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

Inhalt

[1. Abstract [bis 20.12.2017] 4](#_Toc498964598)

[2. Einführung und Motivation [bis 18.10.2017] 4](#_Toc498964599)

[3. Zugrundeliegende Arbeiten [bis 18.10.2017] 4](#_Toc498964600)

[4. Autokorrelation [bis 25.10.2017] 4](#_Toc498964601)

[4.1 Definition 4](#_Toc498964602)

[4.2 Korrelationskoeffizienten 5](#_Toc498964603)

[4.3 Anwendungsbereiche 5](#_Toc498964604)

[4.4 Partielle Korrelation 5](#_Toc498964605)

[4.5 Durbin-Watson-Test 6](#_Toc498964606)

[4.6 Beispiel Autokorrelation 7](#_Toc498964607)

[4.6.1 Beispiel 1 – starke Autokorrelation 8](#_Toc498964608)

[4.6.2 Beispiel 2 10](#_Toc498964609)

[5. Autoregressive to anything [bis 18.11.2017] 12](#_Toc498964610)

[5.1 Zufallszahlen – Mersenne-Twister 12](#_Toc498964611)

[5.2 Zeitreihen / AR-Prozesse 13](#_Toc498964612)

[5.3 Verteilungen 14](#_Toc498964613)

[5.3.1 Normalverteilung 15](#_Toc498964614)

[5.3.2 Exponentialverteilung 16](#_Toc498964615)

[5.3.3 Stetige Gleichverteilung 17](#_Toc498964616)

[5.3.4 Empirische Verteilung 18](#_Toc498964617)

[5.4 ARTA und Autokorrelation [bis 01.11.2017] 18](#_Toc498964618)

[5.4.1 Yule-Walker-Gleichungen 18](#_Toc498964619)

[5.4.2 PearsonsCorrelation [bis 1.11.2017] 19](#_Toc498964620)

[6. ARTA.Standard [bis 15.11.2017] 21](#_Toc498964621)

[6.1 Domain-Modell 21](#_Toc498964622)

[6.2 Implementation 22](#_Toc498964623)

[6.3 Statistische Tests 24](#_Toc498964624)

[6.3.1 Durbin-Watson-Test - Implementation 24](#_Toc498964625)

[6.3.2 ARTAProcess Tests 24](#_Toc498964626)

[6.3.3 Grenzen von ARTA 24](#_Toc498964627)

[6.4 Integration Simio 24](#_Toc498964628)

[7. Test und Auswertung [[bis 25.11.2017] 26](#_Toc498964629)

[7.1 Simulationsumgebung 26](#_Toc498964630)

[7.2 Eigene Simulation 26](#_Toc498964631)

[7.3 Resultate 26](#_Toc498964632)

[8. Anwendungsfall und Simulation [bis 13.12.2017] 26](#_Toc498964633)

[9. Fazit und Ausblick [bis 20.12.2017] 26](#_Toc498964634)

[10. Literaturverzeichnis und Referenzen 27](#_Toc498964635)

[11. Abbildungsverzeichnis 27](#_Toc498964636)

[12. Codefragmente 27](#_Toc498964637)

# Abstract [bis 20.12.2017]

# Einführung und Motivation [bis 18.10.2017]

In der diskreten Ereignis Simulation werden Zufallszahlen zur Beschreibung von Arbeitsschritten und auftretenden Ereignissen benötigt. Standardmässig werden diese Zufallszahlen so erzeugt, dass sie keine Autokorrelationen (Abhängigkeiten) aufweisen.

Die Realität sieht jedoch anders aus[[1]](#footnote-1). 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.Standard) entwickelt werden, welche es ermöglicht, autokorrelierte Zufallszahlen zu erzeugen. Der Grad der Autokorrelation kann selbst definiert werden. ARTA.Standard soll so implementiert werden, dass eine Einbindung in die Simulationssoftware Simio oder andere Simulationstools möglich ist.

# Zugrundeliegende Arbeiten [bis 18.10.2017]

Als Fundament für die vorliegende Studienarbeit dienen die Veröffentlichungen «Autoregressive to anything: Time-series input processes for simulation[[2]](#footnote-2)» und «JARTA — A Java library to model and fit Autoregressive-To-Anything processes[[3]](#footnote-3)».

Die erst genannte Publikation beschreibt den ARTA-Prozess auf der mathematischen Ebene. ARTA (Autoregressive-to-anything) stellt ein bewährtes Modell zur Erzeugung von zufällig generierten Zahlen, mit gegebener Randverteilung und einer Autokorrelation aufweisendem Muster dar. Entwickelt wurde das ARTA-Modells von Marne C. Cario und Barry L. Nelson.

Die zweite Publikation stellt eine Java Implementation vor, welche den ARTA-Prozess abbildet. Mit JARTA werden die Ansätze von ARTA in eine JAVA-Library abgebildet. An einem konkreten Beispiel einer Lagerhaussimulation zeigen Tobias Uhlig und Oliver Rose die Funktionsweise und Wichtigkeit der Abhängigkeiten, wenn es um das Modellieren von stochastischen Prozessen geht. Der Sourcecode von JARTA ist frei verfügbar.

# Autokorrelation [bis 25.10.2017]

Dieser Abschnitt wird den Begriff der Autokorrelation, deren grundlegende Eigenschaften und Charakteristiken erläutern. Anschliessend wird auf die Bereiche, welche Autokorrelation aufweisen eingegangen. Zum Abschluss wird Autokorrelation anhand eines Beispiels aufgezeigt.

## Definition

Autokorrelation setzt sich aus den Wörtern Auto und Korrelation zusammen. «Korrelation» beschreibt einen Zusammenhang zwischen mindestens zwei oder mehreren 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. Dies bedeutet, dass ähnliche oder gleiche Muster erkennbar sind.

Bei Autokorrelation sind also die Werte einer Variable zum Zeitpunkt tn mit den Werten derselben Variable in zeitlich vergangenen Perioden abhängig. 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 (tn) und der, um eine Zeit tn+m, verschobenen Folge besteht.

## Korrelationskoeffizienten

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 schwächer ist die Korrelation. Ein Korrelationskoeffizient mit dem Wert 0 bedeutet, dass keine Korrelation vorhanden ist und die Werte perfekt verteilt[[4]](#footnote-4) sind.



Abbildung : 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.

## Anwendungsbereiche

Autokorrelation kann durch mathematische Formeln ausgedrückt werden, jedoch wird sie in jedem Anwendungsbereich domänenspezifisch definiert. Als die signifikantesten Anwendungsgebiete gelten Statistik, Signalanalyse, Informationstheorie und die Softwaretechnik.

**Autokorrelation in der Statistik:** In der Statistik wird durch die Autokorrelation das Mass des Zusammenhangs zwischen zwei Zufallsvariablen beschrieben. Am häufigsten wird dieses Mass in Form der Korrelationskoeffizienten (Pearson) angegeben.

**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.

**Autokorrelation in der Informationstheorie:** [TODO] Kryptographie erwähnen & Kurzbeschreibung zur Informationstheorie

## Partielle Korrelation

Unter der partiellen Korrelation versteht man das nicht-berücksichtigen von Dritteinflüssen. Eine Korrelation zwischen zwei statistischen Werten a und b kann unter Umständen auf einen gemeinsamen Faktor c zurückgeführt werden. Um diesen Effekt auszuschalten kann das Konzept der partiellen Korrelation eingesetzt werden. Durch eine partielle Korrelation wird der dritte Faktor entweder ausgeschaltet oder gezielt kontrolliert, so dass dieser das Resultat nicht verfälschen kann.

