Discrete Event Simulation

IN2045

Dr. Alexander Klein
Stephan Günther
Prof. Dr.-Ing. Georg Carle

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de
Topics

- Generation of Random Variables
  - Inversion, Composition, Convolution, Accept-Reject
- Distributions – Continuous
  - Uniform, Normal, Triangle, Lognormal
  - Exponential, Erlang-k, Gamma,
- Distributions - Discrete
  - Uniform(discrete), Bernoulli, Geom, Poisson, General Discrete
- Random Number Generator (RNG)
- Linear Congruential Generator (LCG)
- $X^2$ Test
- Serial Test
- Spectral Test
- Shift Register
- Generalized Feedback Shift Register
- Mersenne Twister

Chapter is based on LK 6+8)
Introduction - Random variates

- Generation of U(0,1) random numbers
  - Generation approaches

  - “Real”, “natural” random numbers: sampling from radioactive material or white noise from electronic circuits, throwing dice, drawing from an urn, ...
    - Problems:
      - If used online: not reproducible
      - Tables: uncomfortable, not enough samples
  - USB – Random Number Generator – Developed at TUM
    [Link to website](http://www.heise.de/newsticker/meldung/Appliance-liefert-50-Millionen-Zufallsbits-pro-Sekunde-1125288.html)

- Pseudo random numbers: recursive arithmetic formulas with a given starting value (seed)
  - in hardware: shift register with feedback (based on primitive polynomials as feedback patterns)
  - in software: linear congruential generator (LCG) (Lehmer, 1951), ...
Generating random variates

- All algorithms are based on U(0,1) random variates

- Selection criteria
  - Exactness (generation of the desired distribution)
  - Efficiency
    - Storage requirements (large tables required?)
    - Execution time
      - Marginal execution time (for each sample)
      - Setup time (at start time)
  - Robustness (characteristics do not change for different parameters)
  - Complexity (you have to understand before you implement it)

- Huge literature available
Random variates

- **Measurement**
  - Samples of a random variable $X$
  - What is the distribution function of random variable $X$?

- **Simulation**
  - Distribution function of the random variable is known in advance
  - How to generate samples which follow the distribution of the random variable?

- **Idea**
  - Generation of uniform distributed random numbers $U(0,1)$ (Random number generator)
  - Transformation of the generated numbers according to the desired distribution of the random variable
Inversion (LK 8.2)

- Random variable \( y_i \sim U(0,1) \)
- Transformation of \( y_i \) according to a distribution function \( F(x) \) in a random variable \( X_i \)

\[
\begin{align*}
  y_i &= F(x_i) \\
  x_i &= F^{-1}(y_i)
\end{align*}
\]
Example: Generation of an exponential distribution with a mean value of \( \lambda \)

- **Algorithm:**
  - Generate \( U \sim U(0,1) \) (pseudo random numbers)
  - Return \( X = F^{-1}(U) \)

- Random variable \( y_i \sim U(0,1) \)
- Transformation of \( y_i \) according to a distribution function \( F(x) \) in a random variable \( X_i \)

\[
F(x) = \begin{cases} 
1 - e^{-\frac{x}{\lambda}} & \text{if } x \geq 0 \\
0 & \text{otherwise}
\end{cases}
\]

\[
F^{-1}(u) = -\lambda \ln(1-u) \quad \text{symmetry} \quad F^{-1}(u) = -\lambda \ln u
\]
Composition

- Desired distribution function expressed as a convex combination of other distribution function

\[ F(x) = \sum_{j=1}^{\infty} p_j F_j(x) \text{ where } p \geq 0, \sum_{j=1}^{\infty} p_j = 1 \]

- Generate positive random integer J

\[ P(J = j) = p_j \text{ for } j = 1, 2, \ldots \]

- Return X with distribution function \( F_j \)
Convolution

- Desired random variable can be described as the sum of other random variable
  - 1. Generate \( Y_1, Y_2, Y_3, \ldots, Y_k \)
  - Return \( X = Y_1 + Y_2 + Y_3 + \cdots + Y_k \)

- Example:
  - \( k \)-Erlang distributed random variable with a mean \( \varepsilon \) can be expressed as the sum of \( k \) exponential random variables with a common mean \( k/\varepsilon \)

- Advantage: simple and intuitive approach
- Disadvantage: slow since multiple random number have to be generated in order to get a single sample
Accept-Reject-Method (LK 8.2.4)

- Inverse transform, combination, and convolution are **direct** methods (work directly with the distribution function)

- Accept-Reject is used when other methods fail or are inefficient

- Density function is complex → select a “simpler” density function $r$
Accept-Reject-Method (LK 8.2.4)

- Geometrical interpretation
  \( Y \) will be accepted if the point \((Y, U \cdot t(Y))\) falls under the curve \(f\).
- The acceptance probability is high if \(t(Y) - f(Y)\) is small.
- Majorante von \(f(x)\) \(\forall x : t(x) \geq f(x)\)
Accept-Reject-Method (LK 8.2.4)

- **Indirect approach:**
- **Preparation:**
  - We need a function $t$ that **majorizes** density $f$
    
    $$t(x) \geq f(x) \quad \text{for all } x$$
    
    $$c = \int_{-\infty}^{\infty} t(x) \, dx \geq \int_{-\infty}^{\infty} f(x) \, dx = 1$$
  
  - We obtain a density $r$ by
    $$r(x) = \frac{t(x)}{c}$$

- **Algorithm**
  
  1. Generate a random variable $Y$ according to a density $r$
  
  2. Generate a random number $U \sim U(0,1)$ (independent of $Y$)
  
  3. Return $X = Y$ if $U \leq \frac{f(Y)}{t(Y)}$ (ACCEPT)
     
     Otherwise, go back to step 1 and try again (REJECT)
Accept-Reject-Method (LK 8.2.4)

- Example: beta(4,3) distribution (6th order polynomial, hard to invert)

\[ f(x) = \begin{cases} 
60x^3(1-x)^2 & \text{if } 0 \leq x \leq 1 \\
0 & \text{otherwise} 
\end{cases} \]

Majoring function of \( x \)

![Graph showing the majoring function of \( x \)]

Value at \( x = 0.5 \): 2.0736
Accept-Reject-Method (LK 8.2.4)

- **Efficiency:**
  - Depends on the majorant series \((x)\)
  - Probability of acceptance is \(1/c\)

- **Advantage:**
  - Works for arbitrary density functions

- **Disadvantage:**
  - Number of required \(U(0,1)\) random numbers depends on the generated numbers (may causes problems with some statistics and may result variations of the simulation duration)
  - Requires at two \(U(0,1)\) random numbers in each iterations
How to generate random numbers according to different distributions?
Random numbers - Continuous

- **Uniform distribution:** \( RV \ X \sim U(a,b) \) (LK 8.3.1)
  - Density function: \( f(x) = \frac{1}{b-a}, X \in [a;b] \)
  - Range: \([a;b]\)
  - Distribution function: \( F(x) = \frac{x-a}{b-a} \)
  - Expectation: \( E(X) = \frac{a+b}{2} \)
  - Variance: \( VAR(X) = \frac{(b-a)^2}{12} \)
  - Generation: \( U \sim U(0,1), X = a + (b-a)U \)
Random numbers - Continuous

