

7

Pseudozufallszahlen

7.1

Die Eigenschaften der Pseudozufallszahlen

Zufallszahlen werden in der Naturwissenschaft und Technik bei stochastischen Simulationen eingesetzt, wie die Beispiele im Buch zeigen. Mit Simulationen werden analytisch gewonnene Ergebnisse überprüft und neue Aussagen über das Verhalten von zufallsbeeinflussten Systemen gewonnen. Letzteres meist dann, wenn es nicht möglich ist, analytische Ergebnisse zu berechnen.

Bei diesen Simulationen kommt es darauf an, zuverlässige Aussagen über das simulierte System zu bekommen. Dabei hängt die Qualität der getroffenen Aussagen von der Qualität der Zufallszahlen und damit von der Qualität des Zufallszahlengenerators ab.

Nach der Art der Erzeugung von Zufallszahlen unterscheidet man zwischen *echten Zufallszahlen* und *Pseudo-Zufallszahlen*.

Echte Zufallszahlen können durch physikalische Verfahren, wie das Werfen einer Münze, eines Würfels oder die Betrachtung der Wege von Neutronen erzeugt werden.

Ihre Vorteile sind, keine Periodizitäten, keine Vorhersagbarkeit der Zufallszahlen aufgrund vorangehender Sequenzen, keine irgendwie geartete Systematik, wodurch Daten absolut sicher verschlüsselt werden können, Gewissheit, dass keine versteckten Korrelationen vorliegen und die Gewissheit, dass die Schwankungen in der Gleichverteilung rein stochastisch sind.

Ihre Nachteile sind, dass physikalische Anordnungen der Wartung und Eichung bedürfen und dass erzeugte Zahlenfolgen sich nicht reproduzieren lassen, was für viele Experimente wünschenswert ist.

Pseudo-Zufallszahlen werden durch spezielle deterministische Algorithmen gewonnen. Sie entstehen also nicht zufällig. Daher auch der Zusatz Pseudo.

Ihr Vorteil ist, dass die meisten Programmiersprachen über Sprachelemente zur Erzeugung von Pseudo-Zufallszahlen verfügen und sie so schnell erzeugbar sind.

Ihr Nachteil ist, dass sie unnatürliche Schwankungen in der Gleichverteilung aufweisen.

Alle praktisch bedeutenden Zufallszahlengeneratoren für Pseudo-Zufallszahlen arbeiten rekursiv. Das bedeutet, sie erzeugen

- zuerst eine gewisse Anfangsfolge,
- danach einen nullten Zyklus, also eine Folge, die im ersten Zyklus exakt wiederholt wird usw.

Ein sehr einfacher Algorithmus nach diesem Prinzip ist die folgende Rechenvorschrift:

1. Beginne mit

$$x_0 = 3$$

2. Jede weitere Zufallszahl entsteht aus den beiden letzten Ziffern des Ergebnisses der Multiplikation

$$x_i = [13 \cdot x_{i-1}]_2$$

Es ergibt sich die Zahlenfolge 3,39,7,91,83,79,27,51,63,19,47,11,43,59,67,71,23,99,87,31. Danach beginnt die Zahlenfolge wieder bei 3.

Diese Zahlen weisen eine Gleichverteilung auf dem Intervall von 0 bis 99 auf. Verständlicher Weise gibt es Unterschiede in den Eigenschaften der Pseudo-Zufallszahlen, die zwischen dieser einfachen Zahlenfolgen und einer, die aus einem anderen Algorithmus mit mehreren hundert Pseudo-Zufallszahlen erzeugt wurde.

Nachfolgend möchte ich nicht die unterschiedlichen Methoden zur Erzeugung von Pseudo-Zufallszahlen besprechen, da uns eine Methode in der Programmiersprache VBA vorliegt. Es stellt sich die Frage, wie gut ein Zufallszahlengenerator Pseudo-Zufallszahlen generiert, gemessen an echten Zufallszahlen.

Auch dazu gibt es unterschiedliche Methoden. Nachfolgend betrachten wir stellvertretend den Chi-Quadrat-Test (kurz χ^2 -Test). Dieser betrachtet die Verteilung vorliegender Zahlen. Die Pseudo-Zufallszahlen werden dazu entsprechend ihrer Größe einer von n Kategorien zugeteilt. Da die Pseudo-Zufallszahlen unter VBA im halboffenen Intervall $[0,1)$ durch die Programmfunktion Random erzeugt werden, wird eine Pseudo-Zufallszahl x_i der Kategorie k nach der Formel

$$\frac{k-1}{n} \leq x_i \leq \frac{k}{n} \quad (7.1.1)$$

zugeordnet.

