## Week 3: Monday, Sep 7

# Another error analysis

Consider the following quadratic equation:

$$z^2 - 2z + \mu = 0.$$

The roots of this equation are

$$z = 1 \pm \sqrt{1-\mu}.$$

If we Taylor expand the square root about 1, we find

$$z_- = 1 - \left(1 - \frac{\mu}{2} + O(\mu^2)\right) = \frac{\mu}{2} + O(\mu^2) = \frac{\mu}{2}\left(1 + O(\mu)\right).$$

Let's compare the two formulae numerically over a range of $\mu$:

```
mu = 10.^linspace(-1,-20);
z1 = 1-sqrt(1-mu);
z2 = mu/2;

loglog(mu, z1, '-', mu, z2, ':');
legend('Quadratic formula', 'Taylor estimate', ...
       'Location', 'NorthWest');
```

The results, shown in Figure 1, suggest something is amiss. An error analysis is in order. Keeping track of rounding errors, our computation using the quadratic formula looks like

$$\begin{aligned}
\mathrm{fl}(z_-) &= \left(1 - \sqrt{(1-\mu)(1+\delta_1)}(1+\delta_2)\right)(1+\delta_3) \quad && |\delta_1|, |\delta_2|, |\delta_3| \leq \epsilon_{\text{mach}} \\
&= \left(1 - \sqrt{1-\mu}(1+\delta_4)\right)(1+\delta_3) && |\delta_4| \lesssim 1.5\epsilon_{\text{mach}} \\
&= z_- \left(1 + \delta_4 \frac{\sqrt{1-\mu}}{z_-}\right)(1+\delta_3) \\
&= z_- \left(1 + \frac{2\delta_4}{\mu} + O(\epsilon_{\text{mach}}\mu)\right)(1+\delta_3).
\end{aligned}$$

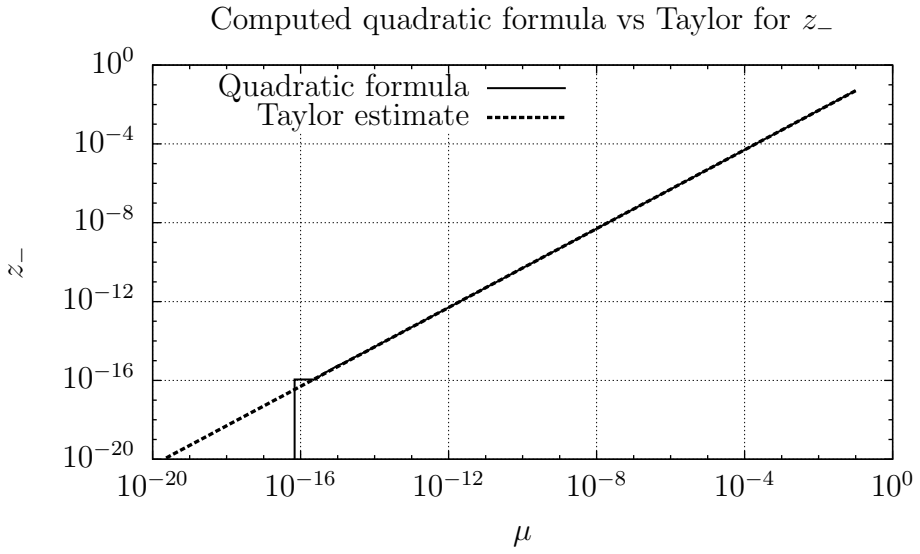


Figure 1: `z1 = 1-sqrt(1-mu)` and `z2 = mu/2` in double precision.

As $\mu$ gets small, the dominant term in the relative error is $\lesssim 3\epsilon_{\text{mach}}/\mu$, again due to *cancellation*.

