Printout

Monday, September 9, 2024 9:20 AM

Section 1

MANE 3351

Subsection 1

Lecture 4

Classroom Management

Agenda

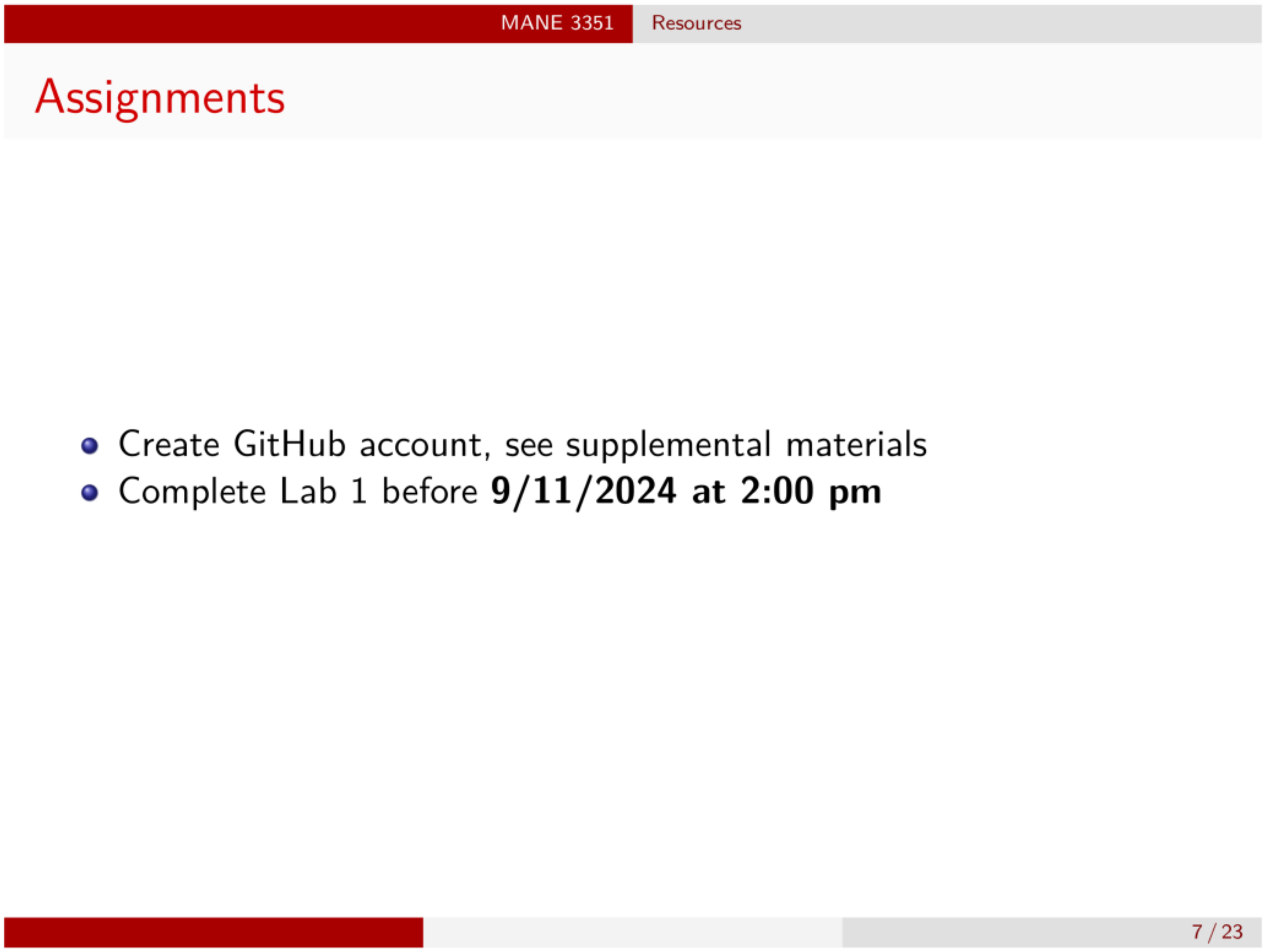
- Numerical Representations in a computer