- **Triangle distribution (1/4):** \( RV \ X \sim \text{triang}(a, b, c) \)  

  - **Density function:**  
    \[
    f(x) = \begin{cases} 
    \frac{2 \cdot (x-a)}{(b-a) \cdot (c-a)} & \text{if } a \leq x \leq c \\
    \frac{2 \cdot (b-x)}{(b-a) \cdot (b-c)} & \text{if } c \leq x \leq b \\
    0 & \text{otherwise}
    \end{cases}
    \]

  - **Distribution function:**  
    \[
    f(x) = \begin{cases} 
    0 & \text{if } x < a \\
    \frac{(x-a)^2}{(b-a) \cdot (c-a)} & \text{if } a \leq x \leq c \\
    1 - \frac{(b-x)^2}{(b-a) \cdot (b-c)} & \text{if } c \leq x \leq b \\
    1 & \text{if } b < x
    \end{cases}
    \]
Random numbers - Continuous

- **Triangle distribution (2/4):** \( RV \ X \sim triang(a,b,c) \) (LK 8.3.15)
  - Use case: Project management / business simulations where only the minimum, maximum and mode are known

  - Mode \( c \)
  - Range \([a;b]\)
  - Expectation:
    \[ E(X) = \frac{a + b + c}{3} \]
  - Variance:
    \[ VAR(X) = \frac{(a^2 + b^2 + c^2 - ab - ac - bc)}{18} \]
Random numbers - Continuous

- **Triangle distribution (3/4):** $RV \ X \sim triang(a,b,c)$  
  (LK 8.3.15)

  - Generation: Inversion

  $$U \sim U(0,1), \ X = \begin{cases} 
  a + \sqrt{U(b-a) \cdot (c-a)} & 0 < U < F(c) \\
  b - \sqrt{(1-U) \cdot (b-a) \cdot (b-c)} & F(c) < U < 1
  \end{cases}$$

![Probability Density Function](image1)

![Cumulative Density Function](image2)

- Probability Density Function
- Cumulative Density Function
**Random numbers - Continuous**

- **Triangle distribution (4/4):** \( RV \ X \sim \text{triang}(a, b, c) \) (LK 8.3.15)

  Use case: risk management / project management

---

**Diagram Description**

- **Optimistic assumption**
- **Expected duration**
- **Pessimistic assumption**

**Axes:**
- **Probability**
- **Time**

**Axes Labels:**
- Minimum duration
- Expected duration
- Maximum duration
Random numbers - Continuous

- **Normal distribution (1/4):**  \( RV \ X \sim N(\mu, \sigma^2) \)  
  \( \text{Density function:} \quad f(x) = \frac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}} \cdot e^{-\left(\frac{(x-\mu)^2}{2\cdot\sigma^2}\right)} \)
  \( \text{Distribution function:} \quad F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt \)
  \( \text{Range:} \quad ]-\infty; \infty[ \)
  \( \text{Mode:} \quad \mu \)
  \( \text{Expectation:} \quad E(X) = \mu \)
  \( \text{Variance:} \quad VAR(X) = \sigma^2 \)
  \( \text{Scalability:} \quad X \sim N(0,1) \Rightarrow (\mu + \sigma X) \sim N(\mu, \sigma^2) \)
Random numbers - Continuous

- **Normal distribution (2/4):** \( RV \ X \sim N(\mu, \sigma^2) \) (LK 8.3.6)

  - Generation: Accept-Reject
    - Two independent random variables \( U_1, U_2 \sim U(0,1) \)
    - \( V_i = 2U_i - 1 \)
    - \( W = V_1^2 + V_2^2 \)
    - Algorithm:
      - Accept if \( W \leq 1 \)
      - \[ Y = \sqrt{-\frac{2 \ln W}{W}} \]
      - \( X_1 = V_1 \cdot Y \)
      - \( X_2 = V_2 \cdot Y \)
      - Reject otherwise
Random numbers

- **Normal distribution (3/4):** \( RV \quad X \sim N(\mu, \sigma^2) \)  \hspace{1cm} \text{(LK 8.3.6)}

![Probability Density Function](image1)
![Cumulative Density Function](image2)

- Probability Density Function
- Cumulative Density Function
Random numbers

- **Normal distribution (4/4):** \( RV \ X \sim N(\mu, \sigma^2) \) (LK 8.3.6)

  Use case: distribution of errors / sizes (nature)

<table>
<thead>
<tr>
<th>Körpergröße</th>
<th>Frauen</th>
<th>Männer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;150 cm</td>
<td>0,6 %</td>
<td>0,1 %</td>
</tr>
<tr>
<td>150–154 cm</td>
<td>4 %</td>
<td>0,1 %</td>
</tr>
<tr>
<td>155–159 cm</td>
<td>12,7 %</td>
<td>0,3 %</td>
</tr>
<tr>
<td>160–164 cm</td>
<td>27,0 %</td>
<td>2,3 %</td>
</tr>
<tr>
<td>165–169 cm</td>
<td>29,1 %</td>
<td>9,0 %</td>
</tr>
<tr>
<td>170–174 cm</td>
<td>17,6 %</td>
<td>19,2 %</td>
</tr>
<tr>
<td>175–179 cm</td>
<td>6,9 %</td>
<td>26,1 %</td>
</tr>
<tr>
<td>180–184 cm</td>
<td>1,8 %</td>
<td>23,9 %</td>
</tr>
<tr>
<td>185–189 cm</td>
<td>0,2 %</td>
<td>12,8 %</td>
</tr>
<tr>
<td>≥ 190 cm</td>
<td>&lt;0,1 %</td>
<td>6,3 %</td>
</tr>
</tbody>
</table>

Körpergröße der Deutschen Statistik des Sozio-oekonomischen Panels (SOEP), aufbereitet durch statista.org
Lognormal distribution (1/3): \( RV \ X \sim LN(\mu, \sigma^2) \)  

- Special property of the lognormal distribution

\[
\text{if } \ Y \sim N(\mu, \sigma^2) \quad \Rightarrow \quad e^Y \sim LN(\mu, \sigma^2)
\]

- Range: \([0, \infty)\)

- Algorithm: Composition

\[
Y \sim N(\mu, \sigma^2) \quad \Rightarrow \quad X = e^Y
\]

- Expectation: \( E(X) = e^{\mu + \frac{\sigma^2}{2}} \)

- Variance: \( \text{VAR}(X) = e^{2\mu + \sigma^2} \left(e^{\sigma^2} - 1\right) \)

Note that \( \mu \) and \( \sigma \) are NOT the mean and the variance of the lognormal distribution!
Random numbers

- **Lognormal distribution (2/3):** \( RV \ X \sim LN(\mu, \sigma^2) \) (LK 8.3.7)