## Durbin-Watson-Test

Die gebräuchlichste Methode um die Existenz von Autokorrelation zu belegen, stellt der Durbin-Watson-Test dar. Durch diesen statistischen Test kann geprüft werden, ob eine Autokorrelation der 1. Ordnung vorliegt. Autokorrelation 1. Ordnung bedeutet, dass aufeinanderfolgende Glieder der Reihe bzw. ihrer Residualgrössen[[5]](#footnote-5) korrelieren. Das Ergebnis eines Durbin-Watson-Tests ist ein numerischer Wert im Bereich von 0 bis 4.

|  |  |  |
| --- | --- | --- |
| **Wert des Tests** | **Korrelationskoeffizient** | **Bedeutung** |
| d = 2 | 0 | Keine Autokorrelation |
| d = 0 | 1 | Perfekte positive Autokorrelation |
| d = 4 | -1 | Perfekte negative Autokorrelation |

Der DW-Test ist durch den folgenden Term definiert:

|  |  |
| --- | --- |
| **Ausdruck** | **Bedeutung** |
|  | Summiert alle Sequenzglieder zwischen t = 2 und T, wobei t und T die Menge aller Werte fednieren Die Anzahl der Beobachtungen entspricht dem Start und -Endwert der Zeitreihe. |
|  | Entsprechen den Residuen/Werte der Reihe. |
|  | *[TODO]* |

## Beispiel Autokorrelation

Folgend dargestellt ist ein Beispiel zur Autokorrelation aus dem Bereich der Kryptographie. In der Kryptographie stellt die Autokorrelation eine Kennzahl für die Ähnlichkeit von Teilen eines Dokuments dar. Mithilfe der Autokorrelation kann unter Umständen die Schlüssellänge eines verschlüsselten Dokuments ermittelt werden.

Als Grundlage dient die Verschlüsselungsmethode Vigenère-Chiffre. Bei diesem Verfahren handelt es sich um ein Substitutionsverfahren, wobei der Klartext in Monogramme (einzelne Zeichen) zerlegt wird und diese anschliessend durch Geheimtextzeichen substituiert werden.

In diesem Beispiel wird vom Standardalphabet mit 26 Buchstaben ausgegangen. Daraus wird eine Matrix (Vigenère-Quadrat) erstellt, welches die 26 Buchstaben immer um eine Position verschoben darstellt.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | **Klartext** | | | | | | | | | | | | | | | |  |  |  |  |  |  |
|  |  | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |  |
|  | A | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |  |
|  | B | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |  |
| **Schlüssel** | C | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | **Geheimtext** |
| D | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| E | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| F | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| G | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| H | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| I | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| J | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| K | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| L | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| M | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| N | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| O | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| P | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| Q | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| R | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| S | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| T | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| U | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| V | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| W | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
|  | X | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |  |
|  | Y | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |  |
|  | Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |  |

Tabelle : Vigenère-Quadrat

Anschliessend muss ein Schlüssel gewählt werden. Dieser sollte möglichst lang und aus einer möglichst zufälligen Sequenz der Buchstaben des Alphabets bestehen. Der Kreuzungspunkt der einzelnen Buchstaben des Klartextes und des Schlüssels können nun innerhalb des Vigenère-Quadrats abgelesen werden und ergeben so das neue, verschlüsselte Zeichen.

Mithilfe der Software Cryptool[[6]](#footnote-6) können solch einfache Verschlüsselungsverfahren aufgezeigt und analysiert werden. Cryptool verwendet folgende Autokorrelationsfunktion C(t), welche die Ähnlichkeit einer Folge[[7]](#footnote-7) (s[i]) = s[1], s[2], …, s[n] und der um t Stellen verschobenen Folge (s[i+t]= s[1 + t], s[2 + t], s[n + t].

**C(t) =**

Wobei A(t) = Anzahl der übereinstimmenden Glieder der Folgen s[i] und s[i + t] im betrachteten Abschnitt

und D(t) = Anzahl der nicht übereinstimmenden Glieder derselben Folgen und Abschnitt ist. n beschreibt die Länge der Sequenz.

Zur Veranschaulichung werden diese Formeln in ein Autokorrelationsdiagramm umgesetzt.

### Beispiel 1 – starke Autokorrelation

|  |
| --- |
| **Schlüssel:**  ABCDEFGHIJKLMNOPQRSTUVWXYZ |
| **Klartext:**  ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ |
| **Verschlüsselter Text:**  ACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWY |

Dadurch, dass die Vigenère-Chiffre auf Substitution und Verschiebung der einzelnen Zeichen basiert, korrelieren die einzelnen Zeichen nach einer gewissen Zeit bzw. Verschiebung miteinander. Das erste, triviale, Beispiel zeigt eine sehr starke Autokorrelation, da der Klartext lediglich ein Vielfaches des Schlüssels ist.



Figure Autokorrelation des unverschlüsselten Textes, Bsp. 1

Wird nun die Autokorrelation des unverschlüsselten Textes betrachtet, so kann man die Verschiebung um 26 Zeichen klar erkennen.



Figure Autokorrelation verschlüsselter Text, Bsp.1

Man würde erwarten, dass auch beim verschlüsselten Text die Autokorrelation bei 26 Verschiebungen am stärksten ist. Jedoch ist dies ein Trugschluss. Die Autokorrelation ist bei 13 Verschiebungen deutlich am stärksten.

Weiter ist der Vergleich zwischen verschlüsseltem Text und des Klartextes spannend. Dort kann gesehen werden, dass «ABCDEFGHIJKLMNOPQRSTUVWXYZ» der Zeichenkette «ACEGIKMOQSUWYACEG-IKMOQSUWY» entspricht. Es wird erkannt, dass genau nach 13 Zeichen wiederrum das Zeichen «A» auftaucht, was auf die obengenannte Struktur des Vigenère-Quadrat hinweist.

### Beispiel 2

|  |
| --- |
| **Schlüssel:**  AUTOKORRELATION |
| **Klartext:** [[8]](#footnote-8)  Die Giraffensind eine Gattung der Säugetiere aus der Ordnung der Paarhufer. Ursprünglich wurde ihr mit Giraffa camelopardalis und der Trivialbezeichnung Giraffe nur eine einzige Art zugewiesen. Molekulargenetische Untersuchungen zeigen jedoch, dass die Gattung wenigstens vier Arten mit sieben eigenständigen Populationen umfasst. Die Giraffen stellen die höchsten landlebenden Tiere der Welt. Zur Unterscheidung vom verwandten Okapi werden sie auch als Steppengiraffen bezeichnet. |
| **Verschlüsselter Text:**  Dcx Usfrwjpn lqbq ecgs Qokkyyg wmf Fäuaxhssiv efs wmf Brxgixu uvv Aatzvhfyk. Ibggiürrlbkv julws svi dme Gbzosfu vowscftlrwizvs ogr nsi Kvtvbizoetxwmvelrr Gbzosfy gib szei pighwte Ukh jixvatelmb. Zofxyezrikpnxbwfcbx Ixhviwfcacbteh sssuve npdhkv, qaml rss Xrxeugo krnczgdsej ztek Ifgeh fwd gzvfpn xqurnmmäbnwxvr Aoiczntchbob ldjlslb. Rve Abfktwvr dtxtzrn xbs röqyjxpn eibqlyusxrve Xtekm rrr Qxzd. Nli Yytxzgphybrebx msx vxzknnxmsx Cbrtt wxzrrn mbs kity ews Lbscpygusfrwjpn umnriwaboh. |



Abbildung Autokorrelation des Klartextes

Auf den ersten Blick vermittelt dieses Diagramm einen willkürlichen Eindruck. Jedoch können auch hier autokorrelierte Strukturen erkannt werden. Diese sind nicht mehr nach einer fixen Anzahl Zeichen erkennbar wie in Beispiel 1, trotzdem sind vereinzelte, sehr starke Korrelationen ersichtlich.