Subsection 2

Resources

Handouts

- Lecture 4 Slides
- Lecture 4 Marked Slides



Assignments

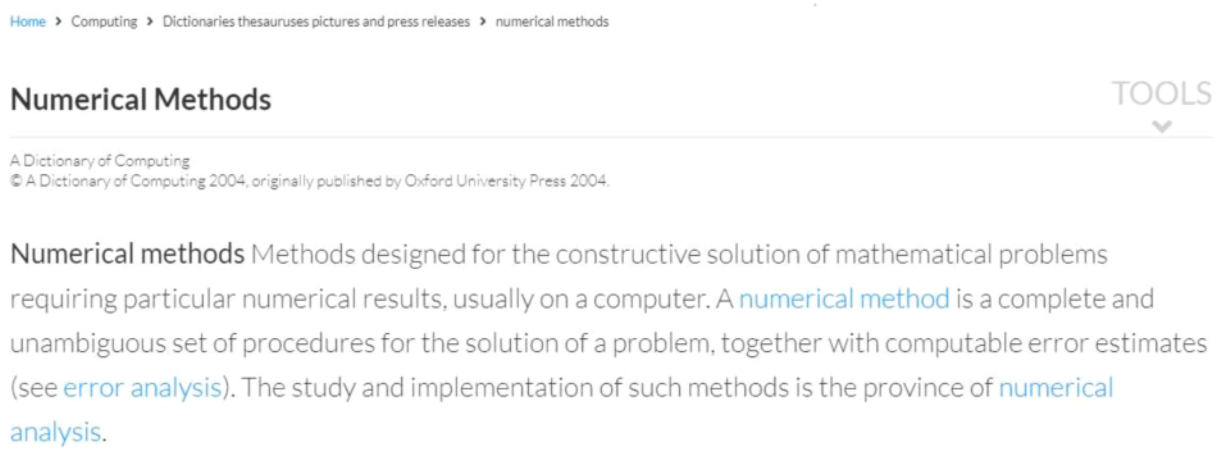
- Create GitHub account, see supplemental materials
- Complete Lab 1 before **9/11/2024 at 2:00 pm**

Subsection 3

Lecture 4 Content

Embrace Your Inner Skeptic

Definition: Numerical Methods



Home › Computing › Dictionaries thesauruses pictures and press releases › numerical methods

