Computer Simulations

A practical approach to simulation

Semra Gündüç

`gunduc@ankara.edu.tr`

Ankara University Faculty of Engineering,

Department of Computer Engineering

Number Representations

Representations of the Numbers

- Computers are consist of large but finite memory and processor which do calculations on the stored numbers.
- Memory is a sequence of electronic components which can only take two values namely bits (0 or 1). This means that all numbers are stored in memory in binary form.
- As a consequence, N bits can store integers in the range $[0, 2N]$.

Number Representations

- Storing arbitrary numbers on the memory require an optimization.
- Since the memory of the computer has limited space the use of this space require a decesion between,
    - 1) either each number will take as much spase as possible or
    - 2) as many numbers with limited precision (number of bits) as possible.

Number Representations

- The numbers must be long enough to give the required high precission of scientific calculations and short enough to store as many numbers as possible.
- To fullfill these requirements the structure of the computer memory is organized such that memory and storage sizes are measured in bytes, kilobytes, megabytes, gigabytes, terabytes, and petabytes (1015).

1 Bit	(binary integers) 0 and 1
1 byte	1B = 8 bits.
1K	1 kB = 1024 bytes.

Number Representations

- Some care is needed here for those who chose to compute sizes in detail because a K is not always a 1000.

Number Representations

Number Representations

- A problem in computer design is how to represent an arbitrary number using a finite amount of memory space, and then how to deal with the limitations arising from that representation.
- As a consequence of computer memories being based on the magnetic or electronic realization of a spin pointing up or down, the most elementary units of computer memory are the two bits .

Number Representations

-
-

Number Representations

- Consequently, binary strings are converted to octal, decimal, or hexadecimal numbers before results are communicated to people.
- Octal and hexadecimal numbers are nice because the conversion loses no precision, but not all that nice because our decimal rules of arithmetic do not work for them.
- Converting to decimal numbers make the numbers easier for us to work with, but unless the number is a power of two, the process leads to a decrease in precision.
- A description of a particular computer system will normally state the word length, that is, the number of bits used to store a number.

Number Representations

-
-

Number Representations

- In order to optimize the memory usage all computer languages define different size variables and constatnts.
- Variables are given name for storage allocation for a finite space in the memory with changing content. Variables and constants may have type, Boolean, Character, Integer, Float, ...
- Each type of variable have finite, predetermined size,which is called “Word size”

Number Representations

- At the beginning each computer brand used different size variable. This create confusion when results are compared. In 1987 IEEE standards for the number representations are accepted. Table 12 presents standard word sizes.

Boolean (True, False)	=	1 bit
Character (A,B,C,. . .)	=	1 byte
Single Precision Integer	=	4 bytes
Double Precision Integer	=	c bytes
Single Precision Float	=	4 bytes
Double Precision Float	=	8 bytes

Table 1: IEEE standard for the variable and constant definitions

Number Representations

Number Representations

Integer Representations

- Integers are typically 4 bytes (32 bits) in length and in the range $-2147483648 \leq 4\text{B} \;\; \text{integer} \leq 2147483647$.

  B : Base,

  E : Exponent,

  S : Sign,

  N : Word Length

  $$I = S \times (a_{N-1}B^{N-1} + a_{N-2}B^{N-2} + \quad \ldots \quad + a_0 B^0)$$

  where $0 \leq a_i < B$.

- Common bases are 10, 8, 16 or 2.

Number Representations

-
-

Number Representations

- In binary ($B = 2$) and Decimal ($B = 10$) notations largest and smallest single precision integers are:

$$
\begin{aligned}
I^{2}_{max} &= [0][1111111111111111111111111111111] = +2^{31} \\
I^{10}_{max} &= +2147483647 \\
I^{2}_{max} &= [1][1111111111111111111111111111111] = -2^{31} \\
I^{10}_{max} &= -2147483648
\end{aligned}
$$

Number Representations

Example: The same number in binary ($B = 2$), Octal ($B = 8$), Hexadecimal ($B = 16$) and Decimal ($B = 10$) notations.

