Random Number Generation Using Normal Numbers

Michael Mascagni[1] and F. Steve Brailsford[2]

Department of Computer Science[1,2]
Department of Mathematics[1]
Department of Scientific Computing[1]
Graduate Program in Molecular Biophysics[1]
Florida State University, Tallahassee, FL 32306 **USA**
AND
Applied and Computational Mathematics Division, Information Technology Laboratory[1]
National Institute of Standards and Technology, Gaithersburg, MD 20899-8910 **USA**
E-mail: mascagni@fsu.edu or mascagni@math.ethz.ch
or mascagni@nist.gov
URL: http://www.cs.fsu.edu/∼mascagni

In collaboration with Dr. David H. Bailey, Lawrence Berkeley Laboratory and UC Davis

Outline of the Talk

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r / \log s$ is a rational number

Normal Numbers: Types of Numbers

- Rational numbers - $\frac{p}{q}$ where p and q are integers
- Irrational numbers - not rational
- b-dense numbers - α is b-dense $\iff$ in its base-b expansion every possible finite string of consecutive digits appears
- If α is b-dense then α is also irrational; it cannot have a repeating/terminating base-b digit expansion
- Normal number - α is b-normal $\iff$ in its base-b expansion every string of k base-b digits appears with a limiting frequency $1/b^k$
- A real number, α, having a different base-b expansion for each integer $b > 2$, may be normal in one base but not in another
- A normal number in base r is normal in base s if $\log r/\log s$ is a rational number

Normal Numbers: More Facts

- Every b-normal sequence is b-dense
- A number that is b-normal for every $b = 2, 3, 4, \ldots$ is said to be absolutely normal
- Almost all real numbers in [0, 1) are absolutely normal, and they are dense in [0,1)
- The non-normal numbers in [0,1) are also uncountable

Normal Numbers: More Facts

- Every *b*-normal sequence is *b*-dense
- A number that is *b*-normal for every $b = 2, 3, 4, \ldots$ is said to be absolutely normal
- Almost all real numbers in [0, 1) are absolutely normal, and they are dense in [0,1)
- The non-normal numbers in [0,1) are also uncountable

Normal Numbers: More Facts

- Every b-normal sequence is b-dense
- A number that is b-normal for every $b = 2, 3, 4, \ldots$ is said to be absolutely normal
- Almost all real numbers in [0, 1) are absolutely normal, and they are dense in [0,1)
- The non-normal numbers in [0,1) are also uncountable

Normal Numbers: More Facts

- Every b-normal sequence is b-dense
- A number that is b-normal for every $b = 2, 3, 4, \dots$ is said to be absolutely normal
- Almost all real numbers in [0, 1) are absolutely normal, and they are dense in [0,1)
- The non-normal numbers in [0,1) are also uncountable

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\ldots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\ldots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\ldots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\ldots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\dots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\dots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\dots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\dots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\ldots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\ldots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\ldots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\ldots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\ldots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\ldots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10:
     $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\ldots$
   - Concatenate the primes in base b:
     $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\ldots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\ldots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\ldots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\ldots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\ldots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\dots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\dots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\dots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\dots$

Special Normal Numbers

Some examples of numbers that are provably normal

1. Champernowne numbers
   - $C_2 = 0.(1)(10)(11)(100)(101)(110)(111)(1000)\ldots$
   - $C_{10} = 0.(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)\ldots$
   - Thus, C_b is b-normal by construction
2. Copeland-Erdös constants
   - Concatenate the primes in base 10: $0.(2)(3)(5)(7)(11)(13)(17)(19)(23)(29)(31)(37)(41)(43)\ldots$
   - Concatenate the primes in base b: $0.((p_1)_b)((p_2)_b)((p_3)_b)((p_4)_b)((p_4)_b)((p_5)_b)\ldots$

Special Normal Numbers

3. Stoneham numbers:

$$\alpha_{b,c} = \sum_{n=c^k>1} \frac{1}{b^n n} = \sum_{k=1}^{\infty} \frac{1}{b^{c^k} c^k}$$

- $\alpha_{b,c}$ is b-normal when c is an odd prime, and b and c are relatively prime

4. Korobov numbers:

$$\beta_{b,c,d} = \sum_{n=c,c^d,c^{d^2},c^{d^3},\dots} \frac{1}{nb^n}$$

- $\beta_{b,c,d}$ is b-normal when $b, c, d > 1$ and b and c are relatively prime

Special Normal Numbers

3. Stoneham numbers:

$$\alpha_{b,c} = \sum_{n=c^k>1} \frac{1}{b^n n} = \sum_{k=1}^{\infty} \frac{1}{b^{c^k} c^k}$$

- $\alpha_{b,c}$ is b-normal when c is an odd prime, and b and c are relatively prime

4. Korobov numbers:

$$\beta_{b,c,d} = \sum_{n=c,c^d,c^{d^2},c^{d^3},\ldots} \frac{1}{nb^n}$$

- $\beta_{b,c,d}$ is b-normal when $b, c, d > 1$ and b and c are relatively prime

Special Normal Numbers

3. Stoneham numbers:

$$\alpha_{b,c} = \sum_{n=c^k>1} \frac{1}{b^n n} = \sum_{k=1}^{\infty} \frac{1}{b^{c^k} c^k}$$

- $\alpha_{b,c}$ is b-normal when c is an odd prime, and b and c are relatively prime

4. Korobov numbers:

$$\beta_{b,c,d} = \sum_{n=c,c^d,c^{d^2},c^{d^3},\ldots} \frac{1}{nb^n}$$

- $\beta_{b,c,d}$ is b-normal when $b, c, d > 1$ and b and c are relatively prime

Special Normal Numbers

3. Stoneham numbers:

$$\alpha_{b,c} = \sum_{n=c^k>1} \frac{1}{b^n n} = \sum_{k=1}^{\infty} \frac{1}{b^{c^k} c^k}$$