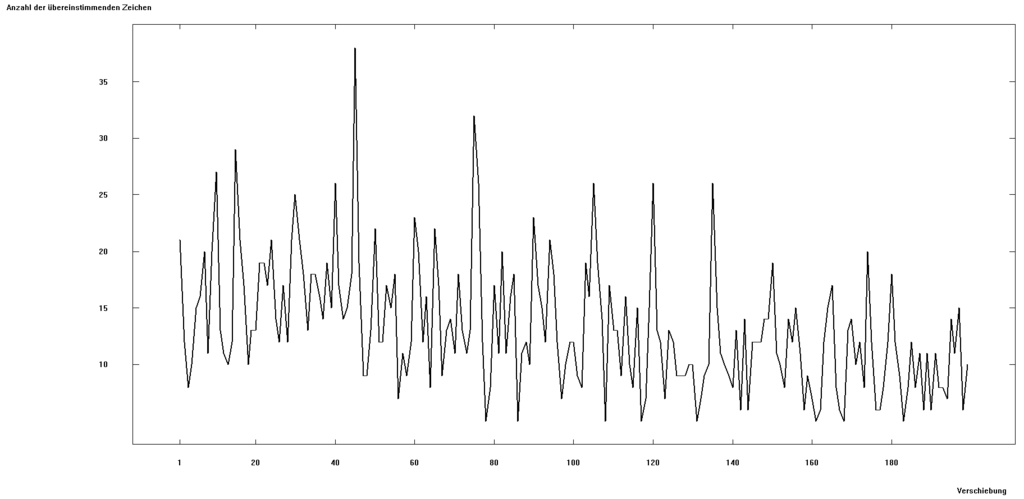


Figure Autokorrelation des verschlüsselten Textes, Bsp. 2 - Giraffen

# Autoregressive to anything [bis 18.11.2017]

Dieses Kapitel befasst sich mit dem ARTA-Prozess, welcher die Grundlage des Projektes bildet/vorgibt. Anhand mathematischer und graphischer Elemente sollen die mitwirkenden Komponenten veranschaulicht werden.

Folgende Grafik bildet die einzelnen Bestandteile des ARTA-Prozesses ab. In den folgenden Kapiteln wird auf die grundlegenden Elemente eingegangen.



Figure Grafische Darstellung der Bestandteile eines ARTA-Prozesses

## Zufallszahlen – Mersenne-Twister

Der ARTA-Prozess benötigt eine Inputsequenz. Diese entstammt aus einem Zufallszahlengenerator. Die Generierung der Zufallszahlen basiert auf dem Algorithmus des Mersenne-Twister[[9]](#footnote-9), entwickelt von Makoto Matsumoto und Takuji Nishimura, 1997. Der Algorithmus existiert in zwei Varianten, wir verwenden MT19937. Die andere Variante wird TT8800 genannt, arbeitet grundsätzlich nach dem gleichen Prinzip, kann jedoch nur eine kleinere Datenmenge verarbeiten.

Mersenne-Twister weist drei Eigenschaften auf, welche ihn für die vorliegende Implementation qualifizieren.

1. Er weist eine extrem lange Periode auf. Dies ist ein Kriterium, welches die Güte des Generators beschreibt. Die Periodenlänge des Mersenne-Twister beträgt p = 219937 – 1 (Mersenne-Primzahl).
2. Alle Werte bzw. Bits der Ausgabesequenz sind hochgradig gleichverteilt. Im Fall des Mersenne-Twister erfolgt diese Verteilung bis zur 623 Dimension[[10]](#footnote-10). Daraus resultiert eine extrem geringe Korrelation zwischen den aufeinanderfolgenden Zufallszahlen.
3. Der Algorithmus ist schnell. Eine Ausnahme bilden hier Rechenarchitekturen bzw. -Systeme, welche nur über einen sehr begrenzten Arbeitsspeicher verfügen.

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

Die Grundlage bildet eine Zahlensequenz. Die Startwerte liegen bei Y1 bis YN, wobei N = 624.Die ersten 624 Werte sind im Idealfall echte Zufallszahlen, jedoch funktioniert der Algorithmus auch mit Pseudozufallszahlen. ARTA.Standard erzeugt diese Zufallszahlen innerhalb der Klasse RandomSource, wobei es sich in diesem Fall lediglich um Pseudozufallszahlen handelt. Die weiteren Werte mit N > 624 werden folgendermassen berechnet:

**h = Yi -N -Yi-N mod 231  Yi-N+1 mod 231**

**Yi = Yi – 227 XOR h/2 XOR ((h mod 2) \* 0x9908B0DF)**

Abschliessend wird ein Tempering durchgeführt, dadurch wird die Gleichverteilung der Zufallszahlen sichergestellt.

|  |  |
| --- | --- |
| **Mersenne-Twister Algorithmus (Tempering)** | **Implementation** |
| X = Yi XOR Yi / 211  Y = x XOR ((x \* 27) & 0x9D2C5680)  Z = y XOR ((y \* 215) & 0xEFC60000)  Zi = z XOR z / 218 | x ^= y >> 11;  y = y ^ (y << 7 & - 0x9D2C5680;  z ^= y << 15 & - 0xEFC60000;  z ^= z >> 18;  return z; |

## Zeitreihen / AR-Prozesse

Eine Zeitreihe[[11]](#footnote-11) beschreibt eine Sequenz von Werten, welche sich an eine bestimmte Struktur halten. Diese Struktur wird durch einen Zeitkoeffizienten definiert. Die einzelnen Werte sind an den entsprechenden Zeitpunkt gebunden. Folgendes Beispiel soll die Grundidee einer Zeitreihe verdeutlichen.

**Yt = **

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **t - Werte** | 0 | 1 | 2 | 3 | 4 | 5 |
| **Yt =** | 1 |  |  |  |  |  |

Tabelle Beispiel Zeitreihe

Die Werte der Zeitreihe Y weisen zum Zeitpunkt t immer die Hälfte des vorhergegangenen Wertes des Zeitpunktes t – 1 auf.

Bei einem AR-Prozess muss der Wert zum Zeitpunkt t nicht nur vom Wert des Zeitpunktes t – 1 abhängen, sondern es ist denkbar, dass er auch vom Wert des Zeitpunktes t – 2 abhängt. Solche autoregressiven Prozesse werden in folgendermassen beschrieben:

**AR(p)**

Der Parameter p gibt dabei die höchste zeitliche Verzögerung (Lag) an. Beim obigen Beispiel ist dieser Lag = 1. Daher kann die Zeitreihe als AR(1) beschreiben werden.

Ein ARTA-Prozess modelliert eine stationäre Zeitserie. Die Basis bildet dabei ein stationärer, autoregressiver Gaussprozess (AR). Die Werte einer autoregressiven Zeitreihe hängen nicht systematisch vom vorhergegangen Werte ab, sondern können auch von Werten zu einem früheren Zeitpunkt abhängen. Der ARTA-zugrundeliegende AR-Prozess ist folgendermassen definiert:

**AR(p) = {Zt; t = 1, 2, ..., n} wobei Zt = α1Zt – 1 + α2Zt-2 + αpZt – p +** **εt**

Zt definiert den stationären AR(1)-Prozess, **εt** steht für zufällige, unabhängige Zufallsvariable der Normalverteilung N(0, 1).

Folgendes Codefragment zeigt die Berechnung des nächsten Sequenzgliedes eines AR(p)-Prozesses auf. «whiteNoseProcess» beschreibt hierbei die Normalverteilung N(0,1) bzw. **εt** in obigem Beispiel.

|  |
| --- |
| public double Next()  {  double value = whiteNoiseProcess.sample();  for(int i = 0; i < alphas.Length; i++) {  value = value + alphas[i] \* values.get(i);  }  values.add(value);  return value;  } |

Codefragment AR-Prozess - Next()-Methode

wobei rh die angestrebte Autokorrelation für den Lag h darstellt. Nun kann der Output des AR(p)-Prozesses durch die CDF (Cumulative Distribution Function) in gleichmässig verteilte Werte transformiert werden. Wird nun die Inverseverteilungsfunktion, auf die sich ergebenden Werte, angewendet, führt dies zu einem Prozess mit der gewünschten Randverteilung.

## ARTA und Autokorrelation [bis 01.11.2017]

Dem AR-Prozess liegt eine natürliche autokorrelierte Struktur zugrunde. Diese ist durch den Lag (Zeitverzögerung), welche durch den Parameter p ausgedrückt wird, gegeben. Die Herausforderung liegt nun darin, diese Autokorrelation auf den darüberliegenden ARTA-Prozess zu transformieren. Diese Aufgabe wird durch ein Yule-Walker-Gleichungssystem[[12]](#footnote-12) gelöst, welches ein numerisches Suchverfahren zur Bestimmung der Regressionskoeffizienten darstellt.