  - Parameters of the normal distribution which is used to generate LN

\[
\begin{align*}
\mu &= E[Y] = \ln \left( \frac{E[X]^2}{\sqrt{E[X]^2 + \text{VAR}[X]}} \right) \\
\sigma^2 &= \text{VAR}[Y] = \ln \left( \frac{E[X]^2}{\sqrt{E[X]^2 + \text{VAR}[X]}} \right)
\end{align*}
\]
Random numbers

- **Lognormal distribution**: $RV \ X \sim LN(\mu, \sigma^2)$  
  Use case: risk management (insurance companies)
Random numbers

- **Exponential distribution (1/2):** \(RV \ X \sim \text{exp}(\lambda)\) (LK 8.3.2)

  - **Density function:** 
    \[ f(x) = \lambda \cdot e^{-\lambda x} \quad \text{für} \quad x \geq 0 \]
  
  - **Distribution function:** 
    \[ F(x) = 1 - e^{-\lambda x} \]
  
  - **Range:** \([0, \infty]\)  
    **Mode:** 0
  
  - **Expectation:** 
    \[ E(X) = \frac{1}{\lambda} \]
  
  - **Variance:** 
    \[ \text{VAR}(X) = \frac{1}{\lambda^2} \]
  
  - **Coefficient of variation:** 
    \[ c_{\text{Var}} = 1 \]
  
  - **Generation:** Inversion 
    \[ U \sim U(0,1), \ X = \frac{-\ln(U)}{\lambda} \]
Random numbers - Continuous

- **Exponential distribution (2/2):** \( RV \, X \sim \text{exp}(\lambda) \)  
  Use case: life time of structures, time between calls/requests

![Probability Density Function](image1)

![Cumulative Density Function](image2)

Pictures taken from Wikipedia
Random numbers - Continuous

- **Erlang-k distribution (1/3):** \( RV \ X \sim k - Erlang(\lambda) \) (LK 8.3.3)

  - \( RV \ X = Y_1 + Y_2 + Y_3 + \cdots + Y_k \) where the Yi’s are IID exponential random variables

  - Density function: 
    \[
    f(x) = \begin{cases} 
      \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} & \text{for } x \geq 0 \\
      0 & \text{Otherwise}
    \end{cases}
    \]

  - Distribution function: 
    \[
    F(x) = \begin{cases} 
      1 - e^{-\lambda x} \sum_{i=0}^{k-1} \frac{\lambda^i x^i}{i!} & \text{for } x \geq 0 \\
      0 & \text{Otherwise}
    \end{cases}
    \]

  RV X represents the sum of k exponential random variables
Random numbers - Continuous

- **Erlang-k distribution (2/3):** \( RV \ X \sim k - Erlang(\lambda) \) (LK 8.3.3)

- Range: \([0, \infty]\)
- Expectation: \( E(X) = \frac{k}{\lambda} \)
- Variance: \( VAR(X) = \frac{k}{\lambda^2} \)
- Mode: \( \frac{k - 1}{\lambda} \)
- Coefficient of variation: \( c_{Var} = \frac{1}{\sqrt{k}} \)
- Generation:
  - Inversion \( U_i \sim U(0,1), X = -\ln \left( \prod_{0 \leq i < k} U_i \right) \)
  - Convolution \( RV \ X = Y_1 + Y_2 + Y_3 + \cdots + Y_k \)
Random numbers - Continuous

- **Erlang-k distribution (3/3):**  
  \[ RV \ X \sim k - Erlang(\lambda) \]  
  (LK 8.3.3)

  Use case: lifetime of structures, delay in transport networks, dimensioning of systems (e.g. call center)

**Probability Density Function**

**Cumulative Density Function**
Random numbers - Continuous

- **Gamma distribution (1/3):** 
  \(RV \ X \sim \text{gamma}(\alpha, \beta)\)  
  (LK 8.3.4)

  - **Density function:** 
    \[
    f(x) = \begin{cases} 
    \frac{\beta^{-\alpha} \cdot x^{\alpha-1} e^{-\frac{x}{\beta}}}{\Gamma(\alpha)} & \text{for } x \leq 0 \\
    0 & \text{Otherwise}
    \end{cases}
    \]

  - **Distribution function:** 
    \[
    F(x) = \begin{cases} 
    1 - e^{-\frac{x}{\beta}} \cdot \sum_{0 \leq i < \alpha} \frac{(-x)^i}{\beta^i i!} & \text{for } x > 0 \\
    0 & \text{Otherwise}
    \end{cases}
    \]

  - **Parameter description:**
    - Location parameter \(\gamma\): Shifting the distribution along the \(x\)-axis
    - Scale parameter \(\beta\): Linear impact on the expectation
    - Shape parameter \(\alpha\): Changes the shape of the distribution
Random numbers - Continuous

- **Gamma distribution(2/3):**  \( RV \ X \sim \text{gamma}(\alpha, \beta) \)  
  \( (LK \ 8.3.4) \)

  - Gamma function:  
    \[
    \Gamma(z) = \begin{cases} 
    \int_0^\infty t^{z-1} e^{-t} \, dt & \text{if } x \geq 0 \\
    0 & \text{if } x < 0
    \end{cases}
    \]

  - Expectation:  
    \( E(X) = \alpha \cdot \beta \)

  - Coefficient of variation:  
    \( c_{\text{Var}} = 1 \)

  - Mode:  
    \[
    \left\{ \begin{array}{ll}
    0 & \text{if } \alpha < 1 \\
    \beta \cdot (\alpha - 1) & \text{if } \alpha \geq 1
    \end{array} \right.
    \]

  - Generation:
    - Step 1  \( X \sim \text{gamma}(\alpha, \beta) \rightarrow X = \beta \cdot Y \quad Y \sim \text{gamma}(\alpha, 1) \)
    - Step 2  Generation of  \( X \sim \text{gamma}(\alpha, 1) \) with Accept-Reject
Random numbers - Continuous

- **Gamma distribution (3/3):** \( RV \ X \sim \text{gamma}(\alpha, \beta) \) (LK 8.3.4)

  Use cases: risk management (insurance companies), service time, down time

![Probability Density Function](image)
Random numbers - Discrete

- **Uniform (discrete) (1/2)**  
  $\text{RV } X \sim DU(i, j)$  
  \[ p(k) = \begin{cases} 
  \frac{1}{j - i + 1} & \text{if } k \in \{i, i + 1, i + 2, \ldots, j\} \\
  0 & \text{Otherwise} 
\end{cases} \]
  
  - **Distribution:**
  - **Range:** $i \leq k \leq j$
  - **Expectation:** $E(X) = \frac{(i + j)}{2}$
  - **Variance:** \( \text{VAR}(X) = \frac{(j - i + 1)^2 - 1}{12} \)
  - **Generation:** Inversion  
    \[ U \sim U(0, 1) \quad X = i + \left\lfloor \left( j - i + 1 \right) \cdot U \right\rfloor \]

DU(0,1) and Bernoulli(0.5) distributions are the same.
Random numbers - Discrete

- **Uniform (discrete) (2/2)**  \( RV \ X \sim DU(i, j) \)  (LK 8.4.2)