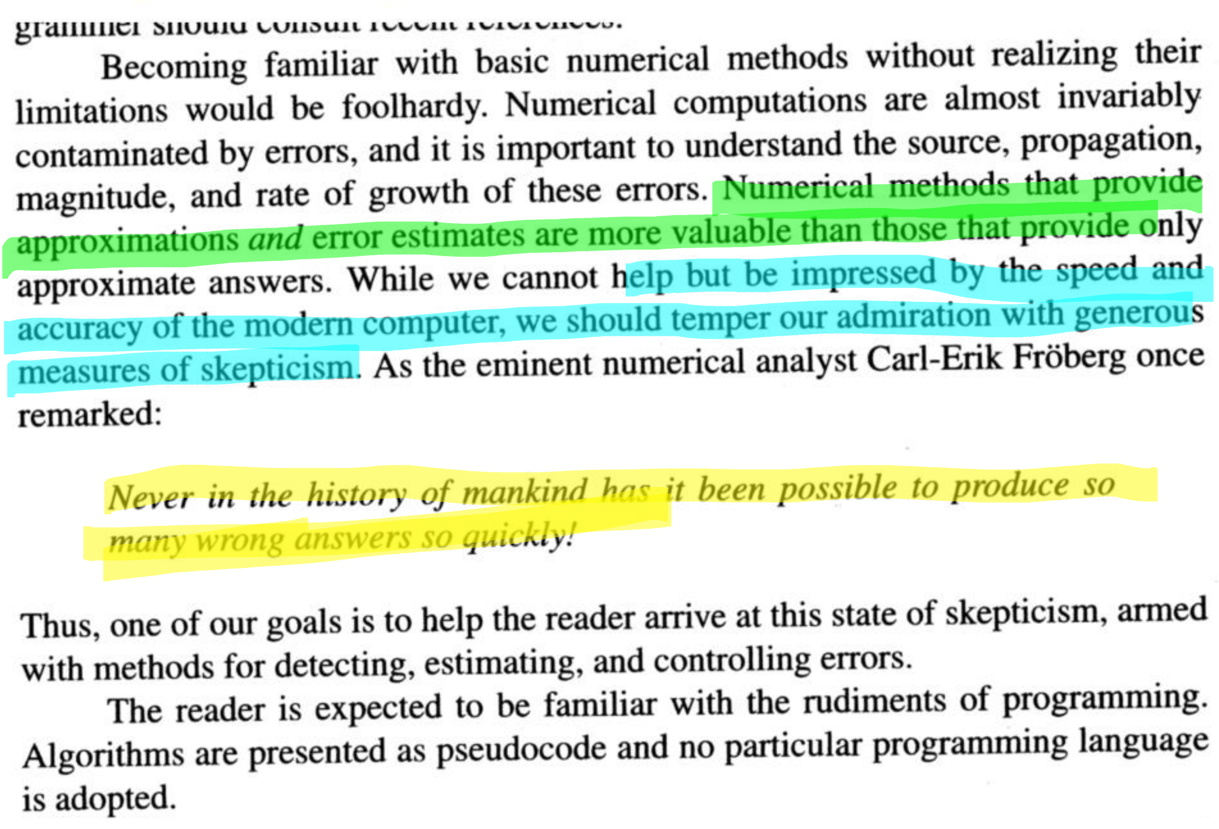
Numerical Methods

A Dictionary of Computing
© A Dictionary of Computing 2004, originally published by Oxford University Press 2004.

Numerical methods Methods designed for the constructive solution of mathematical problems requiring particular numerical results, usually on a computer. A numerical method is a complete and unambiguous set of procedures for the solution of a problem, together with computable error estimates (see error analysis). The study and implementation of such methods is the province of numerical analysis.[a]

[a] "numerical methods." A Dictionary of Computing. Retrieved August 27, 2019 from Encyclopedia.com: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/numerical-methods

Embrace your Inner Skeptic



grammer should consult recent references.

Becoming familiar with basic numerical methods without realizing their limitations would be foolhardy. Numerical computations are almost invariably contaminated by errors, and it is important to understand the source, propagation, magnitude, and rate of growth of these errors. Numerical methods that provide approximations *and* error estimates are more valuable than those that provide only approximate answers. While we cannot help but be impressed by the speed and accuracy of the modern computer, we should temper our admiration with generous measures of skepticism. As the eminent numerical analyst Carl-Erik Fröberg once remarked:

> *Never in the history of mankind has it been possible to produce so many wrong answers so quickly!*

Thus, one of our goals is to help the reader arrive at this state of skepticism, armed with methods for detecting, estimating, and controlling errors.

The reader is expected to be familiar with the rudiments of programming. Algorithms are presented as pseudocode and no particular programming language is adopted.

a

Evaluating Numerical Methods

Numerical methods should consider these two evaluation criterion :

