Entwicklung einer Klassenbibliothek zur Erzeugung autokorrelierter Zufallszahlen

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

|  |
| --- |
| Herbstsemester 2017 |

Autor(en): Anthony Delay

Philipp Bütikofer

Betreuer: Prof. Dr. Andreas Rinkel

Lukas Kretschmar

# Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Version** | **Änderung** | **Autor** |
| 03.10.2017 | 1.0 | Eröffnung des Dokuments, Gliederung der Themen | AD, PB |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Inhalt

[Änderungsgeschichte 2](#_Toc496027351)

[Inhalt 3](#_Toc496027352)

[1. Abstract [bis 22.12.2017] 4](#_Toc496027353)

[2. Einführung und Motivation [bis 18.10.2017] 4](#_Toc496027354)

[3. Zugrundeliegende Arbeiten [bis 18.10.2017] 4](#_Toc496027355)

[3.1 Autoregressive-To-Anything Process [bis 18.10.2017] 4](#_Toc496027356)

[3.2 JARTA [bis 18.10.2017] 4](#_Toc496027357)

[4. Autokorrelation [bis 25.10.2017] 4](#_Toc496027358)

[4.1 Definition 4](#_Toc496027359)

[4.2 Nachweis von Autokorrelation 5](#_Toc496027360)

[4.3 Anwendungsbereiche 5](#_Toc496027361)

[4.4 Beispiel 5](#_Toc496027362)

[5. ARTA.Core [bis 8.11.2017] 6](#_Toc496027363)

[5.1 Zufallszahlen und Autokorrelation 6](#_Toc496027364)

[5.2 Implementation 6](#_Toc496027365)

[5.3 Statistische Tests 6](#_Toc496027366)

[5.4 Integration Simio 6](#_Toc496027367)

[6. Simulation und Auswertung [[bis 25.11.2017] 6](#_Toc496027368)

[6.1 Simulationsumgebung 6](#_Toc496027369)

[6.2 Resultate 6](#_Toc496027370)

[7. Fazit und Ausblick [bis 20.12.2017] 6](#_Toc496027371)

[8. Literaturverzeichnis und Referenzen 6](#_Toc496027372)

# Abstract [bis 20.12.2017]

# Einführung und Motivation [bis 18.10.2017]

In der Simulation von Systemen werden Zufallszahlen zur Beschreibung die einzelnen Arbeitsschritte benötigt. Standardmässig werden diese Zufallszahlen so erzeugt, dass sie keine Abhängigkeiten beziehungsweise Autokorrelationen aufweisen.

Die Realität sieht jedoch anders aus. Es hat sich gezeigt, dass in der Praxis häufig ebendiese Autokorrelationen auftreten. Aufgrund dieser Abhängigkeiten können simulierte und reale Ergebnisse stark voneinander abweichen. Im Rahmen der Studienarbeit HS2017/18 soll eine Klassenbibliothek (ARTA.Core) entwickelt werden, welche es ermöglicht, autokorrelierte Zufallszahlen zu erzeugen. Der Grad der Autokorrelation kann selbst definiert werden. ARTA.Core soll so implementiert und erweitert werden, dass eine Einbindung in die Simulationssoftware Simio möglich ist.

# Zugrundeliegende Arbeiten [bis 18.10.2017]

Als Fundament für die vorliegende Studienarbeit gelten die beiden Dokumente «Autoregressive to anything: Time-series input processes for simulation» und «JARTA — A Java library to model and fit Autoregressive-To-Anything processes».

Die erst genannte Publikation beschreibt den ARTA-Prozess auf der mathematischen Ebene, die zweite stellt eine Java Implementation vor, welche den ARTA-Prozess abbildet. Für die vorliegende Arbeit wird auf Basis von JARTA eine neue Klassenbibliothek, «ARTA.Core», erzeugt, welche den ARTA-Prozess und die Integration in die Simulationssoftware Simio implementiert.

## Autoregressive-To-Anything Process [bis 18.10.2017]

ARTA (Autoregressive to anything) präsentiert ein Modell,

## JARTA [bis 18.10.2017]

# Autokorrelation [bis 25.10.2017]

Dieser Abschnitt soll den Begriff der Autokorrelation einfangen und deren grundlegende Eigenschaften und Charakteristiken aufzeigen. Anschliessend wird auf die Bereiche, welche Autokorrelation aufweisen eingegangen. Um den Themenbereich abzuschliessen wird Autokorrelation anhand eines Beispiels aufgezeigt.

## Definition

Teilt man den Begriff der Autokorrelation in seine beiden Wortstämme auf, so kann man anhand dieser auf dessen Bedeutung schliessen.

Der Wortteil Korrelation beschreibt dabei eine Beziehung bzw. Zusammenhang zwischen mindestens zwei oder mehr Merkmalen, Zuständen, Funktionen oder Ereignissen. Diese Merkmale können sich je nach Anwendungsgebiet sehr stark unterscheiden.

Das Präfix «Auto» zeigt auf, dass die Funktion oder Reihe mit sich selbst korreliert, was bedeutet, dass ähnliche oder gleiche Muster erkennbar sind. Bei Autokorrelation sind also die Werte einer Variable zum Zeitpunkt t mit den Werten derselben Variable in zeitlich vergangenen Perioden korreliert. Die Autokorrelation ist immer zeitabhängig. Der Zusammenhang zwischen Autokorrelation und Zeit kann in Form von Korrelationsfunktionen ausgedrückt werden. Eine Korrelationsfunktion zeigt an, wie viel Ähnlichkeit zwischen der ursprünglichen und der, um eine Zeit t, verschobenen Folge besteht.

