More on Floating Point Numbers
Eigenvalues and Eigenvectors

CS 111: Introduction to Computational Science

Spring 2019 Lecture #11

Ziad Matni, Ph.D.

Administrative

- Homework #5
  - Due next week **WEDNESDAY (5/15) @ 6:00 pm**

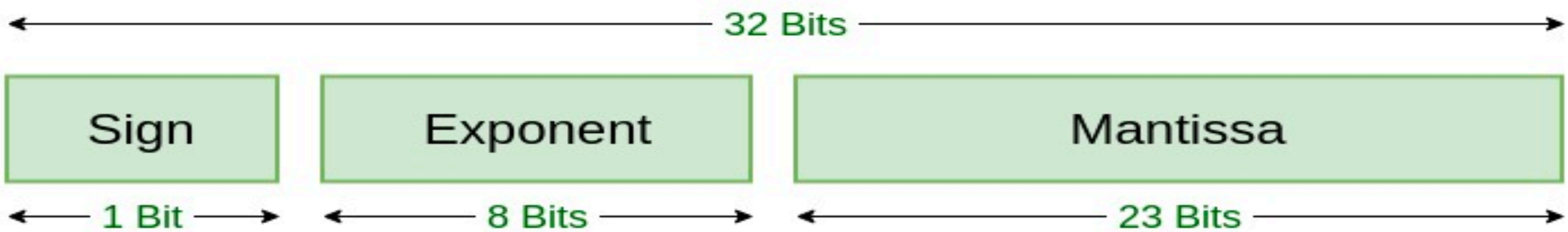
- There will **NOT** be a lecture on **Thursday, May 16th**
  - Section is STILL ON, however!
  - Syllabus updated to reflect schedule changes from original (it's a minor update)

Reviewing Your Midterm #1 Exam

- Your grades are up on GauchoSpace

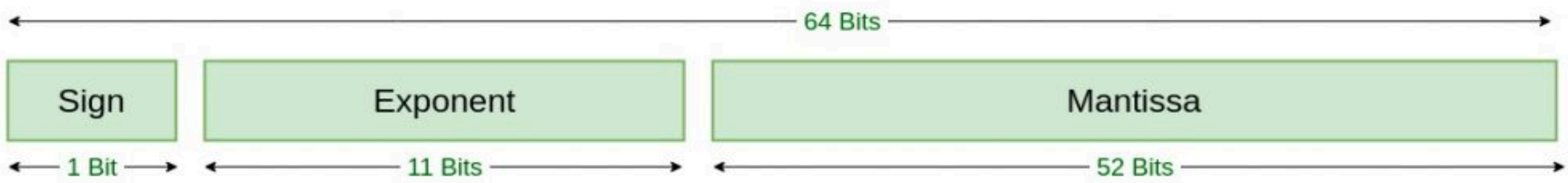
- To review your exams (optional, but not a bad idea), go to your TA's office hours:
  - Last name is A thru M — See **Steven** (Tu. 1 – 3)
  - Last name is N thru Z — See **Shiyu** (Fr. 10 – 12)

- When reviewing your exams:
  - Do not take pictures, do not copy the questions
  - TA cannot change your grade
    - If you have a legitimate case for grade change, the prof. will decide
    - Legitimate case = When we graded, we added the total points wrong
    - Not legitimate case = "Why did you take off *N* points on this question????"



0 = positive
1 = negative

Single Precision
IEEE 754 Floating-Point Standard



Double Precision
IEEE 754 Floating-Point Standard

Machine Epsilon ε

- Gives an **upper bound** on the **relative error** due to rounding in floating point arithmetic
  - Sometimes called **unit roundoff**

- Defined as the difference between 1 and the *next larger floating point number*

- A common way to find ε is to start with an initial value of $\varepsilon = 1$ and keep dividing it by 2, stopping only when $(1 + \varepsilon_{latest}) = 1$
  - This will have ε go from 2^0, to 2^{-1}, to 2^{-2}, to… 2^{-N}
    where N is the most negative exponent that will yield ε
  - Bonus: the mantissa will always be 1.000000… with this exercise

Floating Point Numbers are Discrete

- As opposed to "continuous"
  - Radical idea! Explain please!

