3.14159265358979323846264338327950288419716939937510582097494459230781640628620899
8628034825342117067982148086513282306647093844609550582231725359408128481117450284
1027019385211055596446229489549303819644288109756659334461284756482337867831652712
0190914564856692346034861045432664821339360726024914127372458700660631558817488152
0920962829254091715364367892590360011330530548820466521384146951941511609433057270
3657595919530921861173819326117931051185480744623799627495673518857527248912279381
8301194912983367336244065664308602139494639522473719070217986094370277053921717629
3176752384674818467669405132000568127145263560827785771342757789609173637178721468
4409012249534301465495853710507922796892589235420199561121290219608640344181598136
2977477130996051870721134999999837297804995105973173281609631859502445945534690830
2642522308253344685035261931188171010003137838752886587533208381420617177669147303
5982534904287554687311595628638823537875937519577818577805321712268066130019278766
1119590921642019893809525720106548586327886593615338182796823030195203530185296899
5773622599413891249721775283479131515574857242454150695950829533116861727855889075
0983817546374649393192550604009277016711390098488240128583616035637076601047101819
429555961989467678374494482553797747268471040475346462080466842590694912...

Algorithms for decision support

Random number generators

Stochastic variables occur in simulation at different places:

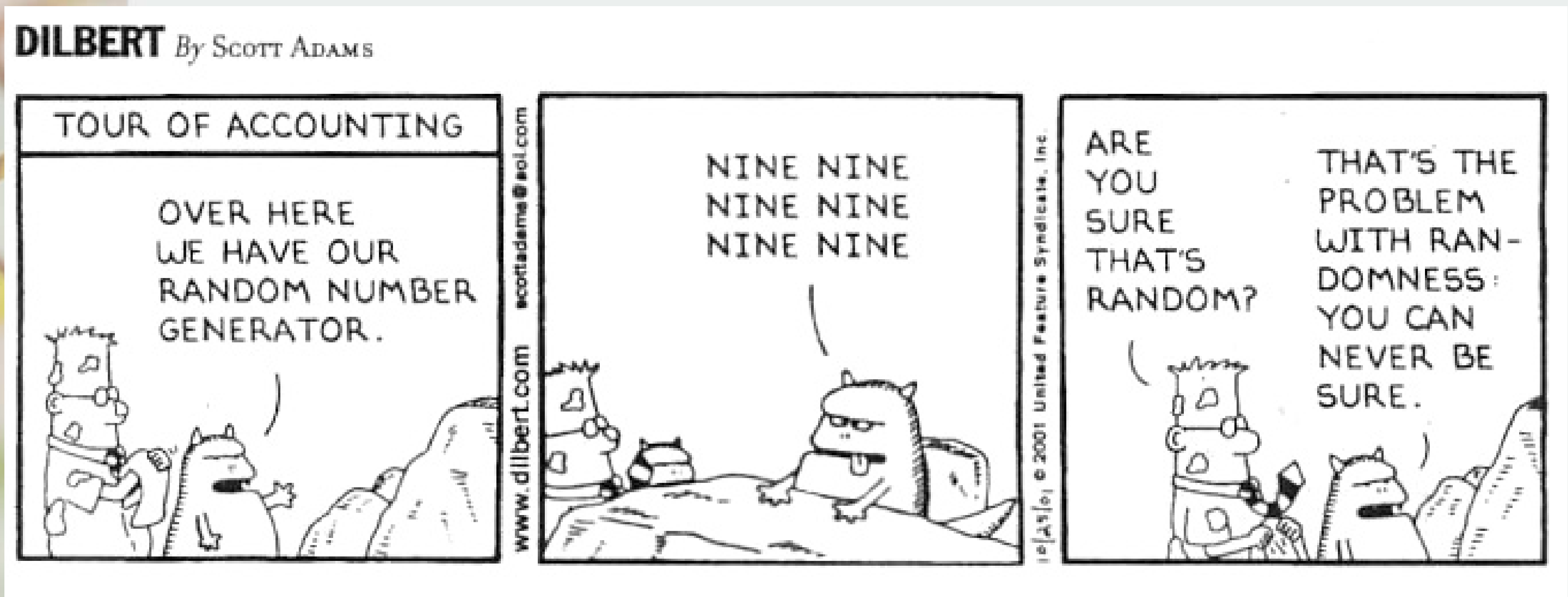
1. Input data are modeled as stochastic variables
   - E.g time until arrival of next customer
2. Generate random variables
   - When you schedule a new Arrival event you have to generate a random number for the time delay
3. Analysis of results