   - $\alpha_{b,c}$ is b-normal when c is an odd prime, and b and c are relatively prime

4. Korobov numbers:

$$\beta_{b,c,d} = \sum_{n=c,c^d,c^{d^2},c^{d^3},\dots} \frac{1}{nb^n}$$

   - $\beta_{b,c,d}$ is b-normal when $b, c, d > 1$ and b and c are relatively prime

Equidistribution

- Let $\{x_i\}$ be an infinite sequence of numbers in [0,1); $E \subset [0,1)$, a subset, and $\#(E; N)$ the number of $\{x_i\}$, $1 \leq i \leq N \in E$, then $\{x_i\}$ is uniformly distributed modulo 1 (UDM1) if:

$$\lim_{N\to\infty} \frac{\#([a,b);N)}{N} = b - a, \quad \forall a,b \in \mathbb{R}, \quad 0 \leq a < b \leq 1$$

- $\{x_i\}$ is UDM1 if $\forall$ Riemann integrable function f on [0,1):

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \int_0^1 f(x)dx$$

- Bohl, Sierpinksi, Weyl theorem: If α is irrational, then $\forall$ Riemann integrable function f on [0,1) we have:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(\{i\alpha\}) = \int_0^1 f(x)dx$$

- Thus $x_i = \{i\alpha\} = i\alpha - \lfloor i\alpha \rfloor$ is UDM1 when α is irrational

Equidistribution

- Let $\{x_i\}$ be an infinite sequence of numbers in [0,1); $E \subset [0,1)$, a subset, and $\#(E;N)$ the number of $\{x_i\}$, $1 \leq i \leq N \in E$, then $\{x_i\}$ is uniformly distributed modulo 1 (UDM1) if:

$$\lim_{N\to\infty} \frac{\#([a,b);N)}{N} = b - a, \quad \forall a,b \in \mathbb{R}, \quad 0 \leq a < b \leq 1$$

- $\{x_i\}$ is UDM1 if $\forall$ Riemann integrable function f on [0,1):

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \int_0^1 f(x)dx$$

- Bohl, Sierpinksi, Weyl theorem: If α is irrational, then $\forall$ Riemann integrable function f on [0,1) we have:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(\{i\alpha\}) = \int_0^1 f(x)dx$$

- Thus $x_i = \{i\alpha\} = i\alpha - \lfloor i\alpha \rfloor$ is UDM1 when α is irrational

Equidistribution

- Let $\{x_i\}$ be an infinite sequence of numbers in [0,1); $E \subset [0,1)$, a subset, and $\#(E; N)$ the number of $\{x_i\}$, $1 \le i \le N \in E$, then $\{x_i\}$ is uniformly distributed modulo 1 (UDM1) if:

$$\lim_{N \to \infty} \frac{\#([a, b); N)}{N} = b - a, \quad \forall a, b \in \mathbb{R}, \quad 0 \le a < b \le 1$$

- $\{x_i\}$ is UDM1 if $\forall$ Riemann integrable function f on [0,1):

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \int_0^1 f(x)dx$$

- Bohl, Sierpinksi, Weyl theorem: If α is irrational, then $\forall$ Riemann integrable function f on [0,1) we have:

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} f(\{i\alpha\}) = \int_0^1 f(x)dx$$

- Thus $x_i = \{i\alpha\} = i\alpha - \lfloor i\alpha \rfloor$ is UDM1 when α is irrational

Equidistribution

- Let $\{x_i\}$ be an infinite sequence of numbers in [0,1); $E \subset [0,1)$, a subset, and $\#(E;N)$ the number of $\{x_i\}$, $1 \leq i \leq N \in E$, then $\{x_i\}$ is uniformly distributed modulo 1 (UDM1) if:

$$\lim_{N\to\infty} \frac{\#([a,b);N)}{N} = b - a, \quad \forall a,b \in \mathbb{R}, \quad 0 \leq a < b \leq 1$$

- $\{x_i\}$ is UDM1 if $\forall$ Riemann integrable function f on [0,1):

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \int_0^1 f(x)dx$$

- Bohl, Sierpinksi, Weyl theorem: If α is irrational, then $\forall$ Riemann integrable function f on [0,1) we have:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(\{i\alpha\}) = \int_0^1 f(x)dx$$

- Thus $x_i = \{i\alpha\} = i\alpha - \lfloor i\alpha \rfloor$ is UDM1 when α is irrational

Equidistribution

- Equivalently the Weyl criterion can be applied empirically: $\{x_i\}$ is UDM1 $\iff$ for every integer $h \neq 0$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} e^{2\pi i h x_i} = 0$$

Note: if $\{x_i\}$ are random then $\sum_{i=1}^{N} e^{2\pi i h x_i} \approx O(\sqrt{N})$

- Discrepancy - the number of points in the sequence falling into an arbitrary set B is close to proportional to the measure of B
- There exists an absolute constant C such that for any positive integer m the discrepancy of any sequence $\{x_i\}$ satisfies

$$D_N < C\left(\frac{1}{m}\sum_{h=0}^{m}\frac{1}{h}\left|\frac{1}{N}\sum_{i=0}^{N-1} e^{2\pi i h x_i}\right|\right)$$

Equidistribution

- Equivalently the Weyl criterion can be applied empirically: $\{x_i\}$ is UDM1 $\iff$ for every integer $h \neq 0$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} e^{2\pi i h x_i} = 0$$

  Note: if $\{x_i\}$ are random then $\sum_{i=1}^{N} e^{2\pi i h x_i} \approx O(\sqrt{N})$

- Discrepancy - the number of points in the sequence falling into an arbitrary set B is close to proportional to the measure of B
- There exists an absolute constant C such that for any positive integer m the discrepancy of any sequence $\{x_i\}$ satisfies

$$D_N < C\left(\frac{1}{m}\sum_{h=0}^{m}\frac{1}{h}\left|\frac{1}{N}\sum_{i=0}^{N-1} e^{2\pi i h x_i}\right|\right)$$