  Use case: backoff distribution, simulation (dice, roulette, …)

\[ p(x) = \frac{1}{j - i + 1} \]
Random numbers - Discrete

- **Bernoulli (1/2)**  \(RV \ X \sim Bernoulli \ (p)\)  (LK 8.4.1)
  - Example: Flipping a coin
  - Distribution:
    \[
    p(k) = \begin{cases} 
    1 - p & \text{if } k = 0 \\
    p & \text{if } k = 1 \\
    0 & \text{Otherwise}
    \end{cases}
    \]
  - Range:
    \[i \leq k \leq j\]
  - Expectation:
    \[E(X) = p\]
  - Variance:
    \[VAR(X) = p \cdot (1 - p)\]
  - Coefficient of variation:
    \[c_{Var} = \sqrt{\frac{1 - p}{n \cdot p}}\]
Random numbers - Discrete

- **Bernoulli (2/2)** $RV \ X \sim Bernoulli \ (p)$ (LK 8.4.1)
  
  - Mode: 0 or 1 (depends on the definition of the outcome)
  
  - Generation: Inversion $U \sim U(0,1)$

  $$X = \begin{cases} 
  0 & \text{if } U < p \\
  1 & \text{Otherwise} 
  \end{cases}$$

- Distribution

  **Bernoulli** (0.3)
Random numbers - Discrete

- **N-Bernoulli (1/2)**  
  \( RV \ X \sim Bernoulli \ (n, p) \)  
  (LK 8.4.4)

  - Example: Flipping a coin \( n \) times
  
  - Distribution:  
    \[ p(k) = \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k} \quad 0 \leq k \leq n \]
  
  - Range:  
    \[ 0 \leq k \leq n \]
  
  - Expectation:  
    \[ E(X) = np \]
  
  - Variance:  
    \[ VAR(X) = n \cdot p \cdot (1 - p) \]
  
  - Coefficient of variation:  
    \[ c_{var} = \sqrt{\frac{1 - p}{n \cdot p}} \]
  
  - Use case: quality management, wrong/right decisions
Random numbers - Discrete

- N-Bernoulli (2/2)  
  \[ RV \ X \sim \text{Bernoulli} \ (n, p) \]  
  
  - Mode: 0 or 1 (depends on the definition of the outcome)
  - Generation: Composition
  
  \[ \text{Bernoulli} \ (n, p) \approx \sum_{0 \leq i < n} \text{Bernoulli} \ (p) \]

- Distribution

\[ \text{Bernoulli} \ (20,0.3) \quad \text{Bernoulli} \ (20,0.7) \]
Random numbers - Discrete

- **Geom (1/2)**  \( RV \ X \sim Geom \ (p) \)  (LK 8.4.5)

- Example: Number of unsuccessful Bernoulli – Experiments until a successful outcome (e.g. number of retransmissions)

- Distribution:  
  \[ p(x) = p \cdot (1 - p)^x \]

- Distribution function:  
  \[ F(x) = 1 - (1 - p)^{\lfloor x \rfloor + 1} \]

- Expectation:  
  \[ E(X) = \frac{1 - p}{p} \]

- Variance:  
  \[ VAR(X) = \frac{1 - p}{p^2} \]

- Coefficient of variation:  
  \[ c_{var} = \sqrt{\frac{1}{1 - p}} \]
Random numbers - Discrete

- **Geom (2/2)** $RV \ X \sim Geom \ (p)$ (LK 8.4.5)

  - **Mode:** 0
  - **Generation:** Inversion $U \sim U(0,1)$
    \[
    X = \left\lfloor \frac{\ln(U)}{\ln(1 - p)} \right\rfloor
    \]
  - **Use case:** delivery ratio in computer networks, risk management
  - **Distribution**

  - $Geom \ (0.7)$ ➔

  - $Geom \ (0.3)$ ➔

  \[ p(0) = p \]
Random numbers - Discrete

- **Poisson(1/3)**  
  \( RV \; X \sim \text{Poisson} (\lambda) \)  
  (LK 6.2.4)

  - **Example:** Number of events that occur in an interval of time when the events are occurring at a constant rate (number of items in a batch of random size)

  - **Distribution:**
    \[
    p(x) = \frac{\lambda^x}{x!} \cdot e^{-\lambda} \quad \text{if} \; x \in \{0, 1, 2, \ldots\}
    \]

  - **Distribution function:**
    \[
    F(x) = \begin{cases} 
    e^{-\lambda} \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!} & \text{if} \; x \geq 0 \\
    0 & \text{if} \; x < 0 
    \end{cases}
    \]

  - **Parameter:** \( \lambda > 0 \)
Random numbers - Discrete

- **Poisson(2/3)** \( RV \ X \sim \text{Poisson} (\lambda) \) (LK 6.2.4)
  - Range: \( \{0, 1, 2, 3, \ldots\} \)
  - Expectation: \( E(X) = \lambda \)
  - Variance: \( \text{VAR}(X) = \lambda \)
  - Coefficient of variation: \( c_{\text{Var}} = \frac{1}{\sqrt{\lambda}} \)
  - Mode
    \[ \begin{cases} 
    \lambda - 1 & \lambda \text{ is an integer} \\
    \lfloor \lambda \rfloor & \text{otherwise} 
    \end{cases} \]
  - Special characteristics:
    - \( x = 0 \) exponential distribution (time interval between two consecutive events)
    - Number of events until a certain point in time is Poisson distributed
    - Period of time until \( n \) events have occurred is Erlang distributed
Random numbers - Discrete

- **Poisson(3/3)**  
  \[ RV \ X \sim Poisson \ (\lambda) \]  
  (LK 6.2.4)  
  Use case: number of (independent) arrivals in a certain time interval
Random numbers - Discrete

- **General Discrete (1/1)**

  \[ RV \; X \sim GD \quad \text{(LK 8.4.3)} \]

  - **Distribution:**
    \[ p(x) = \begin{cases} 
    p_k & \text{if } x = x_k, \; 0 \leq k < n \\
    0 & \text{Otherwise}
    \end{cases} \]

  - **Generation:**
    Inversion
    \[ U \sim U(0,1) \]

    \[ X = x_k \; \text{, falls} \quad \sum_{j=0}^{k-1} p_j \leq U < \sum_{j=0}^{k} p_j \]
Random number generator algorithms and their quality

Some slides/figures taken from:
Oliver Rose
Averill Law, David Kelton
Wikimedia Commons (user Matt Crypto)
Dilbert
Structure of this lecture

- Generating U(0,1) random numbers
  - Motivation
  - Overview on RNG families
- Linear Congruential Generators (LCG)
- Statistical properties, statistical (empirical) tests
  - χ² test for uniformity
  - Correlation tests: Runs-up, sequence
- Theoretical aspects, theoretical tests
  - Period length
  - Spectral test
- RNG that are better than LCG
Recall the inversion method

- Generate uniformly distributed numbers $\in 0.0 \ldots 1.0$
- Compute inverse $A^{-1}(t)$ of PDF $A(t)$
- Generate samples
Generating U(0,1) random numbers is crucial

- For all random number generation methods, we need uniformly distributed random numbers from ]0,1[
  ⇒ U(0,1) random numbers are required

- Mandatory characteristics
  - Random (…obviously)
  - Uniform (make use of the whole distribution function)
  - Uncorrelated (no dependencies): difficult!
  - Reproducible (for verification of experiments)
    → use pseudo random numbers
  - Fast (usually, there is a need for a lot of samples)
RNG in simulation vs. RNG in cryptography

- Also need for random numbers in cryptography
  - Key generation
  - Challenge generation in challenge-response systems
  - ...