## Verteilungen

Die von einem Arta-Prozess erzeugten Zufallszahlen unterliegen einer definierten Verteilung. Jeder Wahrscheinlichkeitsverteilung kann eine Verteilungsfunktion [[13]](#footnote-13)zugeordnet werden. Dabei entspricht der Wert der Verteilungsfunktion an der Stelle x der Wahrscheinlichkeit, dass die zugehörige Zufallsvariable X einen Wert annimmt, der kleiner oder gleich x ist. Die Verteilungsfunktion kann durch zwei Definitionen ausgedrückt werden, einerseits durch das Wahrscheinlichkeitsmass oder mittels einer Zufallsvariable.

Definition mittels Wahrscheinlichkeitsmass: Auf dem Ereignisraum der reellen Zahlen sei das Wahrscheinlichkeitsmass P gegeben. Dies kann durch die Funktion

Ausgedrückt werden. Die Verteilungsfunktion von P lautet:

Die Funktion gibt an der Stelle x an, mit welcher Wahrscheinlichkeit ein Ergebnis aus der Menge (- eintritt.

Definition mittels Zufallsvariable: Ist X eine reelle Zufallsvariable, so definiert sich die Verteilungsfunktion von X folgendermassen:

Durch P wird die Wahrscheinlichkeit ausgedrückt, dass X einen Wert kleiner oder gleich x annehmen wird.

In den folgenden Unterkapiteln, wollen wir die gängigsten Verteilungen im Zusammenhang mit ARTA kurz erklären und deren Definition aufzeigen.

### Normalverteilung

Die Normalverteilung [[14]](#footnote-14)auch Gaussverteilung genannt, stellt ein wichtiger Typ stetiger Wahrscheinlichkeitsverteilungen dar. Ihre grosse Bedeutung beruht unter anderem auf dem zentralen Grenzwertsatz[[15]](#footnote-15).

|  |  |
| --- | --- |
| Name | Normalverteilung |
| Dichtefunktion |  |
| Verteilungsfunktion | Leine elementare Funktion |
| Erwartungswert |  |
| Varianz |  |
| Anwendung | * Messwerte * Summe vieler kleiner Einflüsse * Approximation der Binomialverteilung |



Figure : Verteilungsfunktion (oben) und Dichtefunktion (unten) der Normalverteilung

### Exponentialverteilung

Bei der Exponentialverteilung [[16]](#footnote-16)handelt es sich um eine stetige Wahrscheinlichkeitsverteilung über eine Menge positiver reeller Zahlen, welche durch eine Exponentialfunktion gegeben ist. Ihr Einsatzgebiet liegt in der Beantwortung der Frage der Dauer von zufälligen Zeitintervallen.

|  |  |
| --- | --- |
| Name | Exponentialverteilung |
| Dichtefunktion |  |
| Verteilungsfunktion |  |
| Erwartungswert |  |
| Varianz |  |
| Anwendung | * Prozesse ohne Erinnerungsvermögen * Radioaktivität |



Figure : Verteilungsfunktion (oben) und Dichtefunktion (oben) der Exponentialverteilung

### Stetige Gleichverteilung

Eine stetige Gleichverteilung [[17]](#footnote-17)beschreibt eine stetige Wahrscheinlichkeitsverteilung. Dies bedeutet, dass Werte auf einem Intervall eine konstante Wahrscheinlichkeitsdichte aufweisen. Demnach ist gegeben, dass alle gleichlangen Teilintervalle ebenfalls dieselbe Wahrscheinlichkeit besitzen.

|  |  |
| --- | --- |
| Name | Gleichverteilung |
| Dichtefunktion |  |
| Verteilungsfunktion |  |
| Erwartungswert |  |
| Varianz |  |
| Anwendung | * Verteilung von zufallszahlen * Keine bevorzugten Werte |



Figure : Verteilungsfunktion (oben) und Dichtefunktion (unten) der Gleichverteilung

### PearsonsCorrelation [bis 1.11.2017]

Die Klasse AutoCorrelation übernimmt die Funktion zur Errechnung der Korrelationskoeffizienten sowie der Erzeugung der Korrelationsmatrizen. Durch die Formel von Pearson kann leicht der Korrelationskoeffizient r zwischen zwei Variablen ermittelt werden. Folgendes numerische Beispiel soll dies verdeutlichen.

**r**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X – Werte** | 1 | 3 | 4 | 4 |
| **Y – Werte** | 2 | 5 | 5 | 8 |

Mithilfe der obenstehenden Formel können folgende Werte errechnet werden:

|  |  |
| --- | --- |
| **Ausdruck** | **Bedeutung/numerischer Wert** |
|  | (1)(2) + (3)(5) + (4)(5) + (4)(8) = 69 |
|  | 1 + 3 + 4 + 4 = 12 |
|  | 2 + 5 + 5 + 8 = 20 |
|  | 12 + 32 + 42 + 42 = 42 |
|  | 22 + 52 + 52 + 82 =118 |
| n | Anzahl der Werte/ Länge der Zahlenreihe |

Werden die errechneten Werte nun in die Gleichung eingesetzt, kann der Pearson-Korrelationskoeffizient ermittelt werden:

**r = = 0.866**

Das folgende Codefragment zeigt die Berechnung der Korrelationskoeffizienten, so wie sie in dieser Arbeit umgesetzt ist. Weiter übernehmen diese Klassen die Erzeugung der partiellen Korrelationskoeffizienten und der Korrelationsmatrizen.

|  |
| --- |
| public static double[] CalculateAcfs(double[] data, int maxLag)  {  double[] accs = new double[maxLag + 1];  for (int lag = 0; lag <= maxLag; lag++) {  accs[lag] = CalculateAcf(data, lag);  }  return accs;  }  public static double CalculateAcf(double[] data, int lag)  {  double acc = 0.0;  int len = data.Length;  if (lag < 0)  {  throw new ArgumentException("Negative Lags are not allowed");  }  if (lag > 1)  {  throw new ArgumentException("Lag exceeds sample size");  }  if (lag == 0)  {  acc = 1.0;  }  else  {  acc = Correlation.Pearson(data.CopyOfRange(0, len - lag), data.CopyOfRange(lag, len));  }  return acc;  } |