Equidistribution

- Equivalently the Weyl criterion can be applied empirically: $\{x_i\}$ is UDM1 $\iff$ for every integer $h \neq 0$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} e^{2\pi i h x_i} = 0$$

  Note: if $\{x_i\}$ are random then $\sum_{i=1}^{N} e^{2\pi i h x_i} \approx O(\sqrt{N})$

- Discrepancy - the number of points in the sequence falling into an arbitrary set B is close to proportional to the measure of B
- There exists an absolute constant C such that for any positive integer m the discrepancy of any sequence $\{x_i\}$ satisfies

$$D_N < C\left(\frac{1}{m}\sum_{h=0}^{m}\frac{1}{h}\left|\frac{1}{N}\sum_{i=0}^{N-1} e^{2\pi i h x_i}\right|\right)$$

Equidistribution

- Let $\{\mathbf{x}_i\}$ be an infinite sequence of vectors in $[0,1)^k$; $E \subset [0,1)^k$, a subset, and $\#(E;N)$ the number of $\{\mathbf{x}_i\}$, $1 \le i \le N \in E$, then $\{\mathbf{x}_i\}$ is UDM1 in $\mathbb{R}^k$ if:

$$\lim_{N\to\infty} \frac{\#([\mathbf{a},\mathbf{b});N)}{N} = \prod_{i=1}^{k}(b_i - a_i), \quad \forall [\mathbf{a},\mathbf{b}) \in \mathbb{R}^k$$

- if $1, \alpha_1, \ldots \alpha_k$ are linearly independent over the rationals, then $\mathbf{x}_i = (\{i\alpha_1\}, \ldots, \{i\alpha_k\})$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is k-distributed modulo 1 if $\mathbf{x}_i = (x_i, x_{i+1}, \ldots x_{i+k-1}) \in \mathbb{R}^k$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is ∞-distributed modulo 1 if it is UDM1 in $\mathbb{R}^k$, $\forall k > 0$
- If α is absolutely normal, then its digits can be used to approximate an ∞-distributed sequence modulo 1

Equidistribution

- Let $\{\mathbf{x}_i\}$ be an infinite sequence of vectors in $[0,1)^k$; $E \subset [0,1)^k$, a subset, and $\#(E;N)$ the number of $\{\mathbf{x}_i\}$, $1 \leq i \leq N \in E$, then $\{\mathbf{x}_i\}$ is UDM1 in $\mathbb{R}^k$ if:

$$\lim_{N\to\infty} \frac{\#([\mathbf{a},\mathbf{b});N)}{N} = \prod_{i=1}^{k}(b_i - a_i), \quad \forall[\mathbf{a},\mathbf{b}) \in \mathbb{R}^k$$

- if $1, \alpha_1, \ldots \alpha_k$ are linearly independent over the rationals, then $\mathbf{x}_i = (\{i\alpha_1\}, \ldots, \{i\alpha_k\})$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is k-distributed modulo 1 if $\mathbf{x}_i = (x_i, x_{i+1}, \ldots x_{i+k-1}) \in \mathbb{R}^k$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is ∞-distributed modulo 1 if it is UDM1 in $\mathbb{R}^k$, $\forall k > 0$
- If α is absolutely normal, then its digits can be used to approximate an ∞-distributed sequence modulo 1

Equidistribution

- Let $\{\mathbf{x}_i\}$ be an infinite sequence of vectors in $[0,1)^k$; $E \subset [0,1)^k$, a subset, and $\#(E;N)$ the number of $\{\mathbf{x}_i\}$, $1 \leq i \leq N \in E$, then $\{\mathbf{x}_i\}$ is UDM1 in $\mathbb{R}^k$ if:

$$\lim_{N\to\infty} \frac{\#([\mathbf{a},\mathbf{b});N)}{N} = \prod_{i=1}^{k}(b_i - a_i), \quad \forall[\mathbf{a},\mathbf{b}) \in \mathbb{R}^k$$

- if $1, \alpha_1, \ldots \alpha_k$ are linearly independent over the rationals, then $\mathbf{x}_i = (\{i\alpha_1\}, \ldots, \{i\alpha_k\})$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is k-distributed modulo 1 if $\mathbf{x}_i = (x_i, x_{i+1}, \ldots x_{i+k-1}) \in \mathbb{R}^k$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is ∞-distributed modulo 1 if it is UDM1 in $\mathbb{R}^k$, $\forall k > 0$
- If α is absolutely normal, then its digits can be used to approximate an ∞-distributed sequence modulo 1

Equidistribution

- Let $\{\mathbf{x}_i\}$ be an infinite sequence of vectors in $[0,1)^k$; $E \subset [0,1)^k$, a subset, and $\#(E;N)$ the number of $\{\mathbf{x}_i\}$, $1 \leq i \leq N \in E$, then $\{\mathbf{x}_i\}$ is UDM1 in $\mathbb{R}^k$ if:

$$\lim_{N\to\infty} \frac{\#([\mathbf{a},\mathbf{b});N)}{N} = \prod_{i=1}^{k}(b_i - a_i), \quad \forall[\mathbf{a},\mathbf{b}) \in \mathbb{R}^k$$

- if $1, \alpha_1, \dots \alpha_k$ are linearly independent over the rationals, then $\mathbf{x}_i = (\{i\alpha_1\}, \dots, \{i\alpha_k\})$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is k-distributed modulo 1 if $\mathbf{x}_i = (x_i, x_{i+1}, \dots x_{i+k-1}) \in \mathbb{R}^k$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is ∞-distributed modulo 1 if it is UDM1 in $\mathbb{R}^k$, $\forall k > 0$
- If α is absolutely normal, then its digits can be used to approximate an ∞-distributed sequence modulo 1

