3450:427/527 Applied Numerical Methods I, Kreider

Notes on Stopping Criteria

The iterative root-finding methods rely on a stopping criterion to determine when to terminate the procedure. The absolute error $|x_n - x_{n-1}| < \epsilon$ works well when the root is large in magnitude, but works poorly when the root is small in magnitude. The relative error $\left|\frac{x_n - x_{n-1}}{x_n}\right| < \epsilon$ works well when the root is small in magnitude, but is not as good when the root is large in magnitude. The reason is that if $\epsilon = 10^{-d}$, then

the absolute error gives at least $d-1$ correct digits after the decimal point if the number is not written in floating point representation, and

the relative error gives at least $d-1$ correct digits; if the number is written in floating point representation, this means at least $d-1$ digits in the mantissa are correct.

Depending on the method used, you might see more than $d-1$ digits; for example, each iteration of Newton's Method essentially doubles the number of correct digits, so you might go from 4 to 8 digits even if $d-1$ is only 6.

Examples

1. If the root is $r = 123.456789$ and $d = 6$, you would expect the absolute error stopping criterion to give the approximation 123.45679 and the relative error to give 123.46. Here, the root is large, and the absolute error condition is better.

   If the root is $r = .000123456$, the absolute error condition would be expected to give .00012 (that's 5 digits after the decimal) and the relative error condition would give .00012346. The root is small, and the relative error condition is better.

2. Let $f(x) = (x^2+1)(x-.001)$. The only root is .001, which is small. Let $\epsilon = .1$. Using Newton's Method with a starting value of $x_0 = 50$, the relative error condition gives an approximation of .00100000000 in 14 iterations, while the absolute error condition gives an approximation of .001004182041 in 13 iterations. The absolute error condition simply tells the algorithm to stop too soon. When ϵ is reduced to .001, the absolute error condition gives an approximation of .00100000000 in 14 iterations – we now get enough correct digits after the decimal point to be satisfied. But if the actual root were equal to .000001, that condition would not be sufficient.

3. Let $f(x) = (x^2+1)(x-1000.)$. The only root is 1000, which is big. Let $\epsilon = .1$. Using Newton's Method with a starting value of $x_0 = 2000$, the relative error condition gives an approximation of 1004.347812418753 in 4 iterations. There are in fact 3 correct digits in the mantissa, but that's not good enough here. The absolute error condition gives 1000.000002785269 in 6 iterations, which has 5 correct digits after the decimal point - better than we expect. When ϵ is reduced to .001, the relative error condition gives 1000.000002785269 in 6 iterations, and the absolute error condition gives 1000.000000000000 in 7 iterations. For the big root, the relative error condition stops the calculation prematurely.

Note that the stopping criterion used does NOT change the computed approximations x_n; the criterion merely tells the algorithm when to stop.

Obviously, the best way to write a root-finding routine is to make sure that BOTH criteria are satisfied before stopping so that the results are trustworthy whether the root is small or big. To do this, you would 'AND' the two conditions:

```
if (abserr<tol) & (relerr<tol), break, end
```

Unfortunately, in the text, the authors 'OR' the conditions, which means that the worse condition is always used to stop the calculations. When I reviewed the new edition last fall, I asked them to fix the code, but they didn't.