nmi

2024 spring : lecture 01 : basics : polynomials, fps

numerical methods

as distinguished from other branches of numerical methods and computer science,

1) work with arbitrary real numbers (including rational **approximations** of irrational numbers) and
2) consider **cost** and
3) consider **accuracy**.

this class will provide another way to express, to extend your math.

polynomials

The most fundamental operations of arithmetic are ***addition*** *and* ***multiplication****. These are also the operations needed to evaluate a polynomial p(x) at a particular value x. It is no coincidence that polynomials are the basic building blocks for many computational techniques we will construct.*

evaluation

1) **approximate** p(x) at x while
2) minimizing **operations** and
3) maximizing **accuracy**.

$p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$

method 1, step individually:

$a_4*x*x*x*x + a_3*x*x*x + a_2*x*x + a_1*x + a_0 \sim 14$ ops.

method 2, cache and reuse:

$x_2 = x*x,\ x_3 = x_2*x,\ x_4 = x_3*x \sim 3$ ops;
$p_4 = a_4*x_4,\ p_3 = a_3*x_3,\ p_2 = a_2*x_2,\ p_1=a_1*x \sim 4$ ops;
$p(x) = p_4 + p_3 + p_2 + p_1 + a_0 \sim 4$ ops ~ 11 ops total.

method 3, nest multiplication, horners:

$p(x) = (((a_4*x+a_3)*x+a_2)*x+a_1)*x+a_0 \sim 8$ ops.

binary notation; conversion between decimal

binary notation: $\ldots b_2 b_1 b_0 . b_{-1} b_{-2} \ldots$

conversion to decimal.

$\ldots b_2 * 2^2 + b_1 * 2^1 + b_0 * 2^0 + b_{-1} * 2^{-1} + b_{-2} * 2^{-2} \ldots$

eg, 111.11_2

integer ~ $1*2^2+1*2^1+1*2^0 = 7$.

fractional ~ $1*2^{-1} + 1*2^{-2} = ½ + ¼ = ¾$.

$\Rightarrow 111.11_2 = 7_{10} + ¾_{10} = 7.75_{10}$.

decimal conversion to binary.

eg, 111.25_{10}.

integer ~ 111/2 = 55 R 1 ⇒ 55/2 = 27 R 1 ⇒ 27/2 = 13 R 1 ⇒ 13/2 = 6 R 1 ⇒ 6/2 = 3 R 0 ⇒ 3/2 = 1 R 1 ⇒ 1/2 = 0 R 1 ⇒ 1101111, remainders in reverse order.

fractional ~ 0.25*2 = 0.50 + 0 ⇒ 0.50*2 = 0.00 + 1 ⇒ 0.01, integers in order from left to right.

$\Rightarrow 111.25_{10} = 1101111_2 + 0.01_2 = 1101111.01_2$.

polynomials in the machine

digital representation

$x = [d_{N-1},\ldots,d_1,d_0]$, digital vector

$x = d_{N-1}*b^{N-1} + \ldots + d_1b^1 + d_0*b^0$,

with **precision N** and **base b**.

eg,

base 10: $500_{10} = [5,0,0]$; $[5] = 5_{10}$.

base 02: $[1,0,1] = 101_2 = 1*2^2 + 0*2^1 + 1*2^0$.

fixed/positional representation

using the previous example,

base 02: $101_2 = 1*2^2 + 0*2^1 + 1*2^0$,

where the right hand side of the equality is the **fixed representation** and the left hand subscript is the base or radix r. additionally precision $N \geq 1$ and $r \geq 2$ such that

$x = \sum^N d_k r^k$ has r^N permutations and can also be written as

$r^N = (r-1)(r^{N-1})+(r^{N-1}) = [r-1]_{N-1}[r]_{N-2}\ldots[r]_1[r]_0[r]_{-1}[r]_{-2}\ldots[r]_{-(N-2)}[r]_{-(N-1)}$,

where subscripts denote position wrt exponent.

eg, $N = 3$, $r = 2$.

permutations, $r^N = 2^3 \Rightarrow \{000,001,010,011,100,101,110,111\}$;
magnitude, $\sum^{N-1} d_k r^k \leq \sum^{N-1} (r-1)r^k = r^N - 1 \Rightarrow$ range $[0,r^N-1]$.

sign

note: standard bias = r^{N-1} -1. refer to previous slide for more information.

next transition is slow

method 1) use position d_{N-1} for sign,

$x = [\pm][d_{N-2},\ldots,d_1,d_0]$.

permutations = $r^{N-1}*2$; range = $[-r^{N-1}+1,0), [0,+r^{N-1}-1]$.

method 2) use bias to obtain sign.

all positions used for magnitude and bias is an operation.

ie, $x_{min} = -B$, $x_{max} = r^N - B \Rightarrow$ range $[1 - r^{N-1}, r^{N-1}(r-1)]$ with B as **standard bias** $r^{N-1} - 1$.

eg, N = 3, r = 2, and standard bias.