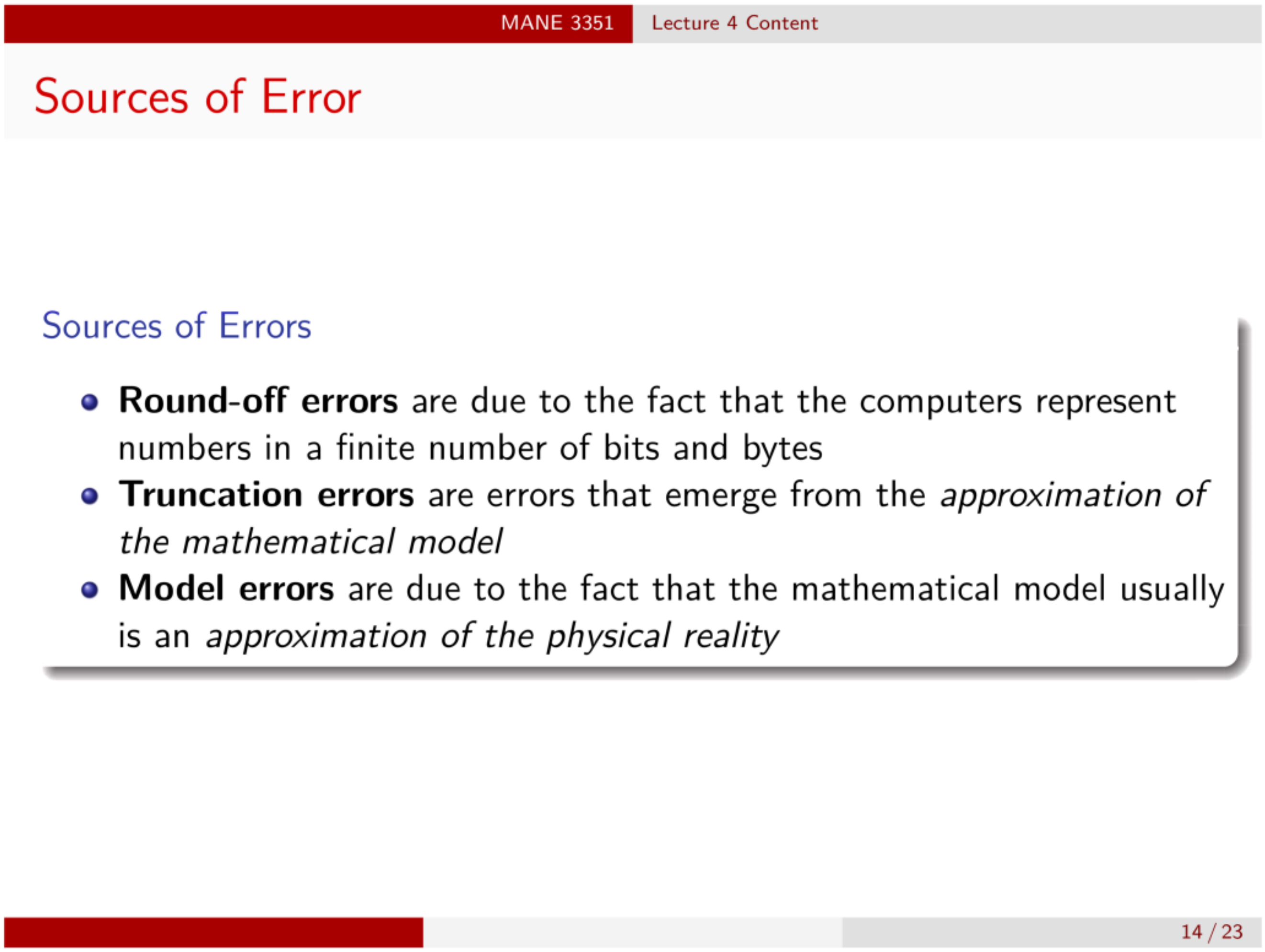
- Accuracy (or error) analysis
- Speed

Introduction to Error Analysis

Error Measurements

Let v_A be the approximate value and v_E be the exact value

- Absolute error: $|v_A - v_E|$
- Relative error: $\frac{|v_A - v_E|}{v_E}$
- Percentage error: $\frac{|v_A - v_E|}{v_E} \times 100\%$



Sources of Error

Sources of Errors

- **Round-off errors** are due to the fact that the computers represent numbers in a finite number of bits and bytes
- **Truncation errors** are errors that emerge from the *approximation of the mathematical model*
- **Model errors** are due to the fact that the mathematical model usually is an *approximation of the physical reality*

boolean

Number Types in Python

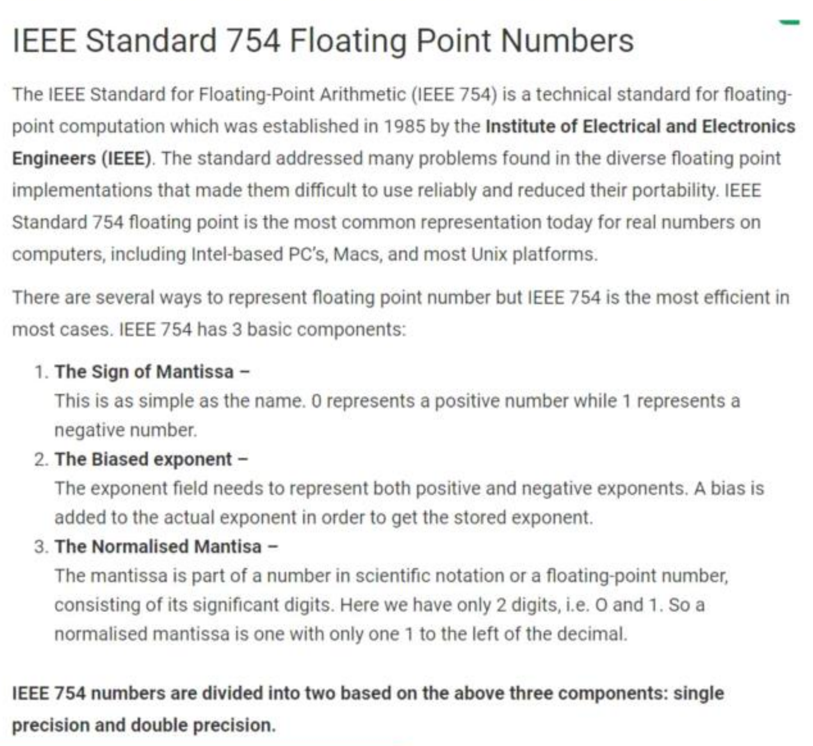
Python supports int, float, and complex.

