Random number generation

CE 391F

April 4, 2013

ANNOUNCEMENTS

- Homework 3 due today
- Homework 4 coming...

Webinar announcement

Femke van Wageningen-Kessels from TU Delft will be giving a webinar titled Traffic Flow Model Tree: A Genealogical Overview of the History of Traffic Flow Modeling"

Webinar is tomorrow at 10 Central time, sponsored by TRB Traffic Flow Committee.

REVIEW

Cellular automata vs. car following

How can we handle lane changing?

How can we incorporate randomness into our models?

OUTLINE

Generating a uniform(0,1) number

1. Middle-square: simple but flawed
2. "Super-random" generator: complex but flawed
3. Statistical tests
4. Linear congruential method: simple but (potentially) good

RANDOM NUMBER GENERATION

What does it mean to generate a random number?

Most computers produce *pseudorandom* numbers: they give the appearance of randomness, while being generated by a formula.

A few historical options for generating random numbers in scientific work...

- Roll dice, draw cards, cast lots...
- Draw balls from a "well-stirred urn"
- Table of 40,000 digits "taken at random from census reports"
- Atmospheric noise

Middle-square method

Let's say we want to generate a sequence of random two-digit numbers.

Begin by picking a *seed value* 1234

The first random number is the middle two digits: 23

Square 23, and pick the middle two digits as the next random number: $23^2 = 0\mathbf{52}9$

Square 52, and get $2\mathbf{70}4$.

So, the sequence begins 23, 52, 70, 90, and so forth.

Even though this sequence is completely deterministic, it gives an appearance of randomness.

Unfortunately, this simple method tends to get stuck in a loop:

23,52,70,90,10,10,10,10,...

Choosing a different seed gives a different sequence:
85,22,48,30,90,10,10,10,...
42,76,77,92,46,11,12,14,19,36,29,84,5,25,62,84,5,25,62,84,...

Researchers have developed much better ways of generating random numbers (and for quantifying how "random" a sequence appears.)

For now, we'll focus on generating a random real number from a uniform(0,1) distribution.

We can use this to simulate a wide variety of random processes. How can we use this to perform an action with probability p?

How can we convert the middle-square method into a uniform(0,1), approximately?

Perhaps the middle-square method is "too simple." What if we went to a really complex algorithm?

Don Knuth experimented with a "super random" algorithm, which executed different randomizing steps a random number of times, based on the digits of the previous ten-digit number X.

1. Let Y be the most significant digit of X. Repeat step 2 Y times.
2. Let Z be the second-most significant digit of X. Jump down $Z + 1$ steps.
3. If $X < 5000000000$, add 500000000 to X.
4. Middle-square X
5. Multiply X by 1001001001 and pick the ten least significant digits.
6. If $X < 100000000$, add 9814055677, otherwise subtract X from 1000000000
7. Swap the first five digits of X and the last five digits.
8. Repeat step 5.
9. Decrease every positive digit in X by one.
10. If $X < 99999$, square X and add 99999. Otherwise subtract 99999 from X.
11. Add zeros to the end of X until it is greater than 100000000.
12. Replace X by the middle ten digits of $X(X - 1)$.

Don't focus on the details, the point is, it's complicated and would seem to do an exceptionally good job of jumbling a number.

Perhaps the middle-square method is "too simple." What if we went to a really complex algorithm?

Don Knuth experimented with a "super random" algorithm, which executed different randomizing steps a random number of times, based on the digits of the previous ten-digit number X.

When he tried this, the program converged almost immediately to 6065038420, which is unchanged by the above algorithm!

Trying another starting value, after the 7401th iteration, the numbers fell into a cycle of length 3178.

In other words, exceptionally complex methods are not necessarily good. Furthermore, exceptionally complex methods are exceptionally difficult to analyze. With simpler methods, we may be able to actually prove that they work "well."