- Additional requirement:
  - Prediction of future “random” values by sampling previous values must not be possible
  - (In simulation: not an issue if there is no real correlation)

- Lighter requirement:
  - RNs are not used constantly, only in ~start-up phases
    ⇒ speed is not of much importance
  - (In simulation: need lots of numbers
    ⇒ speed is very important)
Generation of U(0,1) random numbers

- **Generation approaches**
  - “Real”, “natural” random numbers: sampling from radioactive material or white noise from electronic circuits, throwing dice, drawing from an urn, …
    - Problems:
      - If used online: not reproducible
      - Tables: uncomfortable, not enough samples
  - Pseudo random numbers: recursive arithmetic formulae with a given starting value (seed)
    - In hardware: shift register with feedback (based on primitive polynomials as feedback patterns)
    - In software: Linear Congruential Generator (LCG) [Lehmer, 1951], …
Generation of $U(0,1)$ pseudo-random numbers

Main families:

- Linear Congruential Generator (LCG): the simplest
- General Congruential Generators
  - Quadratic Congruential Generator
  - Multiple recursive generators
- Shift register with feedback (Tausworthe)
  - E.g., Mersenne Twister: state-of-the-art
- Composite generators: output of multiple RNG
  - E.g., use one to shuffle (“twist”) the output of the other
RNG: alternatives unsuitable for simulation

- Algorithms from cryptography
  - For example: counter→AES, counter→SHA1, counter→MD5, etc.
  - Usually way too slow
- Calculate transcendental numbers (e.g., π or e), view their digits as random
  - E.g.: digits of 100,000th decimal place of π onwards
  - Problem: Are they really random?
- Physical generators (cf. previous lecture)
  - Not reproducible, no seed
- Tables with pre-computed random numbers
  - We need too many random numbers, the tables would have to be huge…
Linear Congruential Generators

- Calculate RN from previous RN using some formula.
- Sequence of integers $Z_1, Z_2, \ldots$ defined by

$$Z_i = (a \cdot Z_{i-1} + c) \pmod{m}$$

- with modulus $m$, multiplier $a$,
  increment $c$, and seed $Z_0$

- $c=0$: multiplicative LCG
  Example:
  $$Z_i = 16807 \cdot Z_{i-1} \pmod{2^{31} - 1}$$
  (Lewis, Goodman, Miller, 1969)

- $c>0$: mixed LCG
...but they don’t create floats, but integers > 1?!

- Obviously,
  \[ Z_i = \text{something mod } m \]
  and
  \[ \text{something mod } m < m \]

- ⇒ Just normalise the result!
  - Divide by \( m \)? But then, 1.0 cannot be attained.
  - Better: Divide by \( m - 1 \).
Do they really generate uniformly distributed random numbers?

- Test for uniformity:
  - Create a number of samples from RNG
  - Test if these numbers are uniformly distributed

- A number of statistical tests to do this:
  - \( \chi^2 \) test (deutsch: Chi-Quadrat-Anpassungstest)
  - Kolmogorov-Smirnov test
  - … and a whole lot of others! For example:
    - Cramér-von Mises test
    - Anderson-Darling test

- Graphical examination (not real tests):
  - Plot histogram / density / PDF
  - Distribution-function-difference plot
  - Quantile-quantile plot (Q-Q plot)
  - Probability-probability plot (P-P plot)

(later in course)
Histogram

- Given a series of $n$ measurements $X_i$
- Partition the domain $\min\{X_i\} \ldots \max\{X_i\}$ into $m$ intervals $I_1 \ldots I_m$
- $\sim$ discretised density function
- Recommendation: $m \approx \sqrt{n}$
What the histogram can reveal (1)

Obviously not U(0,1) random variables:

(...okay, we could have calculated min and max instead of plotting the histogram)
What the histogram can reveal (2)

Obviously not U(0,1) random variables:

Histogram of RN

Frequency

RN

0.0
0.2
0.4
0.6
0.8
1.0
What the histogram can reveal (3a)

Looks like a U(0,1) random variable at first sight…:

Histogram of RN

Frequency

0

0.0

0.2

0.4

0.6

0.8

1.0

RN

2000

1500

1000

500

0
What the histogram can reveal (3b)

...but is obviously no U(0,1) random variables: huge gaps!
Is a histogram just a bar plot?

- Gummibears – Original Haribo 300g (~130 Gummibears per package)

“Histograms” are based on samples taken from a 300g package
Is a histogram just a bar plot?

- Gummibears – Eaten by students during the lecture

![Histogram of Gummibears Eaten by Students](image)
Is a histogram just a bar plot?

- Gummibears – Original – 1500g

Based on samples taken from 5 x 300g packages
Is a histogram just a bar plot? – No!

- **Histogram**
  - X axis:
    - some *scalar* value, e.g., [0…1], or $]-\infty…+\infty[$, etc.
    - Divided into bins ("classes")
  - Y axis: number of occurrences per class

- **Barplot**
  - X axis: Some *categorical* value, e.g., colour, or student name, etc.
  - Y axis: number of occurrences per class
Statistical tests

- Does the analytical distribution correspond to the empirical distribution calculated from the sample set?

Statistical tests

- Scenario: Given a set of measurements, we want to check if they conform to a distribution; here: U(0,1)
- Graphs like presented before are nice indicators, but not really tangible: “How straight is that line?” etc.
- We want clearer things: Numbers or yes/no decisions
- Statistical tests can do the trick, but…
  - Warning #1: Tests only can tell if measurements do not fit a particular distribution—i.e., no “yes, it fits” proof!
  - Warning #2: The result is never absolutely certain, there is always an error margin.
  - Warning #3: Usually, the input must be ‘iid’:
    - Independent
    - Identically distributed
  - ⇒You never get a ‘proof’, not even with an error margin!
χ² test (Pearson, 1900)

- Input:
  - Series of \( n \) measurements \( X_1 \ldots X_n \)
  - A distribution function \( f \) (the ‘theoretical function’)

- Measurements will be tested against the distribution
  - formal comparison of a histogram with the density function of the theoretical function

- Null hypothesis \( H_0 \):
  The \( X_i \) are IID random variables with distribution function \( f \)
χ² test: How it works

- Divide the sample range into k intervals of equal probability
- Count how many \( X_i \) fall into which interval (histogram):
  \[ N_j := \text{number of } X_i \text{ in } j\text{-th interval} \ [a_{j-1} \ldots a_j] \]
- Calculate how many \( X_i \) would fall into the \( j\)-th interval if they were sampled from the theoretical distribution:
  \[ p_j := \int_{a_{j-1}}^{a_j} f(x) \, dx \quad (f: \text{density of theor. dist.}) \]
- Calculate squared normalised difference between the observed and the expected samples per interval:
  \[ \chi^2 := \sum_{j=1}^{k} \frac{(N_j - np_j)^2}{np_j} \]
- Obviously, if \( \chi^2 \) is “too large”, the differences are too large, and we must reject the null hypothesis
- **But what is “too large”?**
χ² test: Using the χ² distribution