Der χ^2 -Wert ist wie folgt definiert. Ist w_k die Wahrscheinlichkeit, dass eine Zahl in die Kategorie k fällt, m die Anzahl der getesteten Zahlen sowie deren Produkt

$$h_e = w_k \cdot m \quad (7.1.2)$$

die erwartete Anzahl Zahlen in dieser Kategorie, so wie h_k die Anzahl der tatsächlichen Zahlen der Kategorie, so ist

$$\chi^2 = \sum_{k=1}^n \frac{(h_k - h_e)^2}{h_e} \quad (7.1.3)$$

Die Wahrscheinlichkeit für den Wert von χ^2 bestimmt sich aus der Anzahl der Freiheitsgrade v . Sie beträgt für n Kategorien $v=n-1$, da $h_1+h_2+\dots+h_n = m$ ist. Die so erhaltenen Werte lassen eine Aussage darüber zu, ob die beobachteten Zahlen zufällig sind. Dazu ist

jedoch eine hinreichende Anzahl von Zahlen erforderlich.

Als Faustregel für m gilt, dass für jede Klasse der Erwartungswert der Anzahl Beobachtungen mindestens fünf ist, besser größer.

Betrachten wir zur Vertiefung ein einfaches Würfelexperiment. Mit 500 Würfeln erhalten wir das in der nachfolgenden Tabelle dargestellte Ergebnis.

Tab. 7.1 Beispiel zur Häufigkeitsverteilung gewürfelter Zahlen

Gewürfelte Zahl	Häufigkeit
1	81
2	87
3	84
4	86
5	82
6	80
Summe	500

Die Wahrscheinlichkeit, mit der eine Zahl gewürfelt wird, beträgt $1/6$. Somit beträgt der Erwartungswert für die 6 Kategorien

$$h_e = \frac{500}{6} = 83\frac{1}{3}$$

Der χ^2 -Test prüft, ob die reale Verteilung über die Kategorien h_k dem Erwartungswert h_e und damit einer Gleichverteilung entspricht.

In der Literatur wird

$$h_k = h_e$$

als Nullhypothese bezeichnet. Für jede Kategorie ist dann die Nullhypothese zu prüfen. Wie bereits oben erwähnt, kann der Test mit dem χ^2 -Wert direkt für alle Kategorien geprüft

werden. Dazu wird die quadrierte Differenz von h_k und h_e gebildet und normiert

$$\frac{(h_k - h_e)^2}{h_e},$$

so dass deren Summe das Ergebnis liefert. Siehe Formel (7.1.3).

Tab. 7.2 Auswertung der gewürfelten Zahlen

Gewürfelte Zahl	Häufigkeit		Normierte quadrierte Abweichung
1	81		0,0653
2	87		0,1614
3	84		0,0053
4	86		0,0854
5	82		0,0213
6	80		0,1333
Summe	500	Chiquadrat	0,4720
Erwartungswert	83,333		

Das 0,95-Quantil der χ^2 -Verteilung mit 5 Freiheitsgraden ist

$$\chi^2_{0,95(5)} = 11,07$$

laut Tabelle 7.3.

Da

$$0,4720 < 11,070$$

ist, trifft die Nullhypothese zu, d. h. die Verteilung entspricht der Erwartung und der verwendete Würfel erzeugt gute Zufallswerte.

Tab. 7.3 α -Quantil der χ^2 -Verteilung mit f Freiheitsgraden

f / 1- α	.900	.950	9.75	990	.995	.999
1	2.71	3.84	5.02	6.63	7.88	10.83
2	4.61	5.99	7.38	9.21	10.60	13.82
3	6.25	7.81	9.35	11.34	12.84	16.27
4	7.78	9.49	11.14	13.28	14.86	18.47
5	9.24	11.07	12.83	15.09	16.75	20.52
6	10.64	12.59	14.45	16.81	18.55	22.46
7	12.02	14.07	16.01	18.48	20.28	24.32
8	13.36	15.51	17.53	20.09	21.95	26.12
9	14.68	16.92	19.02	21.67	23.59	27.88
10	15.99	18.31	20.48	23.21	25.19	29.59
11	17.28	19.68	21.92	24.72	26.76	31.26
12	18.55	21.03	23.34	26.22	28.30	32.91

Betrachten wir nun die Pseudozufallszahlen unter VBA in unserem Rechner. Unter Vorgabe der Anzahl Zahlen und der Anzahl Kategorien erhalten wir den nachfolgenden Algorithmus.