$B = r^{N-1} - 1 = 2^{3-1} - 1 = 3 \Rightarrow [000,111]_2 \Rightarrow [0,7]_{10} - B = [-3,+4]_{10}$.

floating point

IEEE 754, $\mathbb{F}(N-1,m,r,b) = \mathbb{F}(64,53,2,2)$.

note: 32-bit is single precision; 64-bit is double precision.

next transition is slow

$x = M.b^E$, where **mantissa M** is an integer represented by sign/magnitude, radix and **precision m** and **exponent E** is an integer represented by bias and same radix. also, M is normalized as 1.F, where **fractional F**

$F = \sum^{m-2} d_k r^{-k}$, $r \geq 2 \Rightarrow x = \pm 1.F\ b^E$.

eg, $\mathbb{F}(N=5,m=3,r=3,b=2)$ with standard bias.

ie, same r for M and E; m includes sign; $m_E = N - m$; and $B = r^{N-1}-1$ with bias power $N - 1 = m_E - 1$. note: b is the base of the exponent not the base of the exponents power.

$x = \pm 1.F *2^E = [s][e_1][e_0]1.[f_1][f_2]$,

where $s \in \{0,1\}$, $m_E = 2$, $e_i \in \{0,1,2\}$, $f_j \in \{0,1,2\}$ and $B = 3^{(5-3)-1} - 1 = 2$. the range of $F = [00,22]_3$ and range of $E = [00,22]_3 - B = [0,8]_{10} - B = [-2,6]_{10}$.

ie, $x = [0,1,1,2,0]_{\mathbb{F}(5,3,3,2)} = (-1)^0 * 1.20_3 * 2^E$,

where $E = (11_3 - B) = (4 - 2)_{10} = 2_{10} \Rightarrow x_{10} = 1.66... *2^2 = 6.66...$

floating point

next transition is slow

eg, $\mathbb{F}$(N=6,m=4,r=3,b=2) with standard bias.

*again: same r for M and E; m includes sign; m_E = N - m; and B = $r^{\mathbf{N-1}}$-1 with **bias power N - 1= m_E - 1**. note: b is the base of the exponent not the base of the exponents magnitude.*

$x = \pm 1.F * 2^E = [s][e_1][e_0]1.[f_1][f_2][f_3]$,

where s $\in \{0,1\}$, $m_E = 2$, $e_i \in \{0,1,2\}$, $f_j \in \{0,1,2\}$ and $B = 3^{(6-4)-1} - 1 = 2$. the range of F = $[000,222]_3$ and range of E = $[00,22]_3 - B = [0,8]_{10} - B = [-2,6]_{10}$. therefore,

$|x_{MIN}| = [0,0,0,0,0,0]_{\mathbb{F}(6,4,3,2)} = (-1)^0 * 1.000_3 * 2^E$, E = -2
$|x_{MAX}| = [0,2,2,2,2,2]_{\mathbb{F}(6,4,3,2)} = (-1)^0 * 1.222_3 * 2^E$, E = 6

$\Rightarrow |x_{MAX}| = (1. + 2*3^{-1} + 2*3^{-2} + 2*3^{-3})*2^6 = \ldots$

note: for this FPS, ± ¼ are the smallest magnitude numbers with $b^E = 2^{-2}$ and the largest magnitude numbers with $b^E = 2^6$.

floating point

normalized vs denormalized

next transition is slow

a base-2 floating point number will always start with "1", so its inclusion is implied. explicitly, $1x2^0$ is a given so the position it might have used is given over to the fractional part of the mantissa. that is the normalized mantissa.

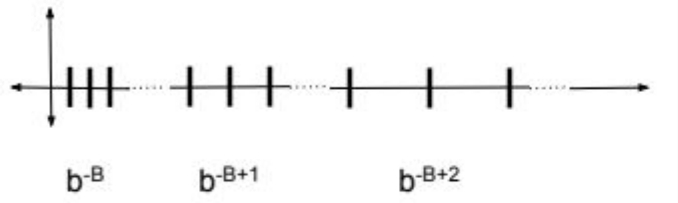
however, if the unbiased exponent is zero, the mantissa is denormalized. ie, there is no implicit "1".

eg*, [0][00000000]0.[00010...0] = $+(1*2^{-4})*(2*^{0-126}) = +2^{-130}$.

so the system definition has that trick for teeny-tiny numbers.

fps observations

- gaps between adjacent numbers scale with size. (ie, consider negative exponent vs positive exponent.)



- machine epsilon, ϵ_{mach}, is the gap between 1 and the next FPN.
- unit roundoff, $u_{mach} = ½ \epsilon_{mach}$.
- for all x, there exists floating point x' such that $|x-x'| \le u_{mach}*|x|$.
- when M normalized, zero represented by $\epsilon = \epsilon_{min} - 1$.
- $\pm\infty$ returned when an operation overflows.
- $x/\pm\infty$ returns 0 and $x/0$ returns $\pm\infty$.
- not a number (NaN) is returned if no well-defined finite or infinite result.

resources, lecture

additional resources:

horners method @wiki @youtube (general)
telescoping sum @wiki

floating point @wiki @youtube#1 #2-pta #2-ptb
unit in last place (ulp) @wiki
machine epsilon @wiki

ieee 754 @wiki

primary resources:

numerical analysis by tim sauer;

18.335j, introduction to numerical methods, mit ocw by steven g johnson; and

math 685, hunter college, spring 2023 with vincent martinez.

resources, python & system

googles FREE, jit crash course @coursera

beginners @python

fps @python

fps info @numpy

colab @google

latex/mathjax @colab

class-specific @github

peer chat @discord

help desk @hunter

next time

some error stuff

homework 01

due tuesday, february 6, noon

submit via blackboard

1. code conversion from decimal to binary.

note: check your work with pythons native conversion but you must code the algorithm.