- The χ² distribution
  - A test distribution
  - Parameter: degrees of freedom (short df)
  - \( \chi^2(k-1 \text{ df}) = \Gamma\left(\frac{1}{2}(k-1) , 2\right) \) (gamma distribution)
  - Mathematically: The sum of n independent squared normal distributions

- Compare the calculated χ² against the χ² distribution
  - If we use k intervals, then χ² is distributed corresponding to the χ² distribution with k–1 degrees of freedom
  - Let \( \chi^2_{k-1,1-\alpha} \) be the \((1-\alpha)\) quantile of the distribution
  - \( \alpha \) is called the confidence level
  - Reject H0 if \( \chi^2 > \chi^2_{k-1,1-\alpha} \) (i.e., the \( X_i \) do not follow the theoretical distribution function)
The $\chi^2$ distribution with $k-1$ degrees of freedom

$f(x)$

Chi-square density with $k-1$ df

Shaded area = $\alpha$

Do not reject

Reject


\( \chi^2 \) test and degrees of freedom

- \( \chi^2 \) test can be used to test against *any* distribution.
- Easy in our case: We know the parameters of the theoretical distribution \( f \) — it’s U(0,1).
- Different in the general case:
  - For example, we may know it’s N(\( \mu \), \( \sigma \)) (normal distribution) but we know neither \( \mu \) nor \( \sigma \).
  - Fitting a distribution: Find parameters for \( f \) that make \( f \) fit the measurements \( X_i \) best.
  - Topic of a later lecture.
- Theoretically:
  - Have to estimate \( m \) parameters.
  - Also have to take \( \chi^2_{k-m-1,1-\alpha} \) into account.
- Practically:
  - \( m \leq 2 \) and large \( k \).
  - Don’t care...
\( \chi^2 \): which parameters?

- **How many intervals** (k)?
  - A difficult problem for the general case
  - **Warning**: A smaller or a greater k may change the outcome of the test!
  - As a general rule, use k between \( n/5 \) and \( \sqrt{n} \)
  - As a general rule, make the intervals equal-sized
  - As another general rule, make sure that \( \forall j: np_j \geq 5 \) (i.e., have enough samples that we expect to have at least 5 samples in each interval)

- \( \Rightarrow \) As a general rule, you need a lot of measurements!

- The larger the number of measurements, the higher the chance that the assumption is rejected.

- **What confidence level**?
  - At most \( \alpha = 0.10 \) (almost too much);
    typical values: 0.001, 0.01, 0.05 [ , and 0.10]
  - The smaller, the higher confidence in the test result
Kolmogorov-Smirnov test (KS test)

- Samples $X_i, i \in 0 \leq i < n$
- Hypothesis:
  Samples $X_i$ are iid and follow the distribution $\hat{F}(x)$
- Definition: empirical distribution $X_i$
  \[ F_n(x) = \frac{\# X_i \leq x}{n} \]  
  ( $F_n(x)$ step function)
- Test $D_n(x)$: largest vertical difference between $F_n(x)$ and $\hat{F}_n(x)$:
  \[ D_n(x) = \max \left\{ F_n(n) - \hat{F}(x) \right\} \]
  \[ D_n^+ = \max_{1 \leq i \leq n} \left\{ \frac{i}{n} - \hat{F}(X_{(i)}) \right\}, D_n^- = \max_{1 \leq i \leq n} \left\{ \hat{F}(X_{(i)}) - \frac{i-1}{n} \right\}, D_n = \max \{D_n^+, D_n^-\} \]

Note: $X_i$ represents the sorted samples in ascending order
Kolmogorov-Smirnov test

- Example 1: $n=4$, samples are iid and follow the distribution $\hat{F}(x)$

Geometric meaning of the K-S test statistic $D_n$ for $n = 4$.

Example 2:

An example in which the K-S test statistic $D_n$ is not equal to $D'_n$.

Kolmogorov-Smirnov test

- $H_0$ is accepted if

$$\left(\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right)D_n < c_{1-\alpha}$$

<table>
<thead>
<tr>
<th>1-(\alpha)</th>
<th>0.850</th>
<th>0.900</th>
<th>0.950</th>
<th>0.975</th>
<th>0.990</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_{1-\alpha}</td>
<td>1.138</td>
<td>1.224</td>
<td>1.358</td>
<td>1.480</td>
<td>1.628</td>
</tr>
</tbody>
</table>

- Advantages:
  - No grouping into intervals required
  - Valid for any sample size, not only for large \(n\)
  - More powerful than \(\chi^2\) for a number of distributions

- Disadvantages:
  - Applicability more limited than \(\chi^2\)
  - Difficult to apply to discrete data
  - If distribution needs to be fitted (unknown parameters), then K-S works only for a number of distributions
Alternatives to $\chi^2$ test

- Other tests:
  - Anderson–Darling test (A–D test)
    - Higher power than K-S for some distributions
  - …a lot of other tests
    - Rule of thumb: The more specialised the test, the higher its power compared to other tests – but the less generally applicable
Tests for uniformity: limitations

- Consider this sequence of drawn "random numbers":

- They are in U(0,1) … but do they seem random!?
Recall our requirements for RNG

- RNs have to be uncorrelated — how should we test this?
- Statistical tests:
  - Draw some random numbers and examine them
    - Runs-up test
    - Serial test
- Theoretical parameters and theoretical tests:
  - Length of period
  - Spectral test
  - Lattice test
Runs-up test

- **Run up**: the length of a contiguous sequence of monotonically increasing $X_i$.

- **Example sequence**:
  - $0.86 >$ length: 1
  - $0.11 < 0.23 >$ length: 2
  - $0.03 < 0.13 >$ length: 2
  - $0.06 < 0.55 < 0.64 < 0.87 >$ length: 4
  - $0.10$ length: 1

- Calculate $r_i$ (number of runs up of length $i$)
- Compute a test statistic value $R$, using the $r_i$ and a bestranging zoo of esoteric constants $a_{ij}$ and $b_j$
- $R$ will have an approximate $\chi^2$ distribution with 6 df.
  - You just have to believe me there – and I have to believe the literature…
Spectral test

- Find possible correlations between subsequently drawn values
- Visual “tests”:
  - 2D plot of $X_i$ and $X_{i-1}$
  - 3D plot of $X_i$ and $X_{i-1}$ and $X_{i-2}$
- Generalisation: Serial test
LCG examples (1/5)

\[ Z_i = a \cdot Z_{i-1} \pmod{61} \]