- Integers can be represented exactly
- Float and complex variables can be represented approximately
- "Floating-point numbers are usually implemented using double in C"[a]

[a]Python 3.12.5 Documentation » The Python Standard Library » Built-in Types. Retrieved September 6, 2024 from: https://docs.python.org/3/library/stdtypes.html

Computer Implementation of Floats

IEEE Standard 754 Floating Point Numbers



IEEE Standard 754 Floating Point Numbers

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the **Institute of Electrical and Electronics Engineers (IEEE)**. The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.

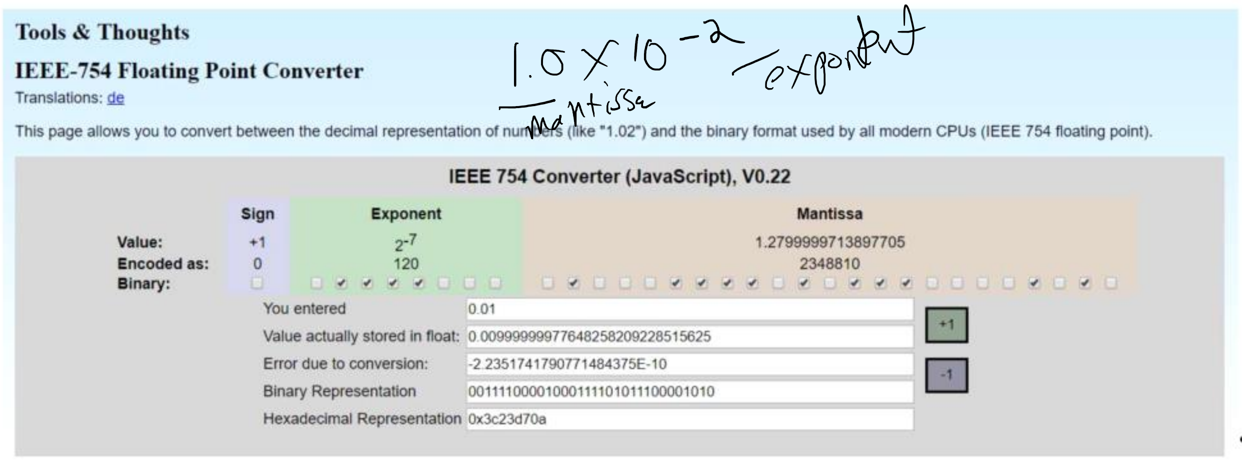
There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:

1. **The Sign of Mantissa –**
   This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
2. **The Biased exponent –**
   The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
3. **The Normalised Mantisa –**
   The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.

Floating Point Representation

$\underbrace{1.0}_{\text{mantissa}} \times 10^{-2}$ exponent



Tools & Thoughts

IEEE-754 Floating Point Converter

Translations: de

This page allows you to convert between the decimal representation of numbers (like "1.02") and the binary format used by all modern CPUs (IEEE 754 floating point).