$$
\begin{aligned}
I_{10} &= 3 \times 10^5 + 3 \times 10^4 + 7 \times 10^3 + 5 \times 10^2 \\
&\quad +5 \times 10^1 + 9 \times 10^0 \\
I_{16} &= 5 \times 16^4 + 2 \times 16^3 + 6 \times 16^2 + 9 \times 16^1 + 7 \times 16^0 \\
I_2 &= 1 \times 2^{18} + 0 \times 2^{17} + 1 \times 2^{16} + 0 \times 2^{15} \\
&\quad +0 \times 2^{14} + 1 \times 2^{13} + 0 \times 2^{12} + 0 \times 2^{11} + 1 \times 2^{10} \\
&\quad +1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 \\
&\quad +1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0
\end{aligned}
$$

Number Representations

-
-

Number Representations

Real Numbers

- Real numbers are represented on computers in either fixed-point or floating-point notation.
- Fixed-point notation can be used for numbers with a fixed number of places beyond the decimal point (radix) or for integers.
- It has the advantages of being able to use two's complement arithmetic and of storing integers exactly 3.

Number Representations

-
-

Number Representations

Fix Point Representation

- In the fixed-point representation with N bits and with two's complement format, a number is represented as

  $$Nfix = sign \times (a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_0 2^0 + \ldots + a_{-m} 2^{-m}),$$

  where $n + m = N - 2$.
- That is, one bit is used to store the sign, with the remaining $N - 1$ bits used to store the a_i values (the powers of 2 are understood).
- The particular values for N,m, and n are machine-dependent.

Number Representations

-
-

Number Representations

Floating Point Number Representation

- Most scientific computations use double–precision floating point numbers $(64b = 8B)$.
- The floating–point representation of numbers on computers is a binary version of what is commonly known as “scientific” or “engineering” notation.

Number Representations

- For example, the speed of light $c = +2.99792458 \times 10^{+8}$ m/s in scientific notation and $+0.299792458 \times 10^{+9}$ or $0.299795498\ E09$ m/s in engineering notation. In any of these cases, the number out front is called the mantissa and contains nine significant figures. The power to which 10 is raised is called the exponent, with the + sign included as a reminder that these numbers may be negative. Floating point numbers are stored on the computer as a concatenation.

Number Representations

-
-

Number Representations

- The two's complement of a binary number is the value obtained by subtracting the number from 2N for an N bit representation. Because this system represents negative numbers by the two's complement of the absolute value of the number, additions and subtractions can be made without the need to work with the sign of the number.

Number Representations

- Since any floating number must be represented in finite number of bits, the floating point system is finite and discrete.

  B Base.

  P Precision (Number of bits speared for the mantissa).

  L, U Exponent Range U:Upper,L:Lower.

Number Representations

-
-

Number Representations

- If E is the exponent the number x represented in base B sytem as,

$$x = (\frac{d_0}{B^0} + \frac{d_1}{B^1} + \frac{d_2}{B^2} + \ldots + \frac{d_{p-1}}{B^{p-1}})B^E$$

  where, $0 \leq d_i < B - 1$, $i = 0, \ldots, p - 1$ $L < E < U$.
  Mantissa : $d_0, d_1, \ldots, d_{p-1}$

- Sigh, exponent and mantissa are stored in seperate fixed-width fields of each floating word. Parameters for typical floating point system using IEEE standards for the exponents of the floting point representation:

	B	p	L	U
Single Precision	2	24	-126	127
Double Precision	2	53	-1022	1023

Number Representations

-
-

Number Representations

- Floating point system is normalized if leading digit is always nonzero unless number represented is zero. In a normalized system if manissa, $m \neq 0, 1 \leq m < B$.
- Not all real numbers x are exactly representable. If a real number is exactly representable in a given precision, this number is called *machine represantable floating point number*. If a real number x is not exactly representable, than it is approximated by $"nearby"$ machine represantable floating point number.

Number Representations

Total number of *normalized* floating numbers is;

$$2(B-1)B^{p-1}(U-L+1)+1$$

1. Smallest normalized pozitive number : B^L,
2. largest normalized number : $B^{U+1}(1-B^{-p})$.
3. This process is called *rounding* and error introduced by this process is called *rounding error*.

Number Representations

Chop : Truncate base B expansion of x after $(p-1)^{\mathrm{st}}$ digit

Round: x is rounded to the nearest representable floating point number. In case of tie, x is roundet to the nearest floating point number with even last digit.

Number Representations

Accuracy of floating point system is characterized by unit roundoff or machine precision.