Codefragment : Berechnung der Korrelationskoeffizienten

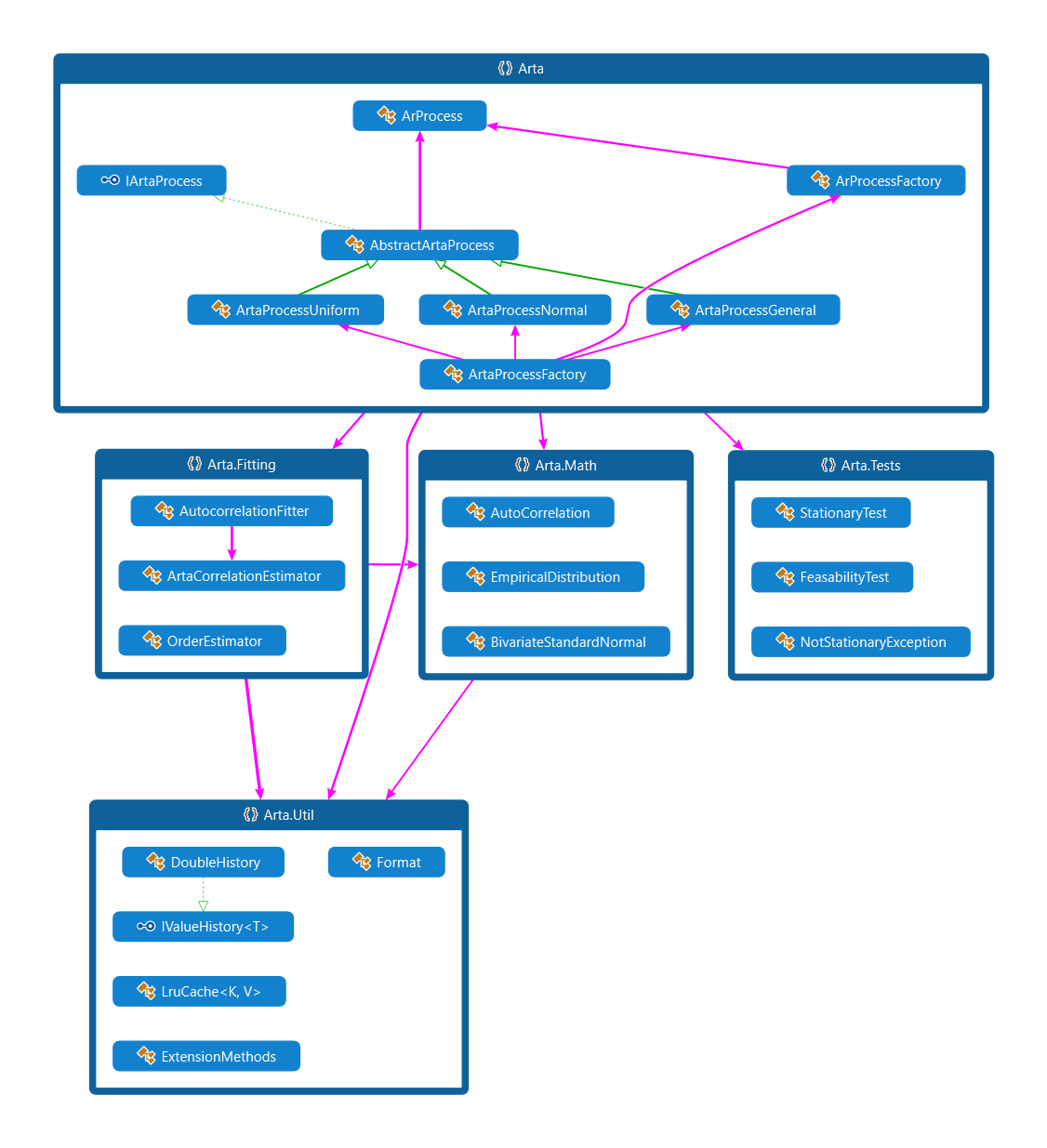
### Grenzen von ARTA [Philipp]

Ein weiterer Aspekt soll die Grenzen von ARTA aufzeigen. Damit ist gemeint, dass auf die angegebenen Schwächen, welche im Dokument «JARTA — A Java library to model and fit Autoregressive-To-Anything processes[[18]](#footnote-18)» genannt sind. *[TODO] Weiterführen, Grenzen aufzeigen*

# ARTA.Standard [bis 15.11.2017]

ARTA.Standard soll einerseits eine Klassenbibliothek als Grundlage der Modellierung stochastischer Prozesse darstellen, andererseits die Möglichkeit zur Integration in die Simulationssoftware Simio bereitstellen. Auf den folgenden Seiten sind die einzelnen, relevanten Klassen und Algorithmen dargelegt, welche essentiell zur Realisierung beitragen. ARTA.Standard greift auf die Sammlung mathematischer Funktionen und Klassen der MathNet.Numerics[[19]](#footnote-19)-Library zurück. Diese stellt eine Vielzahl an ausgewählten Klassen und Funktionen bereit, welche zur Modellierung des ARTA-Prozesses essentiell sind.

## Domain-Modell



Legende:

Calls

Implements

Inherits from

Abbildung : Klassendiagramm ARTA.Standard

## Implementation

Die Kernkomponente liefern die beiden Klassen ArtaProcessFactory und ArProcessFactory, welche den ARTA-Prozess und den darunterliegenden AR(p)-Prozess erzeugen. Die ArProcessFactory erzeugt den AR-Prozess mithilfe eines Zufallszahlengenerators (hier Mersenne-Twister) und gegebenen Autokorrelationskoeffizienten. Somit kann der Grad der Autokorrelation entsprechend frei gewählt werden, solange die Koeffizienten in den entsprechenden Wertebereichen liegen. Die ArtaProcessFactory nimmt den erzeugten AR-Prozess und eine Randverteilung entgegen um den entsprechenden Prozess zu erzeugen.

[TODO] während Implementationsphase konstant erweitern

[TODO] Sequenzdiagramm ergänzen

Folgendes Codefragment (Auszug aus ArProcessFactory.cs) zeigt die Erzeugung eines neunen AR-Prozesses.

|  |
| --- |
| ///<summary>  ///Erzeugt einen AR-Prozess mit den gegebenen Korrelationskoeffizienten.  ///Passt die Alpha-Werte in eine Normalverteilung ein, mit dem Mittelwert 0 und der Varianz kleiner 1  ///</summary>  **public static ArProcess CreateArProcess(double[] arAutocorrelations, RandomGenerator rng)**  **{**  //Erzeugt eine Korrelationsmatrix und gibt die Reihe mit Index 0 als double[] zurück  double[] alphas = ArAutocorrelationsToAlphas(arAutocorrelations);  /\*  Errechnet die Varianz aus den gegebenen Korrelationskoeffizienten und den erzeugten Alpha- Werten  \*/  double variance = CalculateVariance(arAutocorrelations, alphas);  /\*  Erzeugt eine Normalverteilung der zufällig erzeugten Werte des Zufallszahlen-generators, untere Grenze 0.0, obere Grenze @variance. Wendet die Umkehrfunktion der Normalverteilung an um die gewünschte Randverteilung zu erhalten.  \*/  NormalDistribution whiteNoiseProcess = new NormalDistribution(rng, 0.0, Math.Sqrt(variance), NormalDistribution.DEFAULT\_INVERSE\_ABSOLUTE\_ACCURACY);  return new ArProcess(alphas, whiteNoiseProcess);  **}** |

Codefragment : ArProcessFactory.CreateArProcess()

Auf der Basis des erzeugten AR-Prozesses kann die ArtaProcessFactory den entsprechenden ARTA-Prozess instanziieren.

|  |  |
| --- | --- |
|  |  |
| Verteilung | ContinousUniform |
| Korrelationskoeffizienten | -0.1, 0.3 |

|  |
| --- |
| ContinuousUniform distribution = new ContinuousUniform();  double[] artaCorrelationCoefficients = { 0.3, -0.1 };  IArtaProcess arta = ArtaProcessFactory.CreateArtaProcess(distribution, artaCorrelationCoefficients); |