IEEE 754 Converter (JavaScript), V0.22

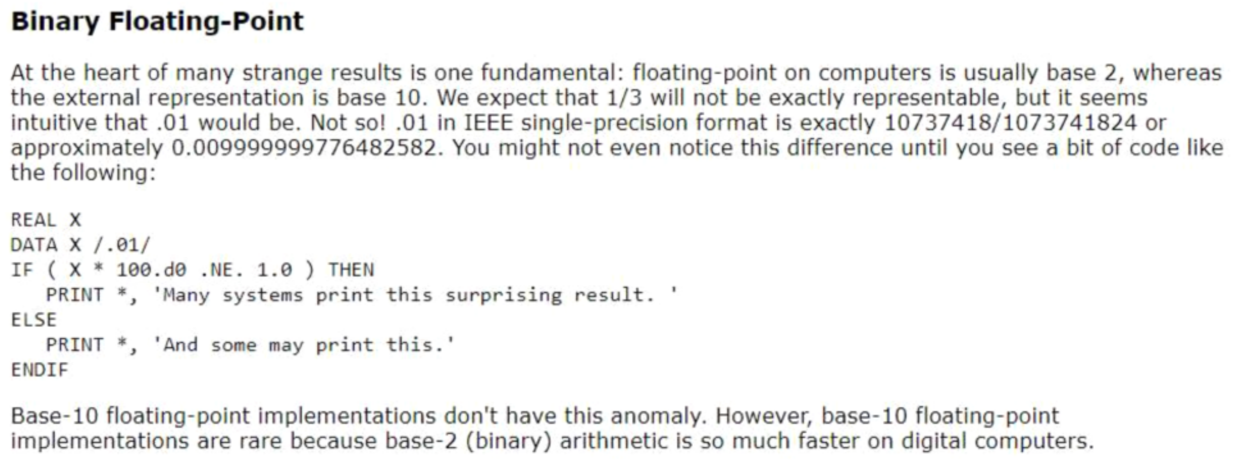
	Sign	Exponent	Mantissa
Value:	+1	2^{-7}	1.2799999713897705
Encoded as:	0	120	2348810
Binary:	☐	☐ ✓ ✓ ✓ ✓ ☐ ☐ ☐	☐ ✓ ☐ ☐ ☐ ✓ ✓ ✓ ✓ ☐ ✓ ☐ ✓ ✓ ✓ ☐ ☐ ☐ ☐ ✓ ☐ ✓ ☐

You entered	0.01
Value actually stored in float:	0.009999999776482582092285156250
Error due to conversion:	-2.2351741790771484375E-10
Binary Representation	00111100001000111101011100001010
Hexadecimal Representation	0x3c23d70a

[a]

[a]FloatConvertor. Retrieved August 28, 2019 from: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Perils of Floating Point