\( a = 7 \) \hspace{1cm} \( a = 43 \) \hspace{1cm} \( a = 31 \)
LCG examples (2/5)

\[ X(n+1) = (3141592653 \times X(n) + 2718281829) \mod 2^{35}, \quad X(0) = 5772156649, \quad 0 < n < 10000 \]
LCG examples (3/5)

\[ X(n+1) = (129 \times X(n) + 1) \mod 2^{35}, \quad X(0) = 0, \quad 0 < n < 50000 \]
LCG examples (4/5)

\[ X(n+1) = (262145 \times X(n) + 1) \mod 2^{35}, \quad X(0) = 47594188, \quad 0 < n < 50000 \]
LCG examples (5/5)

n=81

n=729

n=2197

n=19683
Serial test: “a generalised and formalised version of the plots”

- Consider **non-overlapping** d-tuples of subsequently drawn random variables $X_i$:
  
  $U_1 = (X_1, X_2, \ldots, X_d)$ \quad $U_2 = (X_{d+1}, X_{d+2}, \ldots, X_{2d})$ \quad \ldots

- These $U_i$’s are vectors in the d-dimensional space

- If the $X_i$ are truly iid random variables, then the $U_i$ are truly random iid vectors in the space $[0\ldots 1]^d$
  (the d-dimensional hypercube)

- **Test for d-dimensional uniformity** (rough outline):
  - Divide $[0\ldots 1]^d$ into k equal-sized sub intervals
  - Calculate a value $\chi^2(d)$ based on the number of $U_i$ for each possible interval combination
  - $\chi^2(d)$ has approximate distribution $\chi^2(k^d-1 \text{ df})$
  - Rest: same as $\chi^2$ test above
The infamous RANDU generator

- A LCG with setup:
  \[ Z_i = 65,539 \cdot Z_{i-1} \mod 2^{31} \]

- Advantage: It’s fast.
  - \( \mod 2^{31} \) can be calculated with a simple AND operation
  - 65,539 is a bit more than \( 2^{16} \); thus the multiplication (=expensive operation) can be replaced by a bit shift of 16 bit plus three additions (=cheap operations)
  - Why 65,539? It’s a prime number.

- Disadvantage:
  - An infamously bad RNG! Never, ever use it!
  - \( d \geq 3 \): The tuples are clumped into 15 plains (remember the animated 3D cube? That was RANDU!)

- A lot of simulations in the 1970s used RANDU
  \( \Rightarrow \) sceptical view on simulation results from that time
Theoretical parameters, theoretical tests

- Tests so far: Based on drawing samples from RNG
- No absolute certainty!
  - Usually, only a small subset of entire period is used
  - Remember the $\chi^2$ test

- Theoretical parameters and tests
  - Based directly on the algorithm and its parameters
  - No samples to be drawn – not a real “statistical test”
  - Usually quite complicated
Period length

- After some time, the “random” numbers must repeat themselves.
  Why?
  - LCG: \( Z_i \) is entirely determined by \( Z_{i-1} \)
  - The same \( Z_{i-1} \) will always produce the same \( Z_i \)
  - There are only finitely many different \( Z_i \)
  - How many?
    We take mod \( m \) ⇒ at most \( m \) different values

- Call this the period length
Theorem by Hull and Dobell 1962

- A LCG has full period if and only if the following three conditions hold:
  1. $c$ is relatively prime to $m$ (i.e., they do not have a prime factor in common)
  2. If $m$ has a prime factor $q$, then $(a-1)$ must have a prime factor $q$, too
  3. If $m$ is divisible by 4, then $(a-1)$ must be divisible by 4, too

- Prime numbers play an important role
  - Remember RANDU? At least, it used a prime number…

- Multiplicative RNGs (i.e., no increment $Z_i+c$) cannot have period $m$. (But period $(m-1)$ is possible if $m$ and $a$ are chosen carefully.)
LCG and period length considerations

- On 32 bit machines, $m \leq 2^{31}$ or $m \leq 2^{32}$ due to efficiency reasons ⇒ period length 4.3 billion
- Calculating that many random numbers only takes a couple of seconds on today’s hardware
- Theory suggests to use only $\sqrt{\text{period length}}$ numbers; that’s only 65,000 random numbers
- How many random numbers do we need? Example:
  - Simulate behaviour of 1,000 Web hosts
  - Each host consumes on average 1 random number per simulation second
  - Result: We can only simulate for one minute!
- ⇒ We need much longer period lengths
  - Okay… so let’s just use a 64-bit LCG, no?
Spectral test (coarse description)

- The theoretical variant of the serial test
- Observation by Marsaglia (1968):
  
  "Random numbers fall mainly in planes."

  - Subsequent overlapping (!) tuples $U_i$:
    
    $U_1=(X_1, X_2, \ldots X_d)$  
    $U_2=(X_2, X_3, \ldots, X_{d+1})$  
    
    fall on a relatively small number of $(d-1)$-dimensional hyperplanes within the $d$-dimensional space

  - Note the difference to the serial test! (overlapping)
  - ‘Lattice’ structure

- Consider hyperplane families that cover all tuples $U_i$
- Calculate the maximum distance between hyperplanes. Call it $\delta_d$.
- If $\delta_d$ is small, then the generator can ~uniformly fill up the $d$-dimensional space
Spectral test and LCG

- For LCG, it is possible to give a theoretical lower bound $\delta_d^*$:
  $$\delta_d \geq \delta_d^* = 1 / (y_d m^{1/d})$$
- $y_d$ is a constant whose exact value is only known for $d \leq 8$ (dimensions up to 8)
- LCG do not perform very well in the spectral test:
  - All points lie on at most $m^{1/n}$ hyperplanes (Marsaglia’s theorem)
  - Serial test: similar
  - There are way better random number generators than linear congruential generators.
Discussion of LCGs

- **Advantages:**
  - Easy to implement
  - Reproducible
  - Simple and fast

- **Disadvantages:**
  - Period (length of a cycle) depends on parameters $a$, $c$, and $m$
  - Distribution and correlation properties of generated sequences are not obvious
  - A value can occur only once per period (unrealistic!)
  - By making a bad choice of parameters, you can screw up things massively
  - Bad performance in serial test / spectral test even for good choice of parameters
Beyond LCGs

- Why linear?
  - Quadratic congruential generator:
    \[ Z_i = (a \cdot (Z_{i-1})^2 + a' \cdot Z_{i-1}) \mod m \]
  - But: period is still at most \( m \)

- Why only use one previous \( X_i \)?
  - Multiple recursive generator:
    \[ Z_i = (a_1Z_{i-1} + a_2Z_{i-2} + a_3Z_{i-3} + \ldots + a_qZ_{i-q}) \mod m \]
  - Period can be \( m^q-1 \) if parameters are chosen properly

- Why not change multiplier \( a \) and increment \( c \) dynamically, according to some other congruential formula?
  - Seems to work \(~\) alright
Feedback Shift Register Generators (1/2)

- Linear feedback shift register generator (LFSR) introduced by Tausworthe (1965)
- Operate on binary numbers (bits), not on integers
- Mathematically, a multiple recursive generator:
  \[ b_i = (c_1 b_{i-1} + c_2 b_{i-2} + c_3 b_{i-3} + \ldots + c_q b_{i-q}) \mod 2 \]
  - \( c_i \): constants that are either 0 or 1
  - \( c_q = 1 \) (why?)
  - Observe that + mod 2 is the same as XOR (makes things faster)