Codefragment : Erzeugung eines ARTA Prozesses

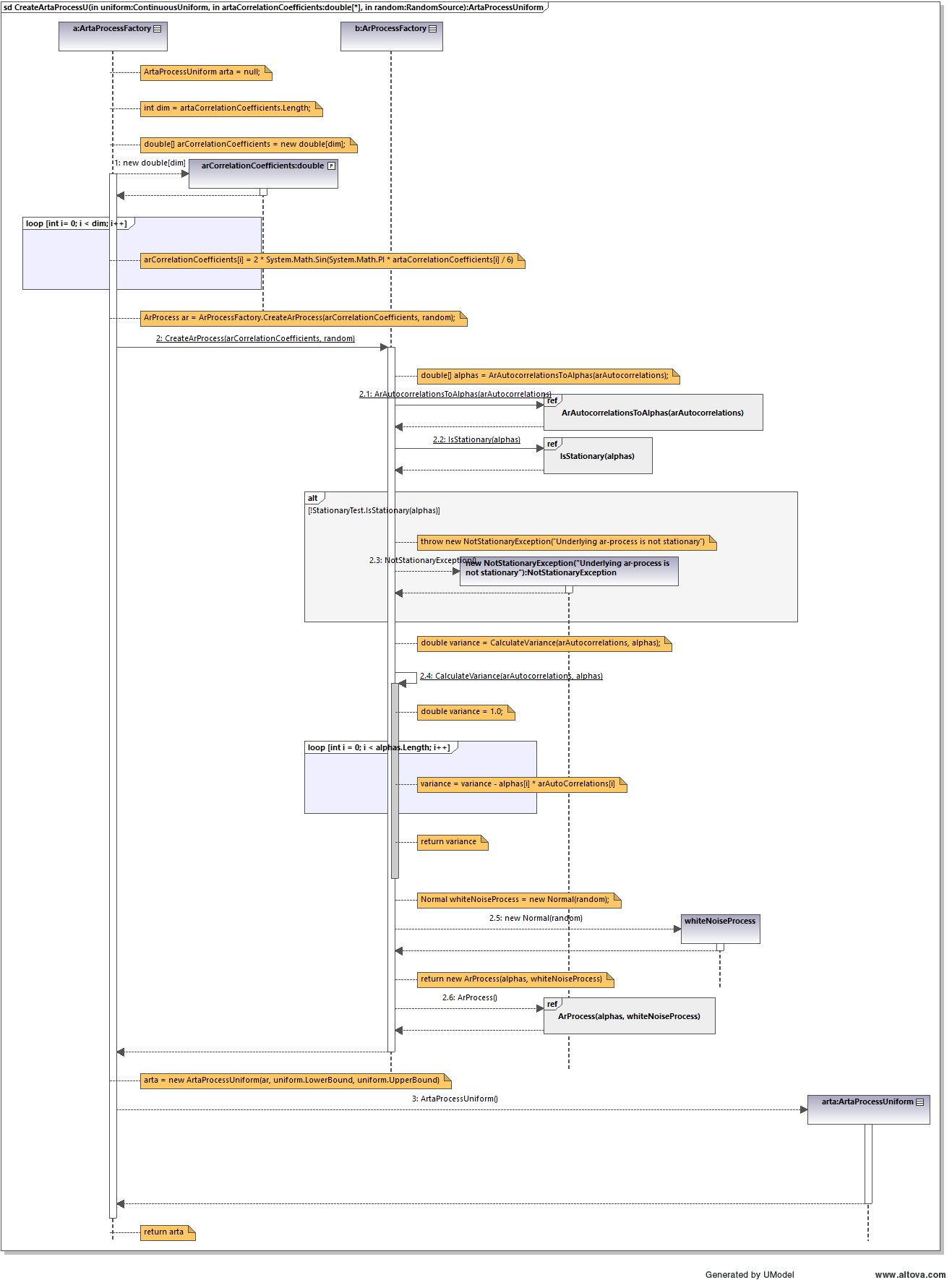


Abbildung : Sequenzdiagramm - Erzeugung eines ARTA Prozesses

[TODO] Anthony + Design mit Context & Distribution

## Statistische Tests [Philipp]

Tests sind in einem separaten Assembly StatisticalTests realisiert. Dabei handelt es sich lediglich um Tests der Klassenbibliothek an sich. Die Integration in Simio wird separat in Form eines Integrationstestes und verschiedener Szenarien getestet.

Als Werkzeug zum Testen der Klassenbibliothek wurde ArtaStatistics implementiert. Dabei handelt es sich um eine Klasse, welche einen ArtaExecutionContext entgegennimmt und anschliessend Abfragen auf einzelne Elemente des ARTA-Prozesses ermöglicht.

|  |  |
| --- | --- |
| **Methode** | **Beschreibung** |
| Initialize(int lag) | Gibt den maximalen Lag an |
| Iterations(int iterations) | Setzt die Anzahl zu generierender ARTA-Zahlen |
| Acfs() | Berechent die Autokorrelationsfunktionen (ACFS) |
| Pacfs() | Berechnet die partiellen Autokorrelationsfunktionen(PACFS) |
| ArtaNumbers() | Erzeugt ARTA-Zahlen |
| Order() | Errechnet die Order |
| Execute() | Gibt die |

|  |
| --- |
| var executionContext = new ArtaExecutionContext(new ExponentialDistribution(), new double[] { -0.4, 0.5 });  var artaProcess = executionContext.CreateArtaProcess();  ArtaStatistics arta = new ArtaStatistics(executionContext).Initialize(10).Iterations(1000).ArtaNumbers().Acfs().Pacfs().Excecute(); |

Codefragment : Nutzung ArtaStatistics

### ARTAProcess Tests

[TODO] Philipp

Weitere Tests sollen die Vollständigkeit und Funktionalität des abgebildeten ARTA-Prozesses abdecken. Dazu werden verschiedene ARTA-Prozesse mit verschiedenen Parameter erzeugt und anschliessend geprüft, ob die resultierenden Werte den Erwartungen entsprechen. Diese Tests decken ebenfalls die verschiedenen Verteilungen ab.

[TODO] Sobald vollständige Tests vorhanden, Codefragment ergänzen

|  |
| --- |
| RealDistribution distribution = new ExponentialDistribution(1.0);  double[] artaCorrelationCoefficients = { 0.3, 0.3, -0.1 };  IArtaProcess arta = ArtaProcessFactory.CreateArtaProcess(distribution, artaCorrelationCoefficients);  double[] data = new double[10000];  for (int i = 0; i<data.Length; i++) {  data[i] = arta.Next();  }  int maxLag = 10;  double[] acfs = AutoCorrelation.CalculateAcfs(data, maxLag);  double[] pacfs = AutoCorrelation.CalculatePacfs(acfs); |

Codefragment : Beispiel eines Tests der ARTAProcessFactory

## Integration Simio [bis 13.12.2017]

Die Integration in die Simulationssoftware Simio ist im Assembly «Arta.Simio» umgesetzt. Die Grundlage bildet ein von Simio bereitgestelltes Visual-Studio-Template. Dieses gibt die Grundstruktur entsprechend vor. Für die Implementation wurde das Template «User-AddIn» verwendet.

Innerhalb der Klasse [Classname] wird ein ArtaElement erzeugt. Das ArtaElement enthält Properties welche später den Arta-Prozess definieren (Korrelationskoeffizienten). Weiter sind drei spezifische Properties implementiert, welche die jeweiligen Verteilungen bereitstellen.

Innerhalb von Simio kann nun ein ArtaElement erzeugt werden und dies als InterarrivalTime-Property einer Source übergeben werden.

[TODO]Verifikation, dass richtige Zeitabstände genommen werden.

[TODO] während Implementation genauer beschreiben.

[TODO] Typisierung dokumentieren, verschiedene Properties aufzeigen

[TODO] Klassendiagramm Arta.Simio verbessern

[TODO] Nachdem Code fertig, entsprechend dokumentieren



# Test und Auswertung [[bis 25.11.2017]

[TODO] Philipp

Dieses Kapitel deckt die Verifikation der Klassenbibliothek ab. In einem ersten Schritt wollen wir die Daten, welche unsere Implementation ausgibt mit den der bestehenden (JARTA) vergleichen. Dazu erzeugen wir verschiedene ARTA-Prozesse mit verschiedenen Parametern, jeweils mit unserer Implementation und in JARTA. Anschliessend geben wir die Sequenz von ARTA-Zahlen in ein Excel-File aus und analysieren die Daten mittels Diagrammen.

Folgende Tabelle soll Aufschluss über die Parameter und Einstellungen über alle Vergleiche geben.

|  |  |
| --- | --- |
| **Bezeichnung** | **Parameter** |
| Anzahl analysierter Zahlen | 100 |
| Lag | 10 |
| Korrelationskoeffizienten | -0.4, 0.5 |
|  |  |
|  |  |

## ARTA – ContinousUniform

|  |  |
| --- | --- |
| **Parameterbezeichnung** | **Wert** |
| Verteilung | ContinousUniform |
| LowerBound | -1 |
| UpperBound | 1 |

### ACFS/PACFS

|  |  |  |  |
| --- | --- | --- | --- |
| **ARTA.Standard** | | **JARTA** | |
| **ACFS** | **PACFS** | **ACFS** | **PACFS** |
| 1 | 1 | 1 | 1 |
| -0.822630072 | -0.822630072 | -0.4165822 | -0.416582159 |
| 0.836934856 | 0.160214621 | 0.5090079 | 0.335467202 |
| -0.762535905 | 0.059270643 | -0.2992634 | 0.059191148 |
| 0.741310287 | 0.029553697 | 0.27696673 | 0.007012443 |
| -0.698079634 | 0.00888604 | -0.1987881 | -0.011560657 |
| 0.670680655 | 0.00558979 | 0.16508882 | -0.001478902 |
| -0.635117809 | 0.00433623 | -0.1258744 | 6.32E-04 |
| 0.609157037 | 0.004504524 | 0.09810887 | -0.00263461 |
| -0.580379504 | 0.001151885 | -0.0899578 | -0.015718683 |
| 0.558146278 | 0.003586952 | 0.07513454 | 0.004676503 |