The difficulty in this calculation is not that the underlying problem is ill-behaved for $\mu$ near zero. Indeed, the *problem* is very well behaved! The difficulty is in the *formula* that we are using to solve the problem. Because $\sqrt{1-\mu}$ is very near one for small $\mu$, the small relative error in computing $\sqrt{1-\mu}$ — bounded by around 1.5 $\epsilon_{\text{mach}}$ — is huge compared to the final answer. A better way to do the computation is to get rid of the subtraction. Note that if $z_-$ and $z_+$ are the two roots, then $z_- z_+ = \mu$; so

$$z_- = \frac{\mu}{z_+} = \frac{\mu}{1+\sqrt{1-\mu}}.$$

If we do a thorough analysis for this formula (left as an exercise to the reader), we find that the relative error is bounded by $3\epsilon_{\text{mach}}$ for small $\mu$.

# Error analysis of matrix-vector multiply

Now let's analyze the floating point error in a matrix-vector product. Each entry in the product vector is a dot product of a matrix row with the vector,

so a basic building block will be an analysis of error in a dot product. The error analysis for a dot product (see section 2.4.5 in the book[1]) gives

$$|\operatorname{fl}(x^T y) - x^T y| \lesssim n|x|^T|y|\epsilon_{\text{mach}}.$$

Now, notice that we could produce a vector $\delta x$ such that $|\delta x|_i \lesssim n\epsilon_{\text{mach}}|x|_i$ so that

$$\operatorname{fl}(x^T y) = (x + \delta x)^T y.$$

That is, we see the computed dot product as an *exact dot product with a perturbed vector*.

Now, this estimate implies that

$$\operatorname{fl}(Ax) = (A + \delta A)x$$

where $|\delta A_{ij}| \leq n\epsilon_{\text{mach}}|A_{ij}|$. In the 1-norm, $\infty$-norm, and Frobenius norm[2], we can easily show

$$\|\delta A\| \leq n\epsilon_{\text{mach}}\|A\|.$$

Using our earlier results about perturbation theory for matrix-vector multiply, we have

$$\frac{\|\operatorname{fl}(Ax) - Ax\|}{\|Ax\|} \lesssim n\epsilon_{\text{mach}}\kappa(A).$$

[1] The analysis we did in class was a little sloppy, and hence we got $n + 1$ rather than $n$. Given that we mostly care about the order of magnitude of error terms, this sort of sloppiness will not be critical.

[2] Whether this statement is true in the 2-norm is left as an exercise to you.

# Aside: Plotting Matlab data in LaTeX

The following is not directly related to the course material, but it's worth knowing. If you use LaTeX to typeset mathematical documents, it's not always trivial to make MATLAB figures that look "nice." Either the axes will be in a different font, or the line widths will be too narrow, or the size will look wrong, or there will be some other niggling issue. The situation is worse if you use a student version of MATLAB that prints "Student version" on every figure you produce, or if you happen to use Octave. There are, however, other graphics packages in the world. I have friends who swear by the Python `matplotlib` library, but my default is `gnuplot`. As an example, the following `gnuplot` script generated the figure in this set of notes:

```
# Compare computed quadratic formula to Taylor

set terminal epslatex size 5in,3in
set output "lec05fig.tex"
set grid
set format y "$10^{%L}$"
set format x "$10^{%L}$"
set title  "Computed quadratic formula vs Taylor for $z_{-}$"
set xlabel "$\\mu$"
set ylabel "$z_{-}$" 5, 0
set log xy
set key top left
set xrange [1e-20:1]
set yrange [1e-20:1]
set xtics 1e-20, 10000, 1
set ytics 1e-20, 10000, 1
set lmargin 10

m1 = 3
m2 = 5

plot "lec05out.txt" using 1:2 title "Quadratic formula" with lines lw m1, \
     "lec05out.txt" using 1:3 title "Taylor estimate" with lines lw m2
```

The `lec05out.txt` file was written by this wrapper around the script `lec05a.m` that appeared in the notes:

```
lec05a
fp = fopen('lec05out.txt', 'w');
for k = 1:length(mu)
  fprintf(fp, '%.16e %.16e %.16e\n', mu(k), z1(k), z2(k));
end
fclose(fp);
```