Equidistribution

- Let $\{\mathbf{x}_i\}$ be an infinite sequence of vectors in $[0,1)^k$; $E \subset [0,1)^k$, a subset, and $\#(E;N)$ the number of $\{\mathbf{x}_i\}$, $1 \le i \le N \in E$, then $\{\mathbf{x}_i\}$ is UDM1 in $\mathbb{R}^k$ if:

$$\lim_{N\to\infty} \frac{\#([\mathbf{a},\mathbf{b});N)}{N} = \prod_{i=1}^{k}(b_i - a_i), \quad \forall[\mathbf{a},\mathbf{b}) \in \mathbb{R}^k$$

- if $1, \alpha_1, \ldots \alpha_k$ are linearly independent over the rationals, then $\mathbf{x}_i = (\{i\alpha_1\}, \ldots, \{i\alpha_k\})$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is k-distributed modulo 1 if $\mathbf{x}_i = (x_i, x_{i+1}, \ldots x_{i+k-1}) \in \mathbb{R}^k$ is UDM1 in $\mathbb{R}^k$
- $\{x_i\} \in \mathbb{R}$ is ∞-distributed modulo 1 if it is UDM1 in $\mathbb{R}^k$, $\forall k > 0$
- If α is absolutely normal, then its digits can be used to approximate an ∞-distributed sequence modulo 1

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n \to \infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{b x_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\,x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\,x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n \to \infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{b x_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{b x_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{bx_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal Numbers and Recursive Sequences

- Associate a real number of the form
  - $\beta = \sum_{i=1}^{\infty} \frac{r_n}{b^n}$, where $\lim_{n\to\infty} r_n = 0$
  - having a PRNG sequence starting at $x_0 = 0$
  - $x_n = \{b x_{n-1} + r_n\}$
  - x_n is then equidistributed $\iff$ β is b-normal.
- Linear Congruential Generator (LCG) with prime additive constant
  - $x_n = a\, x_{n-1} + p \pmod{M}$
  - p is a prime additive constant
  - a is the multiplier
  - M for this generator is 2^{64}

Normal from Sequence

- Consider a recurrence in the form:

$$x_n = \{2x_{n-1} + r_n\}, \text{where } r_n = \begin{cases} \frac{1}{n}, & \text{if } n = 3^k \\ 0, & \text{otherwise} \end{cases}$$

- Thus we have $\beta = \alpha_{2,3}$ in this case
- This leads to a recurrence formula

$$z_n = 2z_{n-1} \pmod{3^j}$$

- Using a binary expansion of a normal sequence means we can then just shift bits.

Normal from Sequence

- Consider a recurrence in the form:

$$x_n = \{2x_{n-1} + r_n\}, \text{where } r_n = \begin{cases} \frac{1}{n}, & \text{if } n = 3^k \\ 0, & \text{otherwise} \end{cases}$$

- Thus we have $\beta = \alpha_{2,3}$ in this case
- This leads to a recurrence formula

$$z_n = 2z_{n-1} \pmod{3^j}$$

- Using a binary expansion of a normal sequence means we can then just shift bits.

Normal from Sequence

- Consider a recurrence in the form:

$$x_n = \{2x_{n-1} + r_n\}, \text{where } r_n = \begin{cases} \frac{1}{n}, & \text{if } n = 3^k \\ 0, & \text{otherwise} \end{cases}$$

- Thus we have $\beta = \alpha_{2,3}$ in this case
- This leads to a recurrence formula

$$z_n = 2z_{n-1} \pmod{3^j}$$

- Using a binary expansion of a normal sequence means we can then just shift bits.

Normal from Sequence

- Consider a recurrence in the form:

$$x_n = \{2x_{n-1} + r_n\}, \text{where } r_n = \begin{cases} \frac{1}{n}, & \text{if } n = 3^k \\ 0, & \text{otherwise} \end{cases}$$

- Thus we have $\beta = \alpha_{2,3}$ in this case
- This leads to a recurrence formula

$$z_n = 2z_{n-1} \pmod{3^j}$$

- Using a binary expansion of a normal sequence means we can then just shift bits.

Getting a Generator from a Stoneham Number

- Recall that $\alpha_{2,3} = \sum_{k \geq 0} \frac{1}{2^{3^k} 3^k}$
- The digits starting at bit 2^{3^m} is $x_{3^m} = \{2^{3^m} \alpha_{2,3}\}$ which can be rewritten as

$$x_{3^m} = \{\sum_{k=1}^{m} \frac{2^{3^m - 3^k}}{3^k}\} + \sum_{k=m+1}^{\infty} \frac{2^{3^m - 3^k}}{3^k}$$

- The second summation is extremely small even when m is not large, call it ϵ_m, thus

$$x_{3^m} = \frac{(3^{m-1} 2^{3^m - 3} + 3^{m-2} 2^{3^m - 3^2} + \ldots + 3 \times 2^{3^m - 3^{m-1}} + 1) \pmod{3^m}}{3^m} + \epsilon_m$$

Getting a Generator from a Stoneham Number

- Recall that $\alpha_{2,3} = \sum_{k\geq 0} \frac{1}{2^{3^k} 3^k}$
- The digits starting at bit 2^{3^m} is $x_{3^m} = \{2^{3^m} \alpha_{2,3}\}$ which can be rewritten as

$$x_{3^m} = \{\sum_{k=1}^{m} \frac{2^{3^m - 3^k}}{3^k}\} + \sum_{k=m+1}^{\infty} \frac{2^{3^m - 3^k}}{3^k}$$

- The second summation is extremely small even when m is not large, call it ϵ_m, thus

$$x_{3^m} = \frac{(3^{m-1} 2^{3^m - 3} + 3^{m-2} 2^{3^m - 3^2} + \ldots + 3 \times 2^{3^m - 3^{m-1}} + 1) \pmod{3^m}}{3^m} + \epsilon_m$$