This lecture

Random number generators

1. Generate numbers U(0,1)
2. Use U(0,1) to generate other distributions

Desirable properties?



Desirable properties

- Numbers seems uniformly divided on [0;1]
- Numbers independent
- Sequence can repeat itself, but after a long while only
- Reproducible
- Separate streams
- Efficient (time and memory)

In **simulation** we use pseudo-random generators:

$$Z_i = f(Z_0, Z_1, \dots, Z_{i-1})$$

Results should be reproducible.

Midsquare method

- Z_0 4 digits (the number is $0.Z_0$)
- Z_0^2 8 digits
- Z_1 middle 4 digits of Z_0^2
- Etc.....

- Examples:
  - 1234, 01522756, 27321529
  - 1049, 1004, 0080, 0064, 0040, 0016, 0002, 0000….
  - 2100, 4100, 8100, 6100, 2100

- Disadvantages:
  - Can converge to zero
  - Repeats quickly

Linear congruential generators

- $Z_i = (a\ Z_{i-1}\ + c)\ MOD\ m$
- $U_i = Z_i / m$

a = multiplier

c = increment

m = modulus

If $c>0$: mixed generator

If $c=0$: multiplicative generator

Applied in Java

Linear congruential generators

- $Z_i = (Z_{i-1} + 5) \, mod \, 13$
  - $1,6,11,3,8,0,5,10,2,7,12,4,9,1,6 \ldots ..$
  - Period 13
- $Z_i = (2Z_{i-1} + 5) \, mod \, 13$
  - $8,8,8, \ldots$
  - $1,7,6,4,0,5,2,9,10,12,3,11,1, \ldots$
  - Period 12 or 1

Prime Modulus Multiplicative Linear Congruential Generator

- $c=0$, $m = p$ prime
- $Z_i = (a^i Z_0)$ MOD p
- Fermat's little theorem:
  - p prime, $1 \leq a<p$ then

$$a^{p-1} = 1 \ (mod\ p)$$

- *Let a* be a primitive element in $\mathbb{F}_p$

$$a^i = 1(\mathrm{mod}\, p) \quad \Leftrightarrow i = k(p-1) \text{ with } k \text{ integer}$$

- Then period $p-1$
- Example $p=7$, $a=3$ primitive element, $a=2$ not primitive

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Prime Modulus Multiplicative Linear Congruential Generator: Mersenne pimes

- Frequently used example $m = 2^{31}-1$
  (is one of the Mersenne primes, i.e. primes of the form $2^q - 1$)

Subtractive generator

- $Z_n = Z_{n-55} - Z_{n-24} \bmod 10^9$
- Seeds computed by:
  - Selection of s_0 from 0,1,2,.., 10^9 -1
  - $s_n = s_{n-2} - s_{n-1} \bmod 10^9$ for $n=1,2,\ldots,54$ $(s_{-1} = 0)$
  - *Reordering of these values*
  - Compute the next 165 values s_{55} to s_{219}. Store the last 55 values to start the sequences Z
- Applied in C#