- ε is the smallest "discrete" unit between any 2 "adjacent" FP numbers in a computer!

Relating FPs and Matrix Condition Number

- Recall: our lectures on Numerical Stability (see lect. 5, 6)

- In a computational matrix **A**, we sometime care about how small changes in it lead to either small or large changes in calculations involving **A**
  - You want small changes in **A** to yield small changes in **x** (where **Ax** = **b**), for example

Matrix Condition Number

Recall: The condition number of matrix **M** is defined as:

$$\kappa(M) = ||M|| \times ||M^{-1}||$$

i.e. kappa(M) = norm of M multiplied by the norm of the inverse of M

SEE MOLER BOOK CHAPTER FOR MORE DETAILS:

https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/lu.pdf

In Python, you can use the `.cond()` function in `linalg`:

```
numpy.lingalg.cond(M, 'fro')
```

Where **`'fro'`** indicates the Frobenius Norm, that is the norm defined as the square root of the sum of the absolute squares of its elements.

Eigenvalues and Eigenvectors

- Recall that when: $\mathbf{A\ x = \lambda\ x}$

  Then we call λ the eigenvalue and **x** is the eigenvector

  $$\lambda_1, \lambda_2, \ldots \lambda_n \in \mathbb{C}$$

- Related theorem 1:
  - A and A^T have the same eigenvalues, but usually different eigenvectors

- Didja know that?
  - Given any matrix **A**, $\mathbf{B = A + A^T}$ is always symmetrical!!?!
  - So, what would $\mathbf{B} - \mathbf{B^T}$ equal to?

- Related theorem 2:
  - Real (that is, $\in \mathbb{R}$) symmetrical matrix has all real eigenvalues

Eigenvalues and Eigenvectors

- `numpy.linalg.eig(A)` returns 2 vectors **d** and **V**:
  - **d** = vector with all of A's eigenvalues
  - **V** = the eigenvector

- `numpy.linalg.eigh(A)` does the same thing *only faster*
  - Uses a different algorithm that `.eig()`
  - BUT should only be used if you know **A** is symmetrical
  - FYI: the h is for "Hermitian"

Eigenvalues and Eigenvectors

- For a *n-by-n* square matrix: $(\mathbf{A} - \lambda\mathrm{I})\boldsymbol{x} = 0$
- It implies that $(\mathbf{A} - \lambda\mathrm{I})$ is singular and so: $\det(\mathbf{A} - \lambda\mathrm{I}) = 0$
- This is mainly what we use to solve for λ and $\mathbf{x}$

- Let's consider the matrix Λ:

  the *n-by-n* diagonal matrix with λ_j as elements

- ..and also consider the matrix $\mathbf{X}$:

  the n-by-n set of corresponding $\mathbf{x_j}$ for each λ_j

Eigenvalues and Eigenvectors

Continued…

- Therefore: $AX = X\Lambda$
  - That is, each column of X is multiplied by it's corresponding λ

- If we multiply both sides by $\mathbf{X^{-1}}$ (if it exists), then:

$$AXX^{-1} = X\Lambda X^{-1} \quad \ldots or \ldots \quad \underline{A = X\Lambda X^{-1}}$$

- *Yet another factorization technique!*

Quick! To the Python-mobile!

Your To-Dos

- **Homework 5 due next week WEDNESDAY**
- Remember: next week – lecture on Tuesday, but NOT on Thursday

For next week, read:

- **NCM book sections:**
  - Chapter 10.1 and 10.2
  - Chapter 2.11
- **PDF found on today's Lecture Files site**
  - "The $25,000,000,000 Eigenvector"

</LECTURE>