Getting a Generator from a Stoneham Number

- Recall that $\alpha_{2,3} = \sum_{k\geq 0} \frac{1}{2^{3^k} 3^k}$
- The digits starting at bit 2^{3^m} is $x_{3^m} = \{2^{3^m}\alpha_{2,3}\}$ which can be rewritten as

$$x_{3^m} = \{\sum_{k=1}^{m} \frac{2^{3^m - 3^k}}{3^k}\} + \sum_{k=m+1}^{\infty} \frac{2^{3^m - 3^k}}{3^k}$$

- The second summation is extremely small even when m is not large, call it ϵ_m, thus

$$x_{3^m} = \frac{(3^{m-1}2^{3^m-3} + 3^{m-2}2^{3^m-3^2} + \ldots + 3 \times 2^{3^m-3^{m-1}} + 1) \pmod{3^m}}{3^m}$$

$$+\epsilon_m$$

Getting a Generator from a Stoneham Number

- $2^{3^m - 3^{m-1}} \equiv 1 \pmod{3^m}$, and similarly for the other terms (proof on the next slide), and so

$$\frac{(3^{m-1} + 3^{m-2} + \ldots + 3 + 1) \pmod{3^m}}{3^m} + \epsilon_m$$

$$= \frac{3^m - 1}{2 \times 3^m} + \epsilon_m = \frac{\lfloor 3^m/2 \rfloor}{3^m} + \epsilon_m$$

- And finally we have

$$x_n = \frac{(2^{n-3^m} \lfloor 3^m/2 \rfloor) \pmod{3^m}}{3^m} + \epsilon$$

- If we choose $m = 33$ then we derive the start of a sequence where 52 bits can be generated at a time using double precision arithmetic, $3^{33} \approx 2^{52}$

Getting a Generator from a Stoneham Number

- $2^{3^m - 3^{m-1}} \equiv 1 \pmod{3^m}$, and similarly for the other terms (proof on the next slide), and so

$$\frac{(3^{m-1} + 3^{m-2} + \ldots + 3 + 1) \pmod{3^m}}{3^m} + \epsilon_m$$

$$= \frac{3^m - 1}{2 \times 3^m} + \epsilon_m = \frac{\lfloor 3^m/2 \rfloor}{3^m} + \epsilon_m$$

- And finally we have

$$x_n = \frac{(2^{n-3^m} \lfloor 3^m/2 \rfloor) \pmod{3^m}}{3^m} + \epsilon$$

- If we choose $m = 33$ then we derive the start of a sequence where 52 bits can be generated at a time using double precision arithmetic, $3^{33} \approx 2^{52}$

Getting a Generator from a Stoneham Number

- $2^{3^m - 3^{m-1}} \equiv 1 \pmod{3^m}$, and similarly for the other terms (proof on the next slide), and so

$$\frac{(3^{m-1} + 3^{m-2} + \ldots + 3 + 1) \pmod{3^m}}{3^m} + \epsilon_m$$

$$= \frac{3^m - 1}{2 \times 3^m} + \epsilon_m = \frac{\lfloor 3^m/2 \rfloor}{3^m} + \epsilon_m$$

- And finally we have

$$x_n = \frac{(2^{n-3^m} \lfloor 3^m/2 \rfloor) \pmod{3^m}}{3^m} + \epsilon$$

- If we choose $m = 33$ then we derive the start of a sequence where 52 bits can be generated at a time using double precision arithmetic, $3^{33} \approx 2^{52}$

Proof of $3^{m-k} * 2^{3^m-3^k} = 3^{m-k} \pmod{3^m}$

- Proof of the main result:

$$3^{m-k} * 2^{3^m-3^k} = 3^{m-k} \pmod{3^m} \text{ for } 1 \le k \le m$$

1. By Euler's generalization of little Fermat, $2^{2*3^{k-1}} = 1 \pmod{3^k}$ for any $k \ge 1$, note that $\phi(3^k) = 3^k * (1 - 1/3) = 3^{k-1} * (3-1) = 2 * 3^{k-1}$
2. And so for some integer M depending only on k and m, $1 \le k \le m$ we have
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 3^k * M$$
3. It follows that
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 2^{3^{k-1}*(3^{m-k+1}-3)} - 1 =$$
$$2^{3^m-3^k} - 1 = 3^k * M \text{ for } 1 \le k \le m$$
4. Multiply both sides of this last equation by 3^{m-k} to get
$$3^{m-k} * 2^{3^m-3^k} = 3^{m-k} \pmod{3^m} \blacksquare$$

Proof of $3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m}$

- Proof of the main result:

$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \text{ for } 1 \le k \le m$$

1. By Euler's generalization of little Fermat, $2^{2*3^{k-1}} = 1 \pmod{3^k}$ for any $k \ge 1$, note that $\phi(3^k) = 3^k * (1 - 1/3) = 3^{k-1} * (3 - 1) = 2 * 3^{k-1}$
2. And so for some integer M depending only on k and m, $1 \le k \le m$ we have
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 3^k * M$$
3. It follows that
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 2^{3^{k-1}*(3^{m-k+1}-3)} - 1 =$$
$$2^{3^m - 3^k} - 1 = 3^k * M \text{ for } 1 \le k \le m$$
4. Multiply both sides of this last equation by 3^{m-k} to get
$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \blacksquare$$

Proof of $3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m}$

- Proof of the main result:

$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \text{ for } 1 \le k \le m$$

1. By Euler's generalization of little Fermat, $2^{2*3^{k-1}} = 1 \pmod{3^k}$ for any $k \ge 1$, note that $\phi(3^k) = 3^k * (1 - 1/3) = 3^{k-1} * (3 - 1) = 2 * 3^{k-1}$
2. And so for some integer M depending only on k and $m, 1 \le k \le m$ we have
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 3^k * M$$
3. It follows that
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 2^{3^{k-1}*(3^{m-k+1}-3)} - 1 =$$
$$2^{3^m - 3^k} - 1 = 3^k * M \text{ for } 1 \le k \le m$$
4. Multiply both sides of this last equation by 3^{m-k} to get
$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \blacksquare$$