Universiteit Utrecht

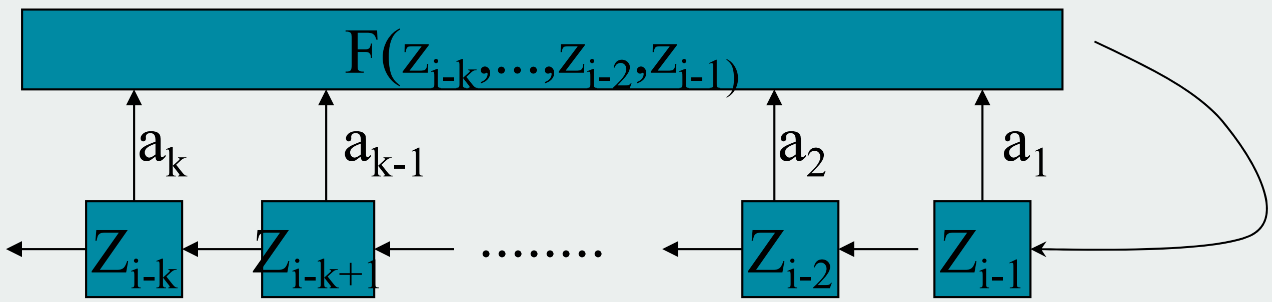
[Faculty of **Science**
Information and Computing Sciences]

Tausworthe generator
(linear feedback shift register)

- Generates bit sequences

$$z_i = (a_1 z_{i-1} + a_2 z_{i-2} + \ldots + a_k z_{i-k}) \bmod 2$$

$$u_n = \sum_{j=1}^{k} z_{(n-1)k+j} 2^{-j}$$



Tausworthe generator (shift register)

- Example

$$Z_i = (Z_{i-3} + Z_{i-4}) \bmod 2$$

$$1101\ 0111\ 1000\ 1001101\ldots$$

$$u_1 = \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{16} = \tfrac{13}{16}$$

$$u_2 = \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{16} = \tfrac{7}{16}$$

Tausworthe generator (shift register)

- Side note: shift register and pseudo-random bit sequences are important for cryptography

Cryptographic RNGs that e.g. use current computer time are not useful for simulation

Mersenne Twister MT 19937

- Designed for simulation
- Twisted generalized feedback shift register
- Period is Mersenne prime 2^p-1
- Generates uniform distribution on $[0,2^k-1]$
- k=32 version has period 2^{19937} -1
- Fast
- Passes statistical tests
- Default in R, Matlab, Python, Ruby, PHP

Newer developments

- WELL family of generators
  - Developed in 2006
  - Well Equidistributional Long-period Linear
  - based on linear recurrences modulo 2 over a finite binary field F_2
- Xorshift
  - Based on linear feedback shift registers
  - Includes taking the exclusive or of a number with a bit-shifted version of itself
- New developments are going on.

Inverse transform (continuous)