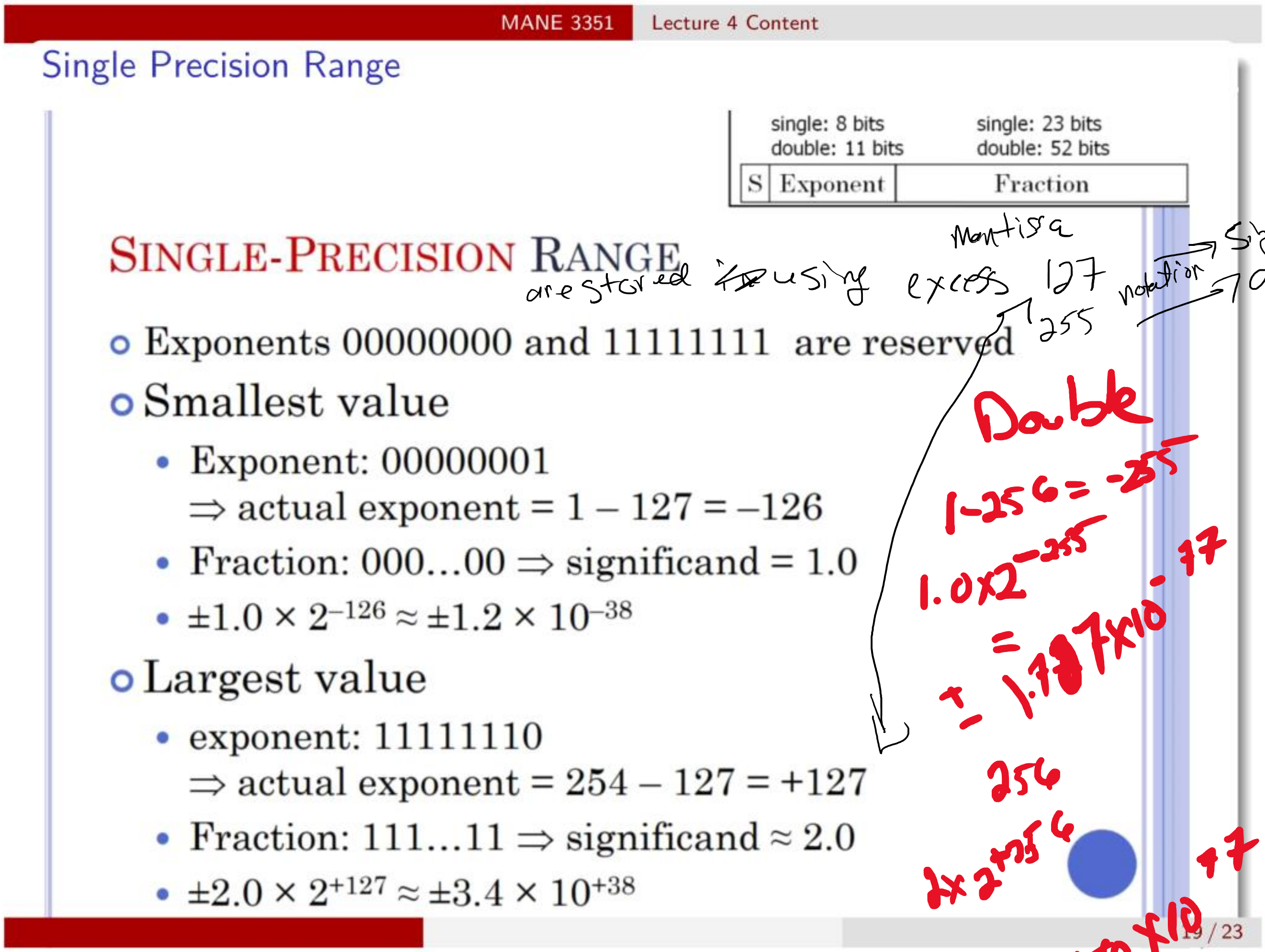
Binary Floating-Point

At the heart of many strange results is one fundamental: floating-point on computers is usually base 2, whereas the external representation is base 10. We expect that 1/3 will not be exactly representable, but it seems intuitive that .01 would be. Not so! .01 in IEEE single-precision format is exactly 10737418/1073741824 or approximately 0.009999999776482582. You might not even notice this difference until you see a bit of code like the following:

```
REAL X
DATA X /.01/
IF ( X * 100.d0 .NE. 1.0 ) THEN
   PRINT *, 'Many systems print this surprising result. '
ELSE
   PRINT *, 'And some may print this.'
ENDIF
```

Base-10 floating-point implementations don't have this anomaly. However, base-10 floating-point implementations are rare because base-2 (binary) arithmetic is so much faster on digital computers. [a]

[a] Perils of Floating Point. Retrieved August 28, 2019 from: http://www.lahey.com/float.htm



Single Precision Range

	single: 8 bits double: 11 bits	single: 23 bits double: 52 bits
S	Exponent	Fraction

Mantisa

SINGLE-PRECISION RANGE are stored ~~[illegible]~~ using excess 127 notation → Single → double

255

- Exponents 00000000 and 11111111 are reserved
- Smallest value
  - Exponent: 00000001 $\Rightarrow$ actual exponent = 1 – 127 = –126
  - Fraction: 000…00 $\Rightarrow$ significand = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- Largest value
  - exponent: 11111110 $\Rightarrow$ actual exponent = 254 – 127 = +127
  - Fraction: 111…11 $\Rightarrow$ significand ≈ 2.0
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

