted. Then I would use an `\hskip` for the distance x . I would then describe the rule as a `\vrule height $(y + 0.5h)$ width h depth $(-y + 0.5h)$` , where h is the length of the side of the square rule. Then I would go back to the origin with an `\hskip` for the distance $-(x + h)$. It may readily be seen that this procedure uses two `\hskip`'s where one could be made to do in repeated plotting of points, but it simplifies the calculations.

To get a vertical wiggly line as in the above example one simply sets (x_0, y_0) to the base of the wiggly line and integrates the equations $\dot{y} = c_1$ and $\dot{x} = -c_2x$ using the `\loop...repeat` structure, where c_1 and c_2 are constants. To get the slanted wiggly lines in the above diagram one just applies a rotation of the coordinate system to the above solution so that new points (x', y') are plotted. To illustrate all of this I give below the code to produce a wiggly line slanted up to the right.

```
% define 12 variables:
\newcount\vone ... \newcount\vtwelve
% set up initial conditions
$\vone=0\vtwo=0\vtthree=7000\vfive=\vtwo
\loop % begin the iteration; assume that we
% have an x (\vtwo) and y (\vone) and rotate
% the coordinates to obtain x' and y'. I take
% the angle of rotation to be 30°, so that
% sin 30° = 0.5 and cos 30° = 0.866.
\vseven=\vfive\multiply\vseven by 866
\divide\vseven by 1000
\vnine=\vone\divide\vnine by 2
\advance\vnine by \vseven
\vsix=\vone\divide\vsix by 10
% here we avoid an overflow
\multiply\vsix by 866\divide\vsix by 100
\veight=\vfive\divide\veight by 2
\multiply\veight by -1
\advance\veight by \vsix
% now \veight is y' and \vnine is x'
\vtten=16384\advance\vtten by \veight
\veleven=16384\advance\veleven by -\veight
\hskip\vnine sp %move x' and raise rule by y'
\vrule height\vtten sp width32768sp
depth\veleven sp
\vtwelve=\vnine\advance\vtwelve by 32768
\hskip-\vtwelve sp
% if not done, increment y, then x
\ifnum\vone<1800000\advance\vone by 20000
\advance\vtwo by\vtthree\vfour=-\vtwo
\divide\vfour by 10\advance\vtthree by\vfour
\advance\vfive by \vtwo\repeat$
```