(1)
$$\epsilon_{mach} = \begin{cases} B^{1-p} & \text{by chopping} \\ \frac{1}{2}B^{1-p} & \text{by rounding} \end{cases}$$

Units of roundoff ϵ_{mach} is determined by the number of digits in mantissa of floating point system.

Number Representations

Underflow: if a number is smaller than representable smallest number this stuation is called Underflow‘w.

Overflow: If a number is larger than the representable largest number thsi stuation is called Overflow.

Both Underflow and Overflow are related to the range of the exponent field.

Number Representations

(2)
$$\epsilon_{mach} = \begin{cases} 2^{-24} & \text{Single Precision} \\ 2^{-53} & \text{Double Precision} \end{cases}$$

Number Representations

Alternative definition of the machine precision can be given by choosing the smallest number, ϵ such that,

$$(1 + \epsilon) > 1$$

```
It= 1  Eps = 0.50000000     Val=1.5000000
It= 4  Eps = 6.25000000E-02 Val=1.0625000
It= 6  Eps = 1.56250000E-02 Val=1.0156250
It= 8  Eps = 3.90625000E-03 Val=1.0039063
It=10  Eps = 9.76562500E-04 Val=1.0009766
It=12  Eps = 2.44140625E-04 Val=1.0002441
It=14  Eps = 6.10351563E-05 Val=1.0000610
It=16  Eps = 1.52587891E-05 Val=1.0000153
It=18  Eps = 3.81469727E-06 Val=1.0000038
It=20  Eps = 9.53674316E-07 Val=1.0000010
It=22  Eps = 2.38418579E-07 Val=1.0000002
It=23  Eps = 1.19209290E-07 Val=1.0000001
```

Number Representations

floating Point Number System

- if a number smaller than representable smallest number Underflow
- If a number larger than the representable largest number, Overflow,
- Underflow and Overflow related to the range of the exponent field.

Number Representations

IEEE floating point standart provides special values to indicate two exceptional stuations.

Inf : Infinity results from dividing a finite number by zero, $1/0$.

NaN : Not a Number, stands for $0/0$, $0 \times \mathrm{Inf}$ or $\mathrm{Inf}/\mathrm{Inf}$.

Inf and NaN are implemented in IEEE arithmetic through special reserved values of exponent field.

Number Representations

Sources of Errors in Computation

Number Representations

General Strategy for Numerical Simulation and Modelling

Well-Posed Problems

Problem is well-posed if solution

1. exists
2. is unique
3. depends continuously on problem data

Otherwise, problem is ill-posed Even if problem is well posed, solution may still be sensitive to input data. Computational algorithm should not make sensitivity worse.

Number Representations

How to Model a Real World Problem Replace difficult problem by easier one having same or closely related solution

1. infinite $\rightarrow$ finite
2. differential $\rightarrow$ algebraic
3. nonlinear $\rightarrow$ linear
4. complicated $\rightarrow$ simple

Solution obtained may only approximate that of original problem.

Number Representations

Sources of Error in Moddeling Approximation

Before computation

1. modeling
2. empirical measurements
3. previous computations

During computation

1. truncation or discretization
2. rounding

Number Representations

Accuracy of final result reflects all these uncertainty in input may be amplified by problem perturbations during computation may be amplified by algorithm.

Number Representations

Example:

Problem:

Computing surface area of Earth using formula $A = 4\pi r^2$

Approximations:

- Earth is modeled as a sphere.
- Value for radius is approximate.
- Value for π requires truncation.
- Values for input data and results of arithmetic operations are rounded in computer.

Number Representations

Absolute Error and Relative Error

$$\textbf{Absolute error} = (\text{approximate} - \text{true})\text{ value}$$

$$\textbf{Relative error} = \frac{\text{absolute error}}{\text{true value}}$$

$$\text{Approx value} \;=\; (\text{true value}) \times (1 + \text{relative error})$$

True value usually unknown, so we estimate or bound error rather than compute it exactly

Relative error often taken relative to approximate value, rather than (unknown) true value

Number Representations

Data Error and Computational Error Typical problem: compute value of function $f : R \to R$ for given argument $x =$ true value of input $f(x) =$ desired result $\hat{x} =$ approximate (inexact) input $\hat{f} =$ approximate function; actually computed

Total error :

$$\begin{aligned} \hat{f}(\hat{x}) - f(x) &= \\ \hat{f}(\hat{x}) - f(\hat{x}) &+ f(\hat{x}) - f(x) \\ \text{computational error} &+ \text{propagated data error} \end{aligned}$$

Algorithm has no effect on propagated data error

Number Representations

Computational Error

Truncation Error : Difference between true result (for actual input)and result produced by given algorithm using exact rithmetic.Truncation of infinite series.