Stochastic desiderata

A pseudorandom $U(0,1)$ sequence would ideally pass the following tests:

- Frequency test (histograms with any bin width should show approximately equal frequency)
- Serial test (correlation should not be evident; equivalently the random number should not be easily predictable)
- Gap test (the sequence should not "avoid" particular intervals for long stretches)
- Poker test (bin data, check frequency of pairs, three-of-a-kind, full house, etc. to "true" $U(0,1)$ probabilities)
- Coupon collector test
- Run test
- Birthday spacings test

For more philosophical and mathematical discussions on what "randomness" actually means, see Knuth, *TAOCP*, volume 2.

Linear Congruential Method

The LCM is a simple algorithm that can actually be quite good according to our desiderata, and provably so. It uses the formula

$$X_{n+1} = (aX_n + c) \mod m$$

and requires four parameters: m, the modulus; a, the multiplier; c, the increment; and X_0, the starting value.

The notation $x \mod m$ means the remainder after dividing x by m. Therefore, $x \mod m$ will always be between 0 and $m - 1$, so $(x \mod m)/m$ can approximate a $U(0, 1)$.

Example: $m = 100$, $a = 13$, $c = 57$, $X_0 = 12$

The first step gives: $X_1 = (13 \times 12 + 57) \mod 100 = 13$

The next step gives $X_2 = (13 \times 13 + 57) \mod 100 = 26$

The next step gives $X_3 = (26 \times 13 + 57) \mod 100 = 95$

The next step gives $X_4 = (95 \times 13 + 57) \mod 100 = 92$

The next step gives $X_4 = (92 \times 13 + 57) \mod 100 = 53$

...and so on.

Not all choices of m, a, and c will lead to a "good" generator.

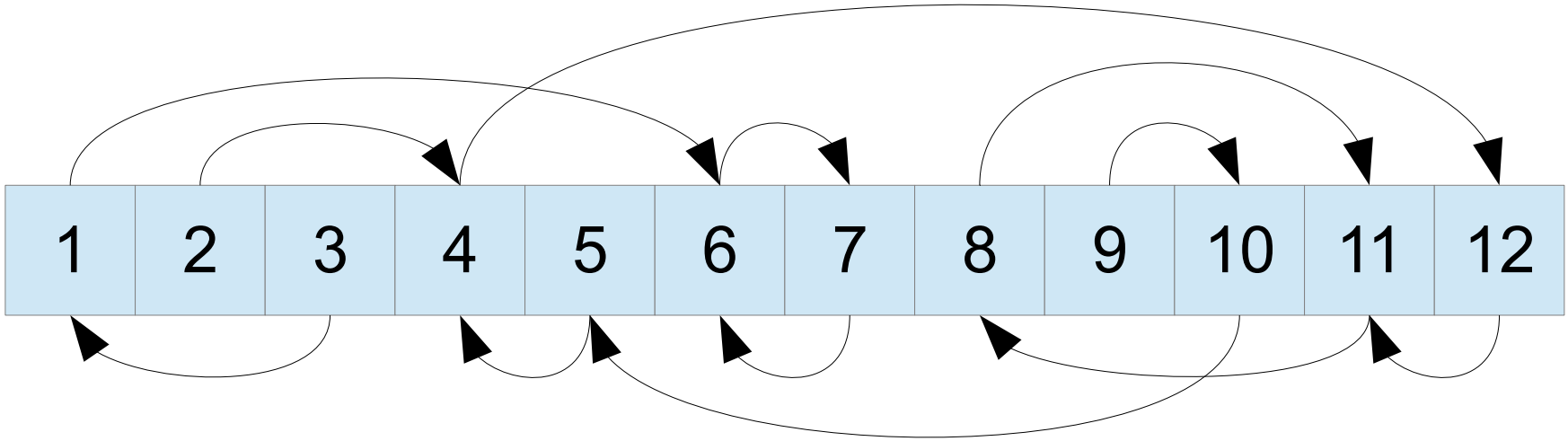
Obviously $a = 0$ or $a = 1$ are horrible choices (why?)

What would happen if $m = 2$?

Will the LCM eventually cycle?

Can we determine how long the cycle will be?

The LCM has no "memory", every number maps onto exactly one number between 0 and $m - 1$.



