CSC 165

floating-point operations

week 11, lecture 3

Danny Heap

heap@cs.toronto.edu

www.cdf.toronto.edu/~heap/165/F09

resources: chapter 7 of course notes

http://docs.python.org/tutorial/floatingpoint.html

http://docs.python.org/library/decimal.html

http://en.wikipedia.org/wiki/IEEE_754-2008

Under the hood

Python floats use IEEE double precision: $\beta = 2$, $t = 53$, $e_{\min} = -1022$, $e_{\max} = 1023$
usually implemented in hardware.

You rarely see this directly, since floats are usually displayed in decimal.
That means that as well as the hardware rounding, you have more rounding
from binary to decimal representations. Not a problem
unless you are in CSC165 trying to understand floats.

You can work around this with `float.hex()` and `float.fromhex()`,
provided you read and write hexadecimal (base 16) fluently.

Or you can add a softare layer and user the python `decimal` module.

operations

Assume $\beta = 10, t = 3, E_{min} = -3, E_{max} = 3$

1.65×10^2

0.271×10^2

For addition, we align the terms and then add.
What happens to $1.65 \times 10^2 + 2.71 \times 10^1$?

1.65×10^2

2.71×10^1

$\times 10^3$

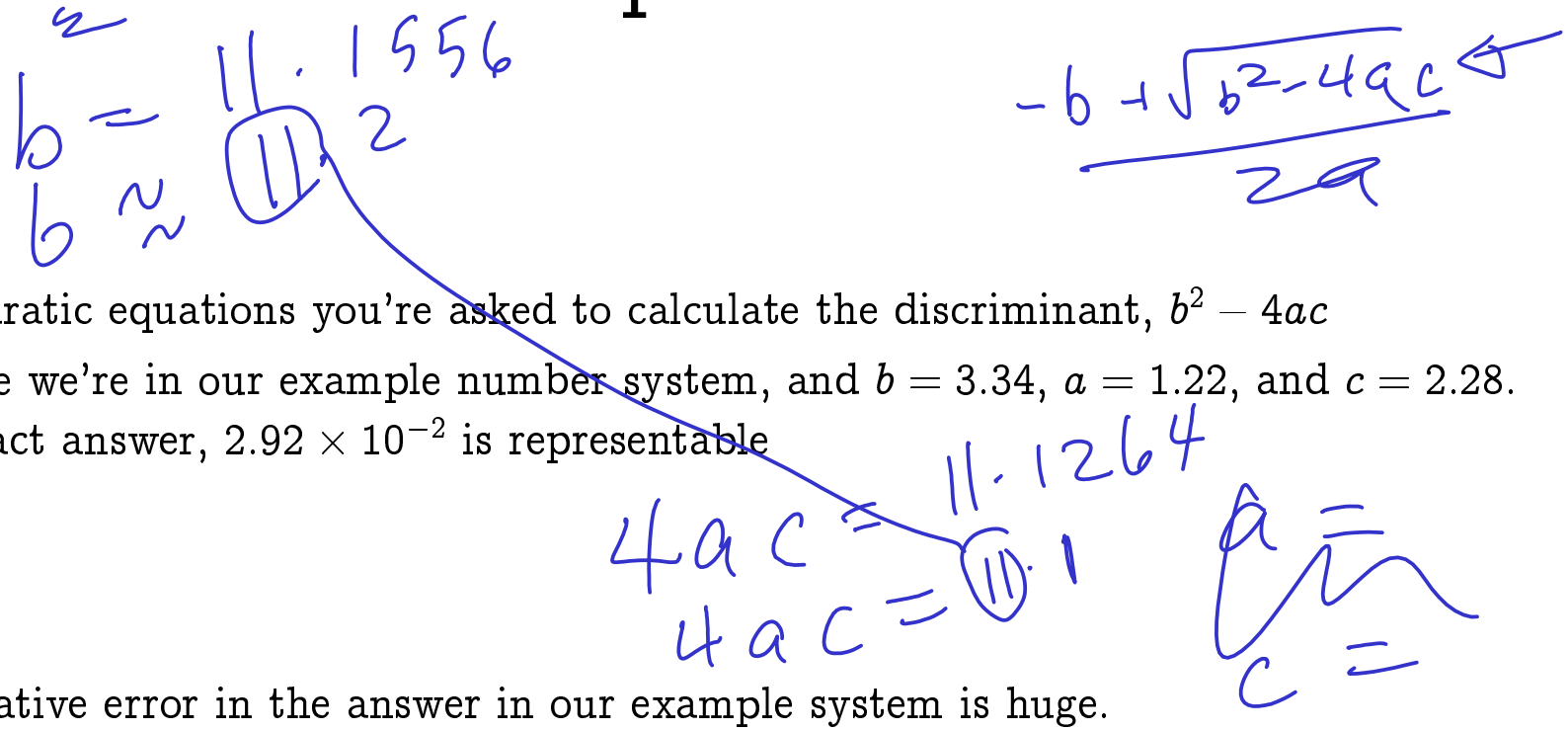
What about the product of the two numbers from the previous example?

Accumulation of error

In our same small number system, set $x = 1.00 \times 10^2$ and $y = 1.00 \times 10^{-1}$

Now
$$((((((((((x+y)+y)+y)+y)+y)+y)+y)+y)+y)+y) \neq x+(y+y+y+y+y+y+y+y+y+y)$$

catastrophic cancelation



In quadratic equations you're asked to calculate the discriminant, $b^2 - 4ac$

Suppose we're in our example number system, and $b = 3.34$, $a = 1.22$, and $c = 2.28$.
The exact answer, 2.92×10^{-2} is representable

The relative error in the answer in our example system is huge.

It's better to be lucky than good. When we carry out the entire
$(-b + \sqrt{b^2 - 4ac})/2a$, the relative error gets swamped by $-b$
and the result is a small relative error! But you can't depend on being lucky.