

Monte Carlo Methods: The Good and the Bad

- What are some *advantages* of MC methods?
- What are some *disadvantages* of MC methods?

- general
- generate predictions if nothing else will
- generalizes easily to many dimensions

- Error corr. ($O(\frac{1}{\sqrt{n}})$) is very slow compared to poly. based methods
- Outcome is nondet.

corrected

Computers and Random Numbers

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
             // guaranteed to be random.  
}
```

[from xkcd]

- How can a computer make random numbers?

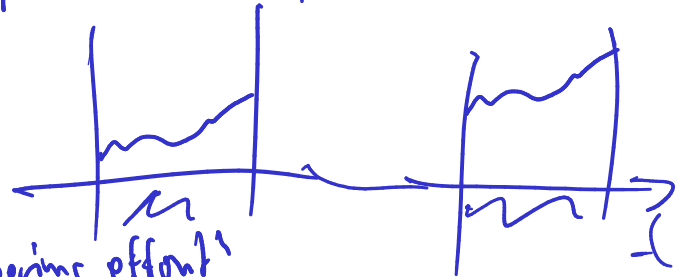
- Gather "randomness" from the environment
- HW RNG
- Kernel /dev/random (blocks)
/dev/urandom (does not block)

Random Numbers: What do we want?

- What properties can 'random numbers' have?


$$E(x) = \frac{1}{n} \sum x_i$$

- Have a specific distribution
- Real-valued/integer valued
- Long "period"
- Uncorrelated with later parts of the sequence
- Repeatable
- Un-predictable
 - "I have no idea"
 - "No amount of engineering effort"
- Usable on a parallel computer



What's a Pseudorandom Number?

- Actual randomness seems like a lot of work. How about 'pseudorandom numbers?'

$$\text{random \#}, \text{new state} = f(\text{state}) \leftarrow$$


The diagram shows a vertical loop of two arrows pointing in opposite directions, representing a cycle. To the right of this loop are five vertical arrows pointing downwards, each followed by the number '1', representing a sequence of operations or states.

- not easy to parallelize
- distributed right
- reproducible

Demo: Playing around with Random Number Generators

Some Pseudorandom Number Generators

Lots of variants of this idea:

- LC: 'Linear congruential' generators ←
- MT: 'Mersenne twister' ←

Remarks:

- Initial state and parameter choice often surprisingly tricky.
Bad choice: Predictable/correlated numbers.
E.g. Debian OpenSSL RNG disaster
- Absolutely **no reason** to use LC or MT any more. (Although almost all random number generators you're likely to find are based on those—Python's `random` module, `numpy.random`, C's `rand()`, C's `rand48()`).
- These are **obsolete**.

Counter-Based Random Number Generation (CBRNG)

- What's a CBRNG?

$$r_n = \text{encryption}(\text{counter}, \text{secret})$$

Demo: Counter-Based Random Number Generation

4 Error, Accuracy and Convergence

Error in Numerical Methods

- Every result we compute in Numerical Methods is inaccurate. What is our model of that error?

$$\text{Result} = \text{true answer} + \text{error}$$

- Suppose the true answer to a given problem is x_0 , and the computed answer is \tilde{x} . What is the **absolute error**?

$$\text{Absolute error} = | \text{true answer} - \text{result} |$$

- What is the **relative error**?

$$\text{relative error} = \frac{\text{Absolute error}}{|\text{result}|}$$

- Why introduce relative error?

"abs. error of 0.1"

result: 10000.1 $\rightarrow \frac{0.1}{10000} = 10^{-5}$

result: 0.00001 $\rightarrow \frac{0.1}{0.00001} = 10^3$

- What is meant by 'the result has 5 accurate digits'?

Because it tells me about the quality of my result



$$\rightarrow 3.1412777$$

"5 acc. digits"

$$\tilde{x} = \underline{12345}9999$$

$$\rightarrow 3.14159$$

$$x = 12345678.$$

$$\frac{|x - \tilde{x}|}{|x|} \approx 10^{-5}$$

What is "4 accurate digits" in terms of rel. error?

$$\underbrace{\frac{|x - \tilde{x}|}{|x|}}_{\text{rel. error}} \leq 10^{-4}$$

$$|\tilde{x} - x_0| \leftarrow \text{abs. error}$$

Measuring Error

- Why is ~~$|\tilde{x}| - |x_0|$~~ **wrong** and a **terrible** measure of the error?

$$\tilde{x} = -10,000 \quad x_0 = 10,000$$

- If \tilde{x} and x_0 are vectors, how do we measure the error?

$$x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\tilde{x} = \begin{pmatrix} 1.1 \\ 1.9 \end{pmatrix}$$

$$\leadsto x - \tilde{x} = \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix}$$

use norm
instead of abs.
value

\leadsto

$$\|x - \tilde{x}\|_1 = |0.1| + |-0.1| = 0.2$$


$$\|x - \tilde{x}\|_2 = \sqrt{0.1^2 + (-0.1)^2}$$

Sources of Error

- What are the main sources of error in numerical computation?

- truncation error \rightarrow associated with "imperfect" models (e.g. polynomials of finite degree)

- rounding error \rightarrow error associated with representing numbers in a computer

3.14159 200 

Digits and Rounding

- Establish a relationship between 'accurate digits' and rounding error.

I have 5 accurate digits.

$$\rightarrow \text{rel. error} = 10^{-5}$$

Condition Numbers

$$f(x) = x \quad \text{(cond. number: } 1)$$

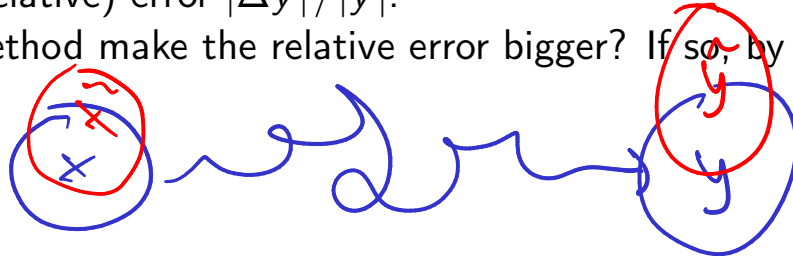
$$f(x) = 5 \quad \text{(cond. number: } 0)$$

- Methods f take input x and produce output $y = f(x)$.

Input has (relative) error $|\Delta x|/|x|$.

Output has (relative) error $|\Delta y|/|y|$.

Q: Did the method make the relative error bigger? If so, by how much?



$$\text{Rel. error in the output} \leq \kappa \cdot \text{Rel error in the input}$$

0.1

10 - 0.01

Condition number: "the factor by which (at most) the rel. error gets worse"

***n*th-Order Accuracy**

Often, truncation error is controlled by a parameter h .

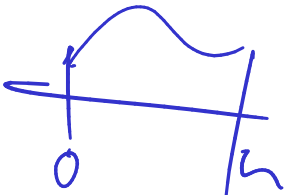
Examples:

- distance from expansion center in Taylor expansions
- length of the interval in interpolation

A numerical method is called '***n*th-order accurate**' if its truncation error $E(h)$ obeys

$$E(h) = O(h^n).$$

Interpolation with 5 points: 5, h^5 order accurate
 $a + bx + cx^2 + dx^3 + ex^4$



$$\text{error} = \underline{\underline{O(h^5)}}$$



5 Floating Point

Wanted: Real Numbers... in a computer

- Computers can represent *integers*, using bits:

$$23 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (10111)_2$$

How would we represent fractions, e.g. 23.625?