Double

$1-256 = -255$

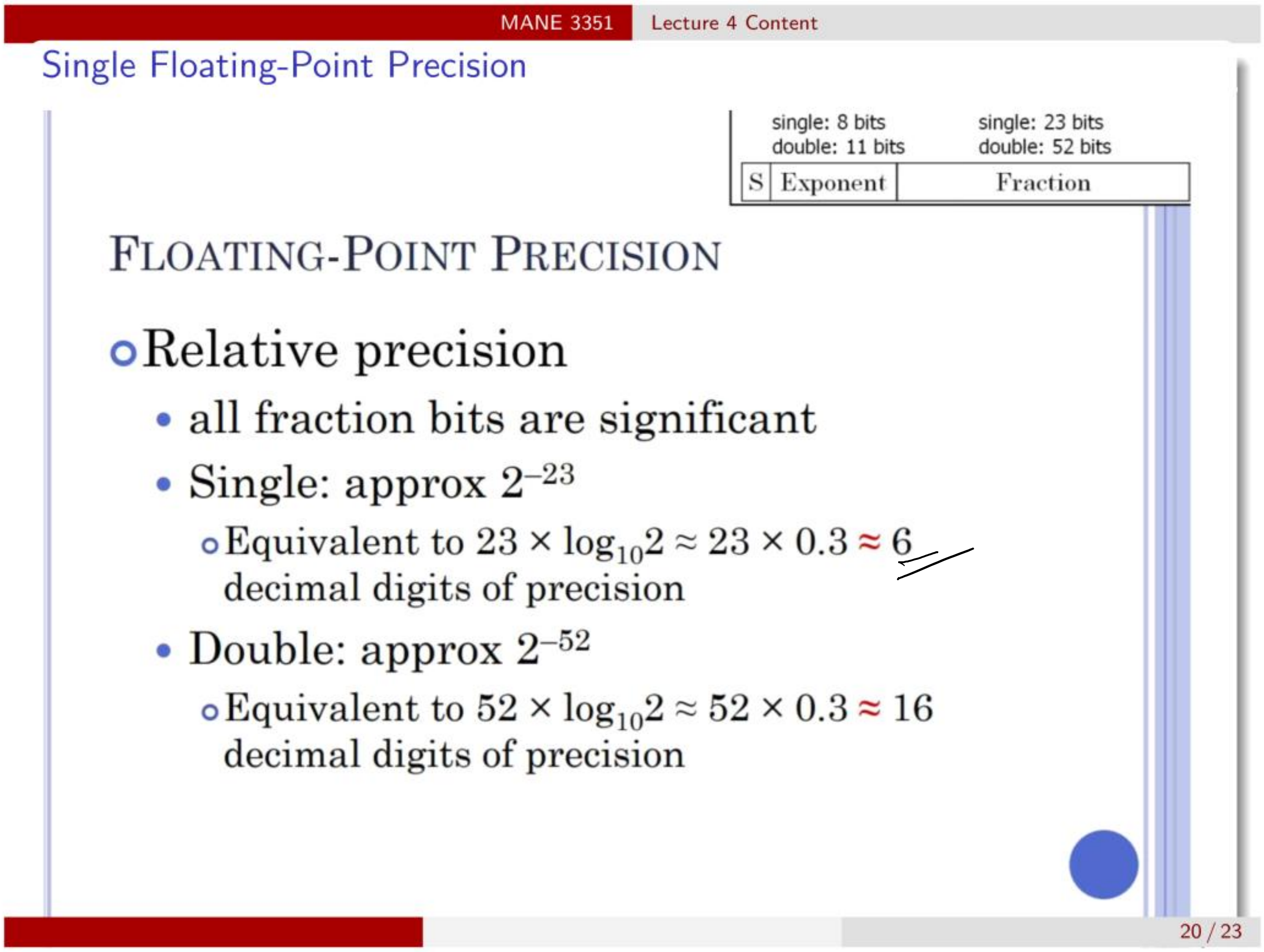
1.0×2^{-255}

$= \pm 1.787\times 10^{-77}$

256

$2\times 2^{+256}$

$2.310\times 10^{+77}$



Single Floating-Point Precision

	single: 8 bits double: 11 bits	single: 23 bits double: 52 bits
S	Exponent	Fraction

Floating-Point Precision

- Relative precision
  - all fraction bits are significant
  - Single: approx 2^{-23}
    - Equivalent to $23 \times \log_{10}2 \approx 23 \times 0.3 \approx \underline{\underline{6}}$ decimal digits of precision
  - Double: approx 2^{-52}
    - Equivalent to $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$ decimal digits of precision

Double Precision Range

Double Floating-Point Precision

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

mathematical true; not well suited for computers

$$\frac{1}{n-1}\left[\sum x_i^2 - \left(\frac{\sum x_i}{n}\right)^2\right]$$

computational better