- In hardware:
Feedback Shift Register Generators (2/2)

- Usually only two $c_j$ coefficients are 1, thus:
  \[ b_i = (b_{i-r} + b_{i-q}) \mod 2 \]

- LFSR create random bits, not integers
  - Easy solution: Concatenate $\ell$ bits to form an $\ell$-bit integer

- Properties
  - Period length [of the $b_i$ bits] = $2^q - 1$, if parameters chosen accordingly (Note: characteristic polynomial has to be primitive over Galois field $\mathbb{F}_2$ …)
  - Period length of the generated ints accordingly lower?
    - Depends on whether $\ell \mid 2^q - 1$ or not
    - This is probably not the case
    - In general: period length = $2^q - 1 / \gcd(2^q - 1, \ell)$  [deutsch: ggT]
    - But there may be some correlation after one $2^q - 1$ “bit period”
  - Statistical properties not very good
  - Combining LFSRs improves statistics and period
Generalised feedback shift register (GFSR)

- Lewis and Payne (1973)

- To obtain sequence of $\ell$-bit integers $Y_1, Y_2, \ldots$:
  - Leftmost bit of $Y_i$ is filled with LFSR-generated bit $b_i$
  - Next bit of $Y_i$ is filled with LFSR-generated bit after some “delay” $d$: $b_{i+d}$
  - Repeat that with same delay for remaining bits up to length $\ell$

- Mathematical properties
  - Period length can be very large if $q$ is very large, e.g., Fushimi (1990): period length $= 2^{521} - 1 = 6.86 \cdot 10^{156}$
  - If $2^\ell < 2^q - 1$, then many $Y_i$’s will repeat during one period run
  - If two bits (as with LFSR), then $Y_i = Y_{i-r} \oplus Y_{i-q}$
Long period lengths and repeated values

“If $2^\ell < 2^q - 1$, then many $Y_i$’s will repeat during a period run.”
- $\ell$: number of bits of the integer output
- $2^q - 1$: period length

Is that good or bad?
- This is a general question – it relates to all RNGs, not only GFSR

Consider this example:
- $\ell = 2 \Rightarrow$ only 4 different numbers
- If $q = 4$ as well, then we always would get, e.g.
  1, 4, 2, 3, 1, 4, 2, 3, 1, 4, 2, 3, 1, 4, 2, 3, 1, 4, 2, 3
- But we would want something like
  1, 4, 2, 2, 1, 4, 3, 1, 1, 4, 3, 3, 1, 4, 2, 3, 2, 2, 4, 1, 4, 3, 2, 3

Clearly, it’s good that numbers repeat during one period
⇒ Clearly, it’s good that we have a very long period length
Mersenne Twister (1/2)

- Before we go into the mathematical details…
  - Very, very long period length: \(2^{19,937} - 1 > 10^{6,000}\)
  - Very good statistical properties: OK in 623 dimensions
  - Quite fast

- State of the art: One of the best we have right now
  - The RNG of choice for simulations
  - Default RNG in Python, Ruby, Matlab, GNU R
  - Admittedly, there are even (slightly) better RNGs, cf. TestU01 paper

- Three warnings:
  - Not suitable for cryptographic applications:
    Draw 624 random numbers and you can predict all others!
  - Can take some time (“warm-up period”) until the stream generates good random numbers
    - Usually hidden from programmer through library
    - If in doubt, discard the first 10,000 … 100,000 drawn numbers
  - There also are other good modern RNGs, e.g., WELL
Mersenne Twister (2/2)

- **Twisted GFSR (TGFSR)**
  - Matsumoto, Kurita (1992, 1994)
  - Replace the recurrence of the GFSR by
    \[ Y_i = Y_{i-r} \oplus A \cdot Y_{i-q} \]
    where:
    - the \( Y_i \) are \( \ell \times 1 \) binary vectors
    - \( A \) is an \( \ell \times \ell \) binary matrix
  - Period length = \( 2^{q\ell-1} \) with suitable choices for \( r, q, A \)

- **Mersenne Twister (MT19937)**
  - Clever choice of \( r, q, A \) and the first \( Y_i \) to obtain good statistical properties
  - Period length \( 2^{19,937-1} = 4.3 \cdot 10^{6001} \) (Mersenne prime: \( 2^n-1 \))
Beyond Mersenne Twister

- Even better alternative: WELL
  - Well Equidistributed Long-period Linear
  - Panneton, L‘Écuyer, Matsumoto: *Improved Long-Period Generators Based on Linear Recurrences Modulo 2*, 2006
  - Period length: $2^k - 1$ where $k \in \{512,1024,19937,44497\}$
  - Better statistical properties than Mersenne twister
  - Speed comparable to Mersenne Twister
  - No warm-up period

- SIMD-oriented Fast Mersenne Twister (SFMT)
  - Faster than Mersenne Twister
  - Uses features of modern CPUs: 128 bit instructions, Pipelining
  - Also has better statistical properties than Mersenne Twister
Digression: Period lengths revisited

What period lengths do we actually require?

- **Estimate #1:**
  - A cluster of 1 million hosts
  - each of which draws $1,000,000 \cdot 2^{32}$ per second (~1,000,000 times as fast as today’s desktop PCs)
  - for ten years
  - will require…
    - $5.6 \cdot 10^{27}$ random numbers
    - (Make the PCs again $10^6$ times faster ⇒ $5.6 \cdot 10^{33}$)

- **Estimate #2:** What’s the estimated number of electrons within the observable universe (a sphere with a radius of ~46.5 billion light years)
  - About $10^{80}$ (± take or leave a few powers of 10)
Test batteries

- A lot of tests, a lot of different RNGs
- How to compare them?
- Benchmark suites (‘Test batteries’)
  that bundle many statistical tests:
  - TestU01 (L’Écuyer)
  - DIEHARD suite (Marsaglia)
  - NIST test suite (National Institute of Standards and Technologies;
    ≡ Physikalisch-Technische Bundesanstalt)
Conclusion: Quality tests for RNG

- **Empirical tests (based on generated samples)**
  - For U(0,1) distribution: $\chi^2$ test
  - For independence: autocorrelation, serial, run-up tests

- **Theoretical tests (based on generation formula)**
  - Basic idea: test for k-dimensional uniformity
  - Points of sequence form system of hyperplanes
  - Computation of distance of hyperplanes for several dimensions k
  - Rather difficult optimization problem

- **Conclusion**
  - Implement/use only tested random number generators from literature, no “home-brewed” generators!
  - When in doubt, use the Mersenne Twister (but not for cryptography!)
RNG: outlook

- A wide research field, still somewhat active
  - Many more algorithms exist
  - Many more tests for randomness exist
  - More are being developed

- If you are interested in this topic, you might want to have a look at this quite readable paper:
  - L’Écuyer, Simard
    TestU01: a C library for empirical testing of random number generators
    ACM Transactions on Mathematical Software, Volume 33, No. 4, 2007