Rounding Error : Difference between results produced by given algorithm and result produced by the same algorithm using limited precision aritmetic. Computational error is the sum of Rounding error and truncation error.

Number Representations

Calculation of the machine precision

```
int main(){
  float eps,sum;
  int i,n;
  n = 25;
  printf("Calculate machine precision 1+eps~1\n");
  eps = 1.0;
  for(i = 1;i<=n;i++){
    eps = eps/2.0d0;
    sum = 1.0 + eps;
    printf("Iter=%2d Eps=%12.8f \
     value=%12.8f\n",i,eps,sum);
  }
}
```

Number Representations

Calculation of the machine precision

```
Calculate machine precision 1+eps ~ 1; eps = eps/2
Iter =  1 Eps =0.50000000 value =1.50000000
Iter =  2 Eps =0.25000000 value =1.25000000
Iter =  3 Eps =0.12500000 value =1.12500000
Iter =  4 Eps =0.06250000 value =1.06250000
Iter =  5 Eps =0.03125000 value =1.03125000
Iter =  6 Eps =0.01562500 value =1.01562500
Iter =  7 Eps =0.00781250 value =1.00781250
Iter =  8 Eps =0.00390625 value =1.00390625
Iter =  9 Eps =0.00195312 value =1.00195312
Iter = 10 Eps =0.00097656 value =1.00097656
Iter = 11 Eps =0.00048828 value =1.00048828
Iter = 12 Eps =0.00024414 value =1.00024414
```

Number Representations

Calculation of the machine precision

```
Iter = 13 Eps =0.00012207 value =1.00012207
Iter = 14 Eps =0.00006104 value =1.00006104
Iter = 15 Eps =0.00003052 value =1.00003052
Iter = 16 Eps =0.00001526 value =1.00001526
Iter = 17 Eps =0.00000763 value =1.00000763
Iter = 18 Eps =0.00000381 value =1.00000381
Iter = 19 Eps =0.00000191 value =1.00000191
Iter = 20 Eps =0.00000095 value =1.00000095
Iter = 21 Eps =0.00000048 value =1.00000048
Iter = 22 Eps =0.00000024 value =1.00000024
Iter = 23 Eps =0.00000012 value =1.00000012
Iter = 24 Eps =0.00000006 value =1.00000000
Iter = 25 Eps =0.00000003 value =1.00000000
```

Number Representations

Computational Error

Calculate exponantial function,

$$f(x) = \exp(x)$$

$$\hat{y} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

```
int main(){
  float x,x2,x3,x4,sum;
  x = -0.5;
  printf("Calculate exp(-0.5) by using Taylor serie
  sum = 1.0+x+(x2=(x*x/2.0))+(x3=(x2*x/3.0))+(x4=(x
   printf("value= %f err= %+f \n",sum,exp(-0.5)-sum
  return(0);
}
```

Number Representations

Series Summation

```
int main(){
  float x,xx,sum;
  int   i,n=7;
  x = -0.5;
  printf("Calculate exp(-0.5) by using Taylor serie
  sum = 1.0;
  xx = 1.0;
  for(i = 1;i<=n;i++){
    xx = xx * x/(float)i;
    sum = sum + xx;
   printf("Iter=%d value=%f err=%+f \n",\
   i,sum,exp(-0.5)-sum);
  }
  return(0);
}
```

Number Representations

bf Calculate exp(0.5) by using Taylor series Expansion

```
It=1 Val=1.5000000 err= 0.14872122
It=2 Val=1.6250000 err= 2.37212181E-02
It=3 Val=1.6458334 err= 2.88784504E-03
It=4 Val=1.6484375 err= 2.83718109E-04
It=5 Val=1.6486980 err= 2.32458115E-05
It=6 Val=1.6487197 err= 1.54972076E-06
It=7 Val=1.6487212 err=0.0000000
```