Proof of $3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m}$

- Proof of the main result:

$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \text{ for } 1 \leq k \leq m$$

1. By Euler's generalization of little Fermat, $2^{2*3^{k-1}} = 1 \pmod{3^k}$ for any $k \geq 1$, note that $\phi(3^k) = 3^k * (1 - 1/3) = 3^{k-1} * (3 - 1) = 2 * 3^{k-1}$
2. And so for some integer M depending only on k and m, $1 \leq k \leq m$ we have
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 3^k * M$$
3. It follows that
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 2^{3^{k-1}*(3^{m-k+1}-3)} - 1 =$$
$$2^{3^m - 3^k} - 1 = 3^k * M \text{ for } 1 \leq k \leq m$$
4. Multiply both sides of this last equation by 3^{m-k} to get
$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \blacksquare$$

NIST National Institute of Standards and Technology U.S. Department of Commerce

Proof of $3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m}$

- Proof of the main result:

$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \text{ for } 1 \leq k \leq m$$

1. By Euler's generalization of little Fermat, $2^{2*3^{k-1}} = 1 \pmod{3^k}$ for any $k \geq 1$, note that
$\phi(3^k) = 3^k * (1 - 1/3) = 3^{k-1} * (3 - 1) = 2 * 3^{k-1}$
2. And so for some integer M depending only on k and $m, 1 \leq k \leq m$ we have
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 3^k * M$$
3. It follows that
$$2^{2*3^{k-1}*3*(3^{m-k}-1)/2} - 1 = 2^{3^{k-1}*(3^{m-k+1}-3)} - 1 =$$
$$2^{3^m - 3^k} - 1 = 3^k * M \text{ for } 1 \leq k \leq m$$
4. Multiply both sides of this last equation by 3^{m-k} to get
$$3^{m-k} * 2^{3^m - 3^k} = 3^{m-k} \pmod{3^m} \blacksquare$$

Mathematical Model: Specific Constants

- Specific generator form of a Normal Constant

$$\alpha_{2,3} = \sum_{k>1} \frac{1}{3^k 2^{3^k}}$$

- This sum produces numbers of the form
  = 0.0418836808315029850712528986245716824260967584654857 ... 10
  = 0.0AB8E38F684BDA12F684BF35BA781948B0FCD6E9E06522C3F35B ... 16

Mathematical Model: Specific Constants

- Specific generator form of a Normal Constant

$$\alpha_{2,3} = \sum_{k>1} \frac{1}{3^k 2^{3^k}}$$

- This sum produces numbers of the form
  = 0.0418836808315029850712528986245716824260967584654857 ... 10
  = 0.0AB8E38F684BDA12F684BF35BA781948B0FCD6E9E06522C3F35B ... 16

Source Code

Source Code

- Seed Generation
- Calculation Code Initial
- Calculation Code Iteration

Source Code

Source Code

- Seed Generation
- Calculation Code Initial
- Calculation Code Iteration

Source Code

Source Code

- Seed Generation
- Calculation Code Initial
- Calculation Code Iteration

Seed Generation

Initialization

- Select a starting index a in the range

$$3^{33}+100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$$

- 'a' can be thought of as the 'seed' of the generator
- Calculate the first value

$$z_0 = (2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor) \pmod{3^{33}}$$

- To return in the unit interval, multiply by 3^{-33}

Generate Iterates

- The next values can be calculated by the recursion

$$z_k = (2^{53} \cdot z_{k-1}) \pmod{3^{33}}$$

Seed Generation

Initialization

- Select a starting index a in the range

$$3^{33}+100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$$

- 'a' can be thought of as the 'seed' of the generator
- Calculate the first value

$$z_0 = (2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor) \pmod{3^{33}}$$

- To return in the unit interval, multiply by 3^{-33}

Generate Iterates

- The next values can be calculated by the recursion

$$z_k = (2^{53} \cdot z_{k-1}) \pmod{3^{33}}$$

Seed Generation

Initialization

- Select a starting index a in the range

$$3^{33}+100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$$

- 'a' can be thought of as the 'seed' of the generator
- Calculate the first value

$$z_0 = (2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor) \pmod{3^{33}}$$

- To return in the unit interval, multiply by 3^{-33}

Generate Iterates

- The next values can be calculated by the recursion

$$z_k = (2^{53} \cdot z_{k-1}) \pmod{3^{33}}$$

Seed Generation

Initialization

- Select a starting index a in the range

$$3^{33}+100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$$

- 'a' can be thought of as the 'seed' of the generator
- Calculate the first value

$$z_0 = (2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor) \pmod{3^{33}}$$

- To return in the unit interval, multiply by 3^{-33}

Generate Iterates

- The next values can be calculated by the recursion

$$z_k = (2^{53} \cdot z_{k-1}) \pmod{3^{33}}$$

Seed Generation

Initialization

- Select a starting index a in the range

$$3^{33}+100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$$

- 'a' can be thought of as the 'seed' of the generator
- Calculate the first value

$$z_0 = (2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor) \pmod{3^{33}}$$

- To return in the unit interval, multiply by 3^{-33}

Generate Iterates

- The next values can be calculated by the recursion

$$z_k = (2^{53} \cdot z_{k-1}) \pmod{3^{33}}$$

Calculation Code Initial

```
// define some constants
    p3i = pow(3.0, 33.0);
    r3i = 1.0 / p3i;
    t53 = pow(2.0, 53.0);
// Calculate starting element.
d2 = expm2 (aa - p3i, p3i);
d3 = aint (0.50 * p3i);
ddmuldd (d2, d3, dd1);
d1 = aint (r3i * dd1[0]);
ddmuldd (d1, p3i, dd2);
ddsub (dd1, dd2, dd3);
d1 = dd3[0];
if(d1 < 0.0)
        d1 = d1 + p3i;
```