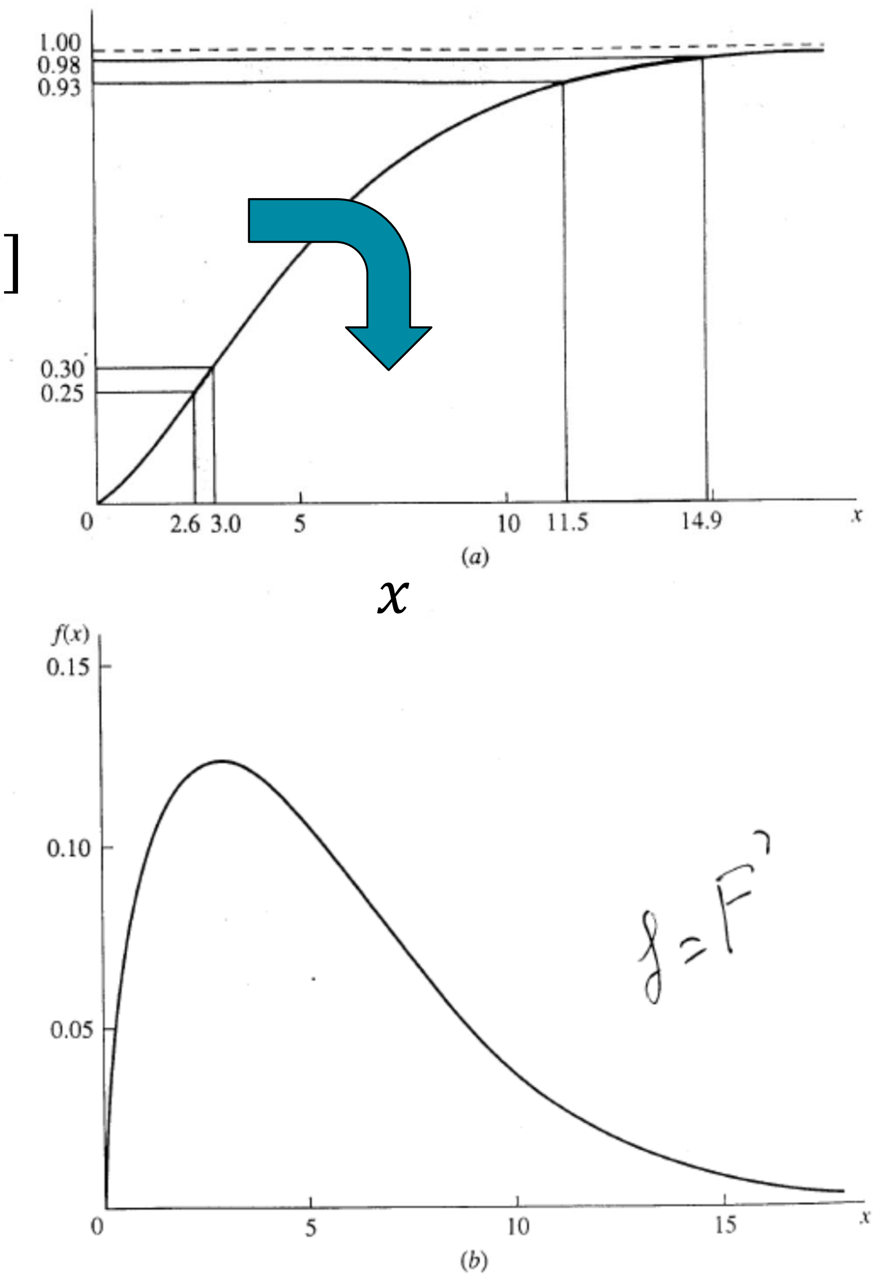
1. Generate u from U[0,1]
2. Return $x = F^{-1}(u)$

Here F is the probability distribution that you want to generate numbers from

$$P(X \leq x) = P(F^{-1}(u) \leq x) = P(u \leq F(x)) = F(x)$$

$u \in U[0;1]$

CDF: F



PDF: f

FIGURE 8.2
(*a*) Intervals for *U* and *X*, inverse transform for Weibull(1.5, 6) distribution;
(*b*) density for Weibull(1.5, 6) distribution.

Inverse transform: exponential distribution

$$F(x) = \begin{cases} 1 - e^{-\frac{x}{\beta}} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

1. Generate *u* from U[0,1]
2. Return

$$x = F^{-1}(u) = -\beta \ln(1-u)$$

Inverse transform (discrete)

1. Generate u from U[0,1]
2. Choose smallest i such that $u \leq F(x_i)$,

   where $x_1 < x_2 < \ldots$ are the possible realizations of the distribution
3. Return $x=x_i$

Example

- Numbers of prints after which ink refill is required

- 1998: 10%
- 1999: 20%
- 2000: 40%
- 2001: 20%
- 2002: 10%

- How to generate from U[0,1] with discrete inverse transform?

Composition

$$F(x)=\sum_{j=1}^{\infty} p_j F_j(x)$$

$$1.\ \text{Generate integer } J \text{ such that } P(J=j)=p_j$$

$$2.\ \text{Generate } X \text{ with distribution } F_J$$

Exercise:

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Composition: exercise

- For flight delays we know the following distribution:
  - 10% is too early, earlines follows a uniform distribution on [0,20] minutes
  - 90% is too late, delay follows an exponential distribution with average 20 minutes
- How to generate random delays without any software library?

Convolution

1. Generate $Y_1, Y_2, \ldots, Y_m$ each with distribution G
2. Return $X = Y_1 + Y_2 + \ldots + Y_m$

Example: k-Erlang

Wrap up

- After this lecture you know the basic algorithms of random number generation
- You are able to answer questions of the type given on slide 24.