Number Representations

Calculate exp(-0.5) by using Taylor series

```
It=1 Val=0.50000000 err= 0.10653067
It=2 Val=0.62500000 err=-1.84693336E-02
It=3 Val=0.60416669 err= 2.36397982E-03
It=4 Val=0.60677087 err=-2.40206718E-04
It=5 Val=0.60651046 err= 2.02059746E-05
It=6 Val=0.60653216 err=-1.49011612E-06
It=7 Val=0.60653061 err= 5.96046448E-08
```

Number Representations

Truncation Error

Number Representations

Example : Calculate cosine function by truncated series expansion

If only two terms are taken,

$$\begin{aligned} \text{absolute} \quad \text{error} &= 1 - \frac{x^2}{2!} - \cos(x) \\ &\sim \mathcal{O}(\frac{x^4}{4!}) \end{aligned}$$

(3)

Truncation error can be reduced by increasing the number of terms included in the expansion until the machine precision is reached.

Number Representations

Example : Calculate cosine function by truncated series expansion

For $x = 1.0$,

$$
\begin{aligned}
f(1.0) &= \cos(1.0) = 0.5403 \\
\hat{f}(1.0) &= 1 - \frac{x^2}{2!} = 0.5 \\
\hat{f}(1.0) &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \mathcal{O}(\frac{x^6}{6!}) \\
\hat{f}(1.0) &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \mathcal{O}(\frac{x^8}{8!})
\end{aligned}
$$

(4)

Number Representations

Errors in Computation

Number Representations

Error in finite difference approximation:

The derivative of a function at $x = x_i$, Taylor Series expansion of the function $f(x)$,

$$f(x_i + \Delta x) = f(x_i) + \left.\frac{df}{dx}\right|_{x=x_i} \Delta x + \frac{1}{2}\left.\frac{d^2 f}{dx^2}\right|_{x=x_i} \Delta x^2 + \cdots$$

leads to first derivatibe at $x = x_i$

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_i} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} + \mathcal{O}(\Delta x)$$

Number Representations

Errors in Computation

The first term not included in the calculations is, $\frac{1}{2!}\frac{\partial^2 f}{\partial x^2}\Big|_{x=x_i}$.

If M bounds $\frac{\partial^2 f}{\partial x^2}\Big|_{x=x_i}$ The truncation error is bounded by

$$\frac{M\Delta x}{2}$$

Rounding error bounded by $2\epsilon/\Delta x$ where ϵ is the error in function values.

$$\text{Total error} = \frac{M\Delta x}{2} + \frac{2\epsilon}{\Delta x}$$

Total error is minimized when $\Delta x \sim 2\sqrt{\epsilon/M}$.

Calculate $y = f(x)$ where $f : \mathbb{R} \to \mathbb{R}$

$$\text{Forward error} = \Delta y = \hat{y} - y \quad \text{where} \hat{y} = f(\hat{x})$$

Number Representations

Example : $f(x) = \sqrt{x}$
Problem is *insensitive* or *well conditioned* if relative change in the input causes similar relative change in the output.
Problem is *sensitive* od *ill conditioned* if relative change in solution can be much larger than the input data.

Number Representations

$$\text{Condition number} = \frac{\text{Relative change in solution}}{\text{Relative change in input data}}$$

$$\text{Condition number} = \frac{||[f(\hat{x}) - f(x)]||\,/f(x)}{||[(\hat{x} - x)/x]||}$$

Problem is sensitive or ill conditioned if $\text{Condition number} >> 1$. Condition number is an amplification factor.

$$\text{Relative forwarderror} = \text{Condition number} \times \text{Relative backward erro}$$

Number Representations

Stable Algorithm: Algorithm is stable if results are relatively insensitive to perturbations dring calculations. For stable algorithm, effects of computational error is no worse than effects of small data errorin input.

Accuracy : Closeness of computed solution to true solution of the problem.

Precision: The number of digits that is trustable.

Number Representations

Stability of algorithm does not guarantie accurate results. Accuracy depends on conditioning of the problem as well as stability of the algoritm. Inaccuracy can esult from aplying stable algorithm to ill-conditioned problems or unstable algorithm to well-conditioned problem.

Number Representations

Example : Tangent is sensitive to its arguments near $\Pi/2$.

$$\begin{aligned} \tan(1.57079) &\sim 1.58058\,10^5 \\ \tan(1.57078) &\sim 6.12490\,10^4 \\ \Delta x = 0.00001 & \quad \Delta f = 9.6809\,10^4 \end{aligned}$$