Calculation Code Iteration

```
dd1[0] = t53 * d1;
dd1[1] = 0.0;
d2 = aint (t53 * d1 / p3i);
ddmuldd (p3i, d2, dd2);
ddsub (dd1, dd2, dd3);
d1 = dd3[0];
if (d1 < 0.0)
        d1 = d1 + p3i;
```

Implementation and Results

- First implement in TestU01
- TestU01 results
- Implementation in SPRNG
- SPRNG results

Implementation and Results

- First implement in TestU01
- TestU01 results
- Implementation in `SPRNG`
- `SPRNG` results

Implementation and Results

- First implement in TestU01
- TestU01 results
- Implementation in `SPRNG`
- `SPRNG` results

Implementation and Results

- First implement in TestU01
- TestU01 results
- Implementation in `SPRNG`
- `SPRNG` results

TestU01 Results

```
SmallCrush
==============================================
Version: TestU01 1.2.1
seed = 5559060566555623

========= Summary results of SmallCrush =========

Version: TestU01 1.2.1
Generator: ubcn_CreateBCNf
Number of statistics: 15
Total CPU time: 00:01:01.67
The following tests gave p-values outside [0.001, 0.9990]:
(eps means a value < 1.0e-300):
(eps1 means a value < 1.0e-15):

Test p-value
==============================================
1 BirthdaySpacings eps
==============================================
All other tests were passed
```

Table: TestU01 Results - SmallCrush

Implementation in SPRNG

- Class Definition
- Initialization Routine
- Iterations

Implementation in SPRNG

- Class Definition
- Initialization Routine
- Iterations

Implementation in SPRNG

- Class Definition
- Initialization Routine
- Iterations

Class Definition

- class BCN : public Sprng
- #define NPARAMS 1 /*** number of valid parameters ***/

```
Sprng * SelectType(int typenum)
{
  switch (typenum)
    case 0: return new LFG;
    case 1: return new LCG;
    case 2: return new LCG64;
    case 3: return new CMRG;
    case 4: return new MLFG;
    case 5: return new PMLCG;
    case 6: return new BCN;
```

Initialization Routine

- Calculate constants

```
BCN::BCN()  /* default constructor */
{
p3i = long long(pow(3.0, 33.0));   // $3^{33}$
r3i = 1.0 / p3i;                   // $\frac{1}{3^{33}}$
t53 = long long(pow(2.0, 53.0));   // $2^{53}$
```

- Calculate Initial Value

```
int BCN::init_rng(int gn, int tg, int seed, int p)
```

Iterations

- Single function to get next random number
- Different versions to return different types

```
int get_rn_int ();                /* returns integer */
long long get_rn_int64 ();        /* returns integer */
float get_rn_flt ();              /* returns float */
double get_rn_dbl ();             /* returns double */
```

SPRNG Timing Results

Type	Integer	Float	Double
lcg	125.0156 MRS	125.0156 MRS	142.8776 MRS
lfg	66.6756 MRS	62.5117 MRS	52.6399 MRS
mlfg	166.6944 MRS	100.0100 MRS	142.8776 MRS
lcg64	142.8776 MRS	62.5078 MRS	71.4388 MRS
cmrg	111.1235 MRS	47.6259 MRS	58.8339 MRS
bcn	23.2591 MRS	22.2257 MRS	22.7309 MRS

Table: Timing C++ interface: (Note: MRS = Million Random Numbers Per Second)

How well does it work in Practice?

- Monte Carlo autocorrelation
- Typical good generator autocorrelation time
- Normal Number generator autocorrelation time
- Bad LCG generator autocorrelation time

How well does it work in Practice?

- Monte Carlo autocorrelation
- Typical good generator autocorrelation time
- Normal Number generator autocorrelation time
- Bad LCG generator autocorrelation time

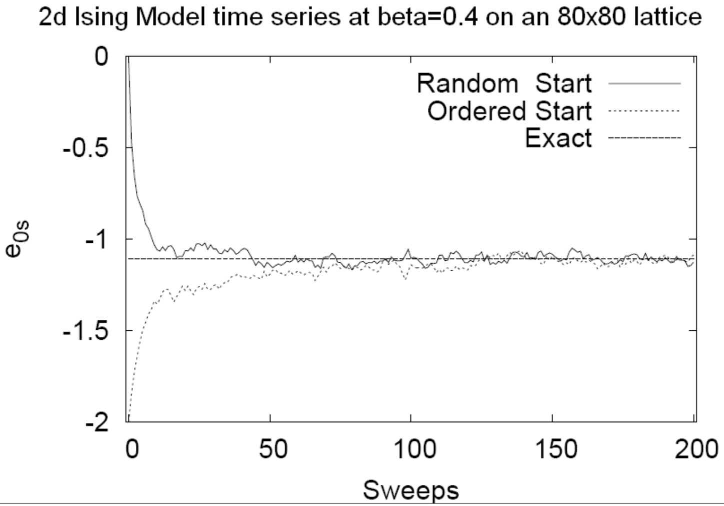
How well does it work in Practice?

- Monte Carlo autocorrelation
- Typical good generator autocorrelation time
- Normal Number generator autocorrelation time
- Bad LCG generator autocorrelation time

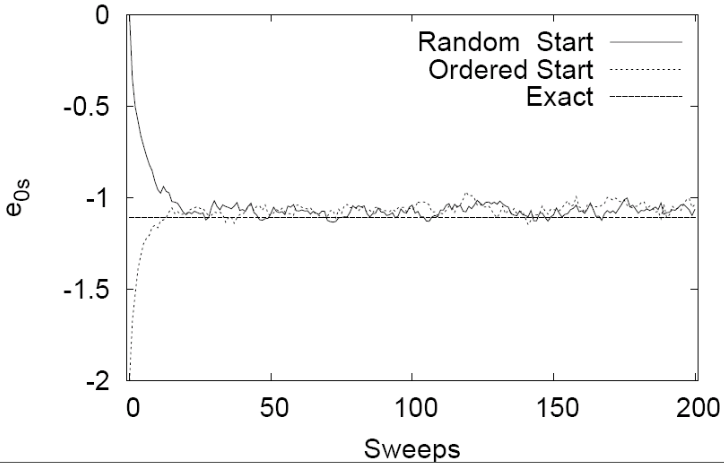
How well does it work in Practice?