Abbildung : ACFS im Vergleich mit ContinousUniform (-1, 1)

Abbildung : PACFS im Vergleich mit ContinousUniform (-1, 1)

Abbildung : Vergleich ARTA-Zahlen mit ContinousUniform (-1, 1)

## ARTA – Normal

|  |  |
| --- | --- |
| **Parameterbezeichnung** | **Wert** |
| Verteilung | Normal |
| Mean | 0 |
| Varianz | 1 |

### ACFS/PACFS

|  |  |  |  |
| --- | --- | --- | --- |
| **ARTA.Standard** | | **JARTA** | |
| **ACFS** | **PACFS** | **ACFS** | **PACFS** |
| 1 | 1 | 1 | 1 |
| -0.772725676 | -0.772725676 | -0.399101088 | -0.39910109 |
| 0.794123716 | 0.197018745 | 0.499673667 | 0.34039199 |
| -0.688091837 | 0.080930834 | -0.276768674 | 0.06616926 |
| 0.656300037 | 0.032123014 | 0.273625941 | 0.02214178 |
| -0.592528152 | 0.01125053 | -0.178230151 | 0.0010887 |
| 0.551392556 | 0.003249382 | 0.159867604 | 0.0037281 |
| -0.499081516 | 0.006603324 | -0.103811582 | 0.01094123 |
| 0.460924491 | 0.00395471 | 0.087651036 | 3.89E-04 |
| -0.418321714 | 0.003405857 | -0.07322328 | -0.016045 |
| 0.385498154 | 0.002817221 | 0.053287197 | -0.00777538 |

Abbildung : ACFS Vergleich ARTA.Standard und JARTA, mit N (0,1)

Abbildung : PACFS Vergleich ARTA.Standard und JARTA, mit N (0,1)

Abbildung : Generierte ARTA-Zahlen mit N (0,1)

## ARTA – Exponential

### ACFS/PACFS

|  |  |  |  |
| --- | --- | --- | --- |
| **ARTA.Standard** | | **JARTA** | |
| **ACFS** | **PACFS** | **ACFS** | **PACFS** |
| 1 | 1 | 1 | 1 |
| -0.352431384 | -0.352431384 | -0.4057 | -0.4057 |
| 0.769765957 | 0.645558076 | 0.519637 | 0.355045 |
| -0.333929006 | 0.282681961 | -0.30831 | 0.053268 |
| 0.61893006 | 0.193684645 | 0.306799 | 0.021615 |
| -0.310477583 | -0.050560009 | -0.22037 | -0.00614 |
| 0.518574645 | -0.1221241 | 0.192266 | -0.00387 |
| -0.284221243 | -0.030407946 | -0.13807 | 0.011617 |
| 0.431806635 | 0.061462457 | 0.124674 | 0.011863 |
| -0.256415253 | 0.056167963 | -0.09608 | -0.00286 |
| 0.358656033 | -0.016576183 | 0.083002 | -0.00109 |

Abbildung : Die ACFS im Vergleich, Exponentialverteilung

Abbildung : Vergleich der PACFS zwischen JARTA und ARTA.Standard, in einer Exponentialverteilung

Abbildung : Vergleich der von ARTA generierten Zahlen, exponentiell verteilt

## Resultate

# Anwendungsfall und Simulation [bis 13.12.2017]

Als letzte Phase der vorliegenden Arbeit, nutzen wir die Klassenbibliothek bzw. die Simio-Erweiterung, um Simulationsprojekte mit autokorrelierten Zufallszahlen zu speisen. In einem ersten Schritt nutzen wir Arta.Simio in einer eigenen Simulation, anschliessend stellen wir das Beispiel eines Lagerhauses von Uhlig und Rank nach. Die zweite Simulation gibt uns die Möglichkeit, einen direkten Vergleich zu wagen.

## Eigene Simulation

### Experimentaufbau

### Simulationsumgebung

[TODO] geeigneten Titel finden

## Lagerhaus

Als Grundlage des Experiments gilt das Beispiel eines Lagersystems. Dieses besteht aus einem Hochregallager mit vier Bahnen (L1 bis L4) und zwei Worker (K1/K2) in Form von Warteschlangen. Verbunden werden sie durch ein Fördersystem, welches über einen Eingang (WE) und Ausgang (WA) verfügt. Die vier Bahnen des Lagers stellen je eine Quelle dar, welche sie je einem separaten Zufallsgenerator mit exponentialverteilten Zufallszahlen speist. Zur Generierung dieser Zufallszahlen wird nun der ARTA-Ansatz genutzt und die Interarrival Time gesetzt.

Anschliessend werden verschiedene Materialflüsse mit Materialangaben definiert.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Materialfluss** | **Aufträge/h** | **Materialfluss** | **Aufträge/h** | **Materialfluss** | **Aufträge/h** |
| WE 🡪 L1 | 22.5 | L1 🡪 K1 🡪 WA | 11.25 | L4 🡪 K1 🡪 WA | 11.25 |
| WE 🡪 L2 | 22.5 | L1 🡪 K2 🡪 WA | 11.25 | L4 🡪 K2 🡪 WA | 11.25 |
| WE 🡪 L3 | 22.5 | L2 🡪 K2 🡪 WA | 11.25 | L1 🡪 K2 🡪 L3 | 2.0 |
| WE 🡪 L4 | 22.5 | L2 🡪 K2 🡪 WA | 11.25 | L2 🡪 K2 🡪 L1 | 2.0 |
| WE 🡪 K1 🡪 WA | 15.0 | L3 🡪 K1 🡪 WA | 11.25 | L3 🡪 K2 🡪 L4 | 2.0 |
| WE 🡪 K2 🡪 WA | 1 | L3 🡪 K2 🡪 WA | 11.25 | L4 🡪 K2 🡪 L2 | 2.0 |

Tabelle : Materialflüsse mit Materialangaben

In einem weiteren Schritt werden die einzelnen Quellen mit Korrelationskoeffizienten parametrisiert. Dabei ist zu erwähnen, dass eine Exponentialverteilung auf Grund ihrer Natur lediglich einen Koeffizienten entgegennimmt. Jede Zeile definiert einen eigenen Durchlauf mit den gesetzten Parametern.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp.** | **Autokorrelationskoeffizient für Quellen** | | | | |
|  | WE | L1 | L2 | L3 | L4 |
| A- | -0.10 | -0.10 | -0.10 | -0.10 | -0.10 |
| B- | -0.25 | -0.25 | -0.25 | -0.25 | -0.25 |
| C- | -0.50 | -0.50 | -0.50 | -0.50 | -0.50 |
| A+ | +0.10 | +0.10 | +0.10 | +0.10 | +0.10 |
| B+ | +0.25 | +0.25 | +0.25 | +0.25 | +0.25 |
| C+ | +0.50 | +0.50 | +0.50 | +0.50 | +0.50 |

Tabelle : Korrelationskoeffizienten der Quellen

Wir stellen diese Simulationsumgebung innerhalb der Simulationssoftware Simio um. Dazu wurde folgendes Model entwickelt.