Therefore, the LCM must eventually cycle, and the period can be no longer than m.

Therefore, we usually want m to be fairly large.

Theorem. The linear congruential sequence has period length m iff:

- c and m are relatively prime
- $a - 1$ is a multiple of every prime dividing m
- If m is a multiple of 4, so is $a - 1$.

Proof. Requires some number theory.

Examples

$m = 18, a = 7, c = 5$ has full period:

$$0, 5, 4, 15, 2, 1, 12, 17, 16, 9, 14, 13, 6, 11, 10, 3, 8, 7, 0, \ldots$$

Violating the first condition, let $m = 18, a = 7, c = 6$:

$$0, 6, 12, 0, \ldots$$

Violating the second condition, let $m = 18, a = 6, c = 5$:

$$0, 5, 17, 17, \ldots$$

Violating the third condition, let $m = 16, a = 7, c = 5$

$$0, 5, 8, 13, 0, \ldots$$

Caution!

Period length is not the only relevant factor! $a = c = 1$ has full period, but is completely useless as a "random" number generator.

The other statistical tests should be used to ensure randomness. A good generator should pass *all* of those tests.

Passing any test does not mean the generator is good; but failing any test means it is bad.

Some values used in practice:

$m = 2^{32}, a = 1664525, c = 1013904223$

$m = 2^{32}, a = 22695477, c = 1$

$m = 2^{31}, a = 1103515245, c = 12345$

To turn this into a $U(0, 1)$ variate, divide by m

The system is governed by the following rules, all four of which are applied to each vehicle in the stated order:

- **Acceleration:** If the velocity v is less than v_{max}, and the distance to the next car ahead is greater than $v + 1$, the speed increases by 1.
- **Car-following:** If the distance to the next vehicle is j and $j \leq v$, the speed decreases to $j - 1$.
- **Randomization:** Generate a random number between 0 and 1. If less than p and $v > 0$, decrease v by 1.
- **Motion:** The vehicle advances v cells.

Note that if you are using the LCM, you just need to keep track of the previous number used to generate the next one. You do have to select a seed X_0 to begin with.

Other distributions

In traffic simulation, we are often concerned with other types of distributions in addition to the uniform. A typical example:

- Vehicles enter the facility according to a Poisson process (i.e., time between new vehicles is exponentially distributed)
- v_{max} for different vehicles follows a uniform distribution between 4 and 6.
- l for different vehicles follows a normal distribution with mean 10 and standard deviation 2.
- and so on...

If we were doing a car-following simulation, we may have distributions for T and λ, and each time a vehicle is generated, we need to pick an appropriate value.

PROBABILITY REVIEW

Review of basic concepts

A random variable X has the cumulative distribution function F_X if $P(X \leq x) = F_X(x)$ for all x.

Cumulative distribution functions are nonnegative, nondecreasing, tend to 0 as $x \to -\infty$ and to 1 as $x \to \infty$.

If F_X is continuous and differentiable almost everywhere, X is a continuous random variable, and its derivative f_X is the probability density function of X.

$$P(a < X < b) = \int_a^b f_X(x)\, dx = F_X(b) - F_X(a)$$

Some common distributions

The *uniform* distribution between A and B has the pdf $f_X(x) = 1/(B-A)$ if $x \in [A, B]$ and 0 otherwise, and the cdf $F_X(x) = (x-A)/(B-A)$ if $x \in [A, B]$, 0 if $x < A$, and 1 if $x > B$.

The *exponential* distribution with mean λ has the pdf $f_X(x) = \frac{1}{\lambda}\exp(-x/\lambda)$ if $x \geq 0$ and 0 otherwise, and cdf $F_X(x) = 1 - \exp(-x/\lambda)$ if $x \geq 0$ and 0 otherwise.

The *normal* distribution with mean mean μ and variance σ^2 has the pdf $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-(x-\mu)^2/2\sigma^2)$, and its cdf has no closed form.

So far, we've discussed how to generate random numbers from a uniform distribution with $A = 0$ and $B = 1$. How can we generate random numbers from the distributions on the previous slide?