- Monte Carlo autocorrelation
- Typical good generator autocorrelation time
- Normal Number generator autocorrelation time
- Bad LCG generator autocorrelation time

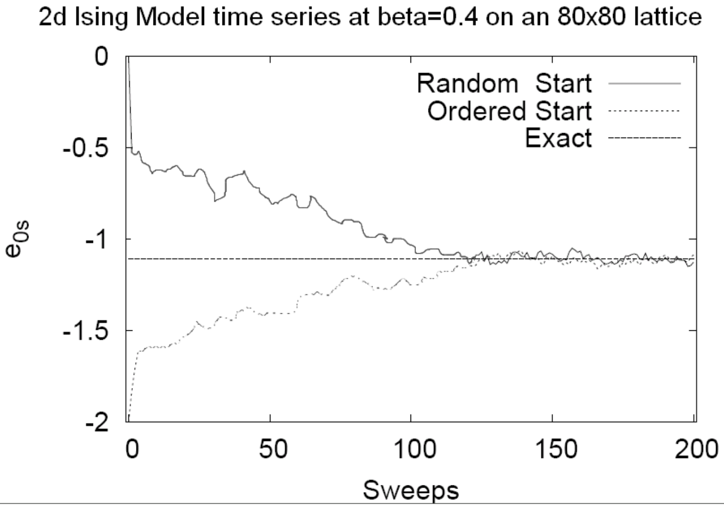
Typical good generator autocorrelation time



Normal Number generator autocorrelation time



Typical bad generator autocorrelation time



Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Conclusions and Future Work

- The Random Number Generator seems to work very well by passing all the tests, except one.
- The Normal Number generator runs a bit slower than the other generators.
- Future Options
- The b and c constants chosen can be changed as long as they are co-prime.
- The speed needs to be improved.
- Implement on GPU

Questions?

Bibliography

[D. H. Bailey (2004)]
A Pseudo-Random Number Generator Based on Normal Numbers,
http://crd.lbl.gov/∼dhbailey/dhbpapers/normal-random.pdf, 8 pages.

[D. H. Bailey and R. E. Crandall (2004)]
Random Generators and Normal Numbers,
Experimental Mathematics, **11(4)**: 527–546.

[D. H. Bailey and D. J. Rudolph (2002)]
An Ergodic Proof that Rational Times Normal is Normal,
http://www.nersc.gov/∼dhbailey/dhbpapers/ratxnormal.pdf, 2 pages.

[S. F. Brailsford and M. Mascagni and D. H. Bailey (2014)]
Normal Numbers as Efficient Sources of Pseudorandom Digits,
in preparation from SFB's MS Thesis, `SPRNG` *Gets a Normal Number Generator*

Bibliography

[D. H. Bailey (2004)]
A Pseudo-Random Number Generator Based on Normal Numbers,
http://crd.lbl.gov/∼dhbailey/dhbpapers/normal-random.pdf, 8 pages.

[D. H. Bailey and R. E. Crandall (2004)]
Random Generators and Normal Numbers,
Experimental Mathematics, **11(4)**: 527–546.

[D. H. Bailey and D. J. Rudolph (2002)]
An Ergodic Proof that Rational Times Normal is Normal,
http://www.nersc.gov/∼dhbailey/dhbpapers/ratxnormal.pdf, 2 pages.

[S. F. Brailsford and M. Mascagni and D. H. Bailey (2014)]
Normal Numbers as Efficient Sources of Pseudorandom Digits,
in preparation from SFB's MS Thesis, SPRNG *Gets a Normal Number Generator*

Bibliography

[D. H. Bailey (2004)]
A Pseudo-Random Number Generator Based on Normal Numbers,
http://crd.lbl.gov/∼dhbailey/dhbpapers/normal-random.pdf, 8 pages.

[D. H. Bailey and R. E. Crandall (2004)]
Random Generators and Normal Numbers,
Experimental Mathematics, **11(4)**: 527–546.

[D. H. Bailey and D. J. Rudolph (2002)]
An Ergodic Proof that Rational Times Normal is Normal,
http://www.nersc.gov/∼dhbailey/dhbpapers/ratxnormal.pdf, 2 pages.

[S. F. Brailsford and M. Mascagni and D. H. Bailey (2014)]
Normal Numbers as Efficient Sources of Pseudorandom Digits,
in preparation from SFB's MS Thesis, SPRNG *Gets a Normal Number Generator*

Bibliography

[D. H. Bailey (2004)]
A Pseudo-Random Number Generator Based on Normal Numbers,
http://crd.lbl.gov/∼dhbailey/dhbpapers/normal-random.pdf, 8 pages.

[D. H. Bailey and R. E. Crandall (2004)]
Random Generators and Normal Numbers,
Experimental Mathematics, **11(4)**: 527–546.

[D. H. Bailey and D. J. Rudolph (2002)]
An Ergodic Proof that Rational Times Normal is Normal,
http://www.nersc.gov/∼dhbailey/dhbpapers/ratxnormal.pdf, 2 pages.

[S. F. Brailsford and M. Mascagni and D. H. Bailey (2014)]
Normal Numbers as Efficient Sources of Pseudorandom Digits,
in preparation from SFB's MS Thesis, SPRNG *Gets a Normal Number Generator*