Mon 1/23 numerical computation

Computers can only do basic computations

- Mult = shift and addition
- Div = shift and subtraction
- Add = like exclusive xor

Abs value

- Write my own

More terms in a taylor series = better approx.

Twos complement is fast

Pade approximants

Sqrt has no taylor series

Binary search

Floating point number difference < Epsilon instead of equality

Invert the square root funct with newton iterate

Scaling in and out of square root and log

Round-off errors with any floating point arithmetic

- 5 standard rounding modes
- Adding and subtracting with diff magnitudes, the smaller number can just be rounded away
- Precision error from subtracting diff magnitudes

Absolute (x' - x) and relative error (x' - x) / x

Floating points are not real nums

Wed 1/25

Floating point arithmetic

- Computer arithmetic: numbers are approx.
- Rounding errors

Round-off errors

- Rounding because it has to fit values into finite representation
- 5 types defined by IEEE 754
    - X
    - X
    - X
    - X
    - To zero, truncation

Relative error

- |x' - x| / |x|

Arrays and strings

Array

- Collection of same type elements
- Vector, matrix (array of vectors)
- More dimensions: tensor
    - Arrays of arrays

- Ordered [i]

Matrices
- M[x][y]
- Row major order

It's the algorithm that matters!! Can make things faster
The name of an array is a pointer to the element 0 (base address)
- It acts like pass by reference
- Don't want to copy on stack they can be too big

& address of
- Gives memory location of var

sizeof()
- Bytes used by var

Searching
- Traversing

Find item
- Return first index of the item

Ordering
- Ordering makes a search more efficient

Binary seach
- If ordered, this is the fastest you can search
- Middle? Go left or right

Friday 1/27

Graphing: add to print each term
Can always add more to code
Our results can and will be a little off from his numbers

<u>Strings</u>
Def: array of characters ending in \0
char s[] = "hello"
char *s = "hello"
char s[100] 99 char printable (the last one is \0)
strcmp() would compare the chars by their ascii value
strlen() loop with counter until end of array
strcpy() have two char arrays and copy each char

<u>Pointers</u>
Set to NULL not nothing/zero
Pointers point to assigned address
& means "address of"
* "the thing the pointer points to", dereference pointer

Pros
- Passing large data structures

- Instead of a copy, it is a pointer
- Dynamic data structs

Arithmetic

- ++ and - - increment and decrment to the next or previous pointer, increment by bytes based on the size of the pointer (int uint64_t etc)
- + addition is scaled by size of thing in pointer
- - finds diff between the addresses
- Can be compared with < > =
- str[1] same as *(str + 1)
- Array of var type starts at 1000, arr[i] is 1000+(i * sizeof(var type))
- Math that makes sense: adding int to pointer, subtract int from pointer, subtracting two pointers

Function pointers

- Points to executable code
- Dereferecing it yields the function

Ex.

Funct called increment

void (*f_ptr)(int *) = increment

f_ptr(&x); ← this will invoke increment funct

Function/Jump tables

- Array of pointers to functions
- Depending on what index of the array you pick, it will run that function