Grundsätzlich gilt die Aussage, «Korrelation gilt als Mass eines Zusammenhangs». Dieses Mass kann numerisch in Form von Korrelationskoeffizienten ausgedrückt werden und beantwortet die Frage nach der Stärke und der Richtung des Zusammenhangs. Bei Korrelationskoeffizienten handelt es sich um Zahlen, welche in einem Intervall zwischen -1 und 1 liegen. Eine Korrelation die den Koeffizienten 1 aufweist wird als perfekte positive, bei -1 als perfekte negative Korrelation bezeichnet. Je weiter sich der Korrelationskoeffizient dem Wert 0 nähert, umso weniger bzw. schwächer ist die Korrelation.



Abbildung 1: Korrelationskoeffizient

Im Zusammenhang mit dem Begriff Korrelationskoeffizient taucht der Ausdruck Pearson-Korrelation auf. Dieser ist nach Karl Person benannt, welcher das Mass der Korrelation in Form des Korrelationskoeffizienten in Zusammenarbeit mit Auguste Bravais entwickelt hat. Dies ist daher speziell erwähnenswert, da auf den Algorithmus von Pearson innerhalb der in dieser Arbeit erzeugten Klassenbibliothek zurückgegriffen wird. Weitere Informationen zum Gebrauch und Implementation befinden sich in Abschnitt «ARTA.Core [bis 8.11.2017]».

Autokorrelation kann durch mathematische Formeln ausgedrückt werden, jedoch wird sie in jedem Anwendungsbereich unterschiedlich, spezifisch definiert.

## Nachweis von Autokorrelation

Die gebräuchlichste Methode um die Existenz von Autokorrelation zu belegen stellt der Durbin-Watson-Test dar. Durch diese Art statistischer Test kann geprüft werden, ob eine Autokorrelation der 1. Ordnung vorliegt.

## Anwendungsbereiche

Autokorrelation kann in verschiedenen Gebieten vorgefunden werden. Als die signifikantesten gelten die Statistik, die Signalanalyse, Informationstheorie und die Softwaretechnik.

**Autokorrelation in der Statistik:**

**Autokorrelation in der Signalanalyse und Bildverarbeitung:** In diesem Anwendungsgebiet wird eine Autokorrelationsfunktion genutzt, um die Korrelation eines Signales mit sich selbst zu unterschiedlichen Zeitverschiebungen eingesetzt. Somit kann beispielsweise der Zusammenhang zwischen Faltung und Autokorrelation aufgezeigt werden. IN der Bildverarbeitung wird die zeitliche Komponente durch eine örtliche ersetzt. Dadurch lässt sich beispielsweise Objekterkennung realisieren.

**Autokorrelation in der Softwaretechnik:** Anwendung findet die Autokorrelation hier im sogenannten Korrelationstest. Dieser beschreibt ein Verfahren, welches die Plausibilität einzelner Parameter einer Funktion und deren Kombinationen überprüft.

## Beispiel

Caesar-Chiffre

Vigenere

# ARTA.Core [bis 8.11.2017]

## Zufallszahlen und Autokorrelation

### Mersenne-Twister

Die Generierung der Zufallszahlen basiert auf dem Algortihmus des Mersenne-Twister, entwickelt von Makoto Matsumoto und Takuji Nishimura, 1997. Der Algorithmus existiert in zwei Varianten, die hier eingesetzte wird MT 19937 genannt.

ARTA.Core implementiert den Mersenne-Twister innerhalb der Klasse MersenneTwister.cs. Im folgenden Abschnitt wird anhand des Codes die Funktionsweise des zugrundeliegenden Algorithmus erklärt.

|  |  |
| --- | --- |
|  |  |
| protected override int Next(int bits)  {  if (this.mti >= 624)  {  int mtNext = mt[0];  for (int k = 0; k < 227; k++)  {  int mtCurr = mtNext;  mtNext = mt[(k + 1)];  int s = mtCurr & int.MinValue | mtNext & int.MaxValue;  this.mt[k] = (mt[(k + 397)] ^ s >> 1 ^ MAG01[(s & 0x1)]);  }  for (int k = 227; k < 623; k++)  {  int mtCurr = mtNext;  mtNext = mt[(k + 1)];  int b = mtCurr & int.MinValue | mtNext & int.MaxValue;  this.mt[k] = (mt[(k + 65309)] ^ b >> 1 ^ MAG01[(b & 0x1)]);  }  int t = mtNext & int.MinValue | mt[0] & int.MaxValue;  mt['?'] = (mt['?'] ^ t >> 1 ^ MAG01[(t & 0x1)]);  mti = 0;  }  int y = this.mt[(mti++)];  y ^= y >> 11;  y = y ^ (y << 7 & - 0x9D2C5680;  y ^= y << 15 & - 0xEFC60000;  y ^= y >> 18;  return y;  } |  |
|  |  |

Als Startwerte gelten die Werte Y1 bis YN, wobei N = 624

### PearsonsCorrelation

### Cholesky Decomposition

### Distributions

## Implementation

## Statistische Tests

## Integration Simio

# Simulation und Auswertung [[bis 25.11.2017]

## Simulationsumgebung

## Resultate

# Fazit und Ausblick [bis 20.12.2017]

# Literaturverzeichnis und Referenzen