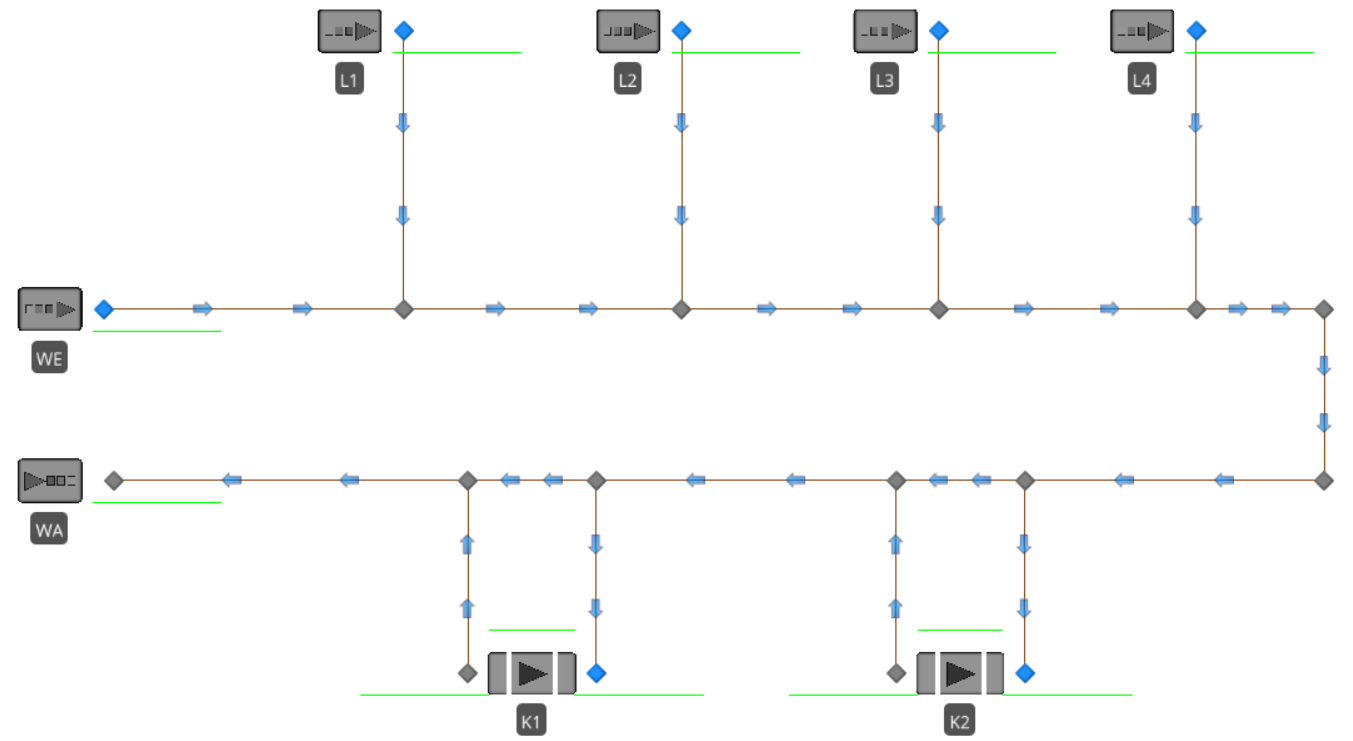


Abbildung : Lagerhaus

# Fazit und Ausblick [bis 20.12.2017]

# Literaturverzeichnis und Referenzen

# Abbildungsverzeichnis

[Abbildung 1: Korrelationskoeffizient 5](#_Toc499629187)

[Abbildung 2 Autokorrelation des Klartextes 10](#_Toc499629188)

[Abbildung 3: Klassendiagramm ARTA.Standard 21](#_Toc499629189)

[Abbildung 4: Sequenzdiagramm - Erzeugung eines ARTA Prozesses 23](#_Toc499629190)

[Abbildung 5: ACFS im Vergleich mit ContinousUniform (-1, 1) 27](#_Toc499629191)

[Abbildung 6: PACFS im Vergleich mit ContinousUniform (-1, 1) 28](#_Toc499629192)

[Abbildung 7: Vergleich ARTA-Zahlen mit ContinousUniform (-1, 1) 29](#_Toc499629193)

[Abbildung 8: ACFS Vergleich ARTA.Standard und JARTA, mit N(0,1) 30](#_Toc499629194)

[Abbildung 9: PACFS Vergleich ARTA.Standard und JARTA, mit N(0,1) 31](#_Toc499629195)

[Abbildung 10: Generierte ARTA-Zahlen mit N(0,1) 32](#_Toc499629196)

[Abbildung 11: Lagerhaus 34](#_Toc499629197)

# Codefragmente

[Codefragment 1 AR-Prozess - Next()-Methode 14](#_Toc499629198)

[Codefragment 2: Berechnung der Korrelationskoeffizienten 20](#_Toc499629199)

[Codefragment 3: ArProcessFactory.CreateArProcess() 22](#_Toc499629200)

[Codefragment 4: Erzeugung eines ARTA Prozesses 22](#_Toc499629201)

[Codefragment 5: Beispiel eines Tests der ARTAProcessFactory 24](#_Toc499629202)

1. Pereira, D. et al.: Autocorrelation effects in manufacturing systems performance: a simulation analysis. In: Laroque, C. et al. (Hrsg.): Proceedings of the Winter Simulation Conference (WSC), Berlin, 9.–12. Dez. 2012, S. 123:1–123:12. [↑](#footnote-ref-1)
2. Modeling and generating multivariate time-series input processes using a vector autoregressive technique, 10.1145/937332.937333 [↑](#footnote-ref-2)
3. JARTA — A Java library to model and fit Autoregressive-To-Anything processes, 10.1109/WSC.2013.6721508 [↑](#footnote-ref-3)
4. Siehe Kapitel [x] zum Thema Verteilungen [↑](#footnote-ref-4)
5. Vertiefter Einblick Residuum: <https://de.wikipedia.org/wiki/Residuum_(Statistik)> [↑](#footnote-ref-5)
6. https://www.cryptool.org/de/cryptool1 [↑](#footnote-ref-6)
7. Die Indexierung der Folge ist 1 - basiert [↑](#footnote-ref-7)
8. Quelle : https://de.wikipedia.org/wiki/Giraffen [↑](#footnote-ref-8)
9. Matsumoto, M.; Nishimura, T. (1998). [*"Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator"*](http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/ARTICLES/mt.pdf) 3–30. [*doi*](https://en.wikipedia.org/wiki/Digital_object_identifier):[*10.1145/272991.272995*](https://doi.org/10.1145%2F272991.272995) [↑](#footnote-ref-9)
10. N-Dimensional: Wird die Ausgabesequenz in Tupel von je n Zahlen zerlegt, so sind diese gleichverteilt im n-dimensionalen Raum. [↑](#footnote-ref-10)
11. Quelle: Zeitreihenanalyse- Einstieg und Aufgaben von Thomas Mazzoni, FernUniversität in Hagen [↑](#footnote-ref-11)
12. <https://de.wikipedia.org/wiki/Yule-Walker-Gleichungen>, Peter J. Brockwell, Richard A. Davis: *Time Series. Theory and Methods, Springer*. 2. verb. Aufl. Springer, New York 2006, [ISBN 978-0-387-97429-3](https://de.wikipedia.org/wiki/Spezial:ISBN-Suche/9780387974293). [↑](#footnote-ref-12)
13. Quellen: <https://de.wikipedia.org/wiki/Verteilungsfunktion>

    Wahrscheinlichkeitsrechnung und Statistik, A. Müller, HSR [↑](#footnote-ref-13)
14. Bild und «Steckbrief» entnommen aus dem Skript zu Wahrscheinlichkeit und Statistik von A. Müller, HSR [↑](#footnote-ref-14)
15. Zentraler Grenzwertsatz: <https://de.wikipedia.org/wiki/Zentraler_Grenzwertsatz> [↑](#footnote-ref-15)
16. Bild und «Steckbrief» entnommen aus dem Skript zu Wahrscheinlichkeit und Statistik von A. Müller, HSR [↑](#footnote-ref-16)
17. Bild und «Steckbrief» entnommen aus dem Skript zu Wahrscheinlichkeit und Statistik von A. Müller, HSR [↑](#footnote-ref-17)
18. JARTA — A Java library to model and fit Autoregressive-To-Anything processes, 10.1109/WSC.2013.6721508 [↑](#footnote-ref-18)
19. <https://numerics.mathdotnet.com/>

    <https://github.com/mathnet/mathnet-numerics> [↑](#footnote-ref-19)