About that machine epsilon ...

James F. Epperson

September 13, 2014

On August 29, 2014, I had a very pleasant email exchange with Prof. Kathleen Shannon of Salisbury State University, who had some questions about the Matlab solution code I gave in the Solutions Manual for Exercise 10 in §1.3. This webpage is based on that discussion.

The solution code is kind of obvious: Take some initial value—which I'll call the *seed* and denote it by x_0—and then decrement it recursively with a factor I'll call θ ($\theta < 1$), according to $x_{n+1} = \theta x_n$ until we have $1 = 1 + x_N$. Call this value x_N our (approximate) machine epsilon, $x_N = \mathbf{u}_* \approx \mathbf{u}$. Now it should be obvious that $\mathbf{u}_*$ depends on both x_0 and θ, but it also will be affected by the rounding of the finite precision arithmetic, and since we are trying to approximate a very small quantity, that rounding could be a big issue.

I wrote my code thinking as a numerical analyst, meaning I took a small seed and a large decrement—so the code would not run long and would not overshoot the true value of $\mathbf{u}$ by very much—but neither could be represented exactly in floating point arithmetic. My code returns $\mathbf{u}_* = 1.101642356786233 \times 10^{-16}$. Prof. Shannon wrote a code that didn't care about efficiency but worried more about exact arithmetic: her seed was 1 and her decrement was $\theta = 1/2$ (*not* 0.5). She got $\mathbf{u}_* = 1.110223024625157 \times 10^{-16}$. Since her value is larger than mine, it has to be a better approximation to $\mathbf{u}$.

An interesting exercise would be to compute different values of $\mathbf{u}_*$ for different values of x_0 and θ and plot the results. I may do that, or something very similar, in the near future.

I suspect that any future edition of the text will include a version of this discussion, along with a footnote crediting Prof. Shannon.