Tab. 7-4 Chiquadrat-Test für VBA Pseudo-Zufallszahlen

Eingabe von m und n
k = 1, 1, m
Erzeuge Pseudozufallszahl x_i
Ordne die Zufallszahl einer Kategorie zu
$\frac{k-1}{n} < x_i < \frac{k}{n}$
Bestimme die normiert quadrierten Abweichungen
$\frac{(h_k - h_e)^2}{h_e}$
Bestimme Chiquadrat
$\chi^2 = \sum_{k=1}^n \frac{(h_k - h_e)^2}{h_e}$

Code 7-1 Chiquadrat-Test für VBA Pseudo-Zufallszahlen

```
Option Explicit

Sub Chiquadrat_Test()
    Dim i, m, n, k, s, a() As Long
    Dim x, he, chi_2 As Double

    ThisWorkbook.Worksheets("Chiquadrat").Cells.Clear

    m = InputBox("Anzahl der Pseudo-Zufallszahlen: ")
    n = InputBox("Anzahl der Kategorien: ")

    Cells(1, 4) = m
    Cells(2, 4) = n
    ReDim a(n)

    Randomize
    x = Timer

    'Erzeugung der Pseudo-Zufallszahlen
    'und Kategorie-Zuordnung
    s = 0
    For k = 1 To m
        x = Rnd(x)
        For i = 1 To n
            If x <= (i / n) Then
                a(i) = a(i) + 1
                Cells(i, 1) = a(i)
                s = s + a(i)
                i = n
            End If
        Next i
    Next k
    Cells(n + 1, 1) = s
    he = s / n
    Cells(n + 3, 1) = he

    'Bestimmung der normierten
    'quadrierten Abweichungen
    'und Chiquadrat
    s = 0
    For i = 1 To n
        chi_2 = (a(i) - he) ^ 2 / he
        Cells(i, 3) = chi_2
        s = s + chi_2
    Next i
End Sub
```

```
Cells(n + 1, 3) = s
End Sub
```

Ein Programmlauf mit 10000 Pseudo-Zufallszahlen und 12 Kategorien liefert das unter Abbildung 7-1 dargestellte Ergebnis.

	A	B	C	D
1	825		0,08333333	10000
2	850		0,33333333	12
3	827		0,04813333	
4	861		0,91853333	
5	854		0,51253333	
6	854		0,51253333	
7	824		0,10453333	
8	820		0,21333333	
9	852		0,41813333	
10	805		0,96333333	
11	799		1,41453333	
12	829		0,02253333	
13	10000		5,5448	
14				
15	833,333333			
16				

Abb. 7-1 Testergebnis eines Programmlaufs

Hier liefert das Programm

$$5,5448 < 19,675$$

und damit eine gute Gleichverteilung.

7.2

Integration nach der Monte Carlo Methode

Die Prozedur zur Berechnung eines Viertelkreises nach der Monte-Carlo-Methode soll mit unterschiedlichen Laufwerten gestaltet werden. Am einfachsten ist dies mit der Anweisung Inputbox zu lösen.

Code 7-2 Flächenberechnung eines Viertelkreises nach der Monte-Carlo-Methode

```
Option Explicit

Sub MonteCarlo_Test()
    Dim s, x, y, z As Double
    Dim n, m As Long
    Dim i, j As Long
    Dim c As String

    ThisWorkbook.Worksheets("MonteCarlo").Cells.Clear
    n = InputBox("Anzahl der Iterationen:")
    Cells(1, 1) = "Anzahl der Iterationen = "
    Cells(1, 2) = n
    Cells(2, 1) = "Exakter Wert:"
    Cells(2, 2) = Atn(1)
    Cells(4, 1) = "Berechnungen:"
    Cells(4, 2) = "Differenzen:  "
    Randomize
    z = Timer

    For i = 1 To 20
        m = 0
        For j = 1 To n
            x = Rnd(z)
            y = Rnd(z)
            If x * x + y * y < 1 Then
                m = m + 1
            End If
        Next j
        Cells(i + 4, 1) = Str(m / n)
        Cells(i + 4, 2) = Atn(1) - Cells(i + 4, 1)
    Next i

    'Mittelwert
    s = 0
    For i = 5 To 24
        s = s + Cells(i, 1)
    Next i
    Cells(26, 1) = "Mittelwert:"
    Cells(26, 2) = s / 20

    'Minimum
    s = Abs(Cells(5, 2))
    j = 5
    For i = 6 To 24
        If Abs(Cells(i, 2)) < s Then
```

```

        s = Abs(Cells(i, 2))
        j = i
    End If
Next i
c = "A" + LTrim(Str(j)) + ":B" + LTrim(Str(j))
Range(c).Cells.Activate
End Sub

```

Gleichzeitig werden die Differenzen zum exakten Wert bestimmt. Ebenso wird abschließend der Mittelwert berechnet und die minimale Differenz zum exakten Wert markiert.

	A	B
1	Iterationen:	100000
2	Exakter Wert:	0,785398163
3		
4	Berechnungen:	Differenzen:
5	0,78676	-0,001361837
6	0,78734	-0,001941837
7	0,78472	0,000678163
8	0,78653	-0,001131837
9	0,78291	0,002488163
10	0,78448	0,000918163
11	0,78385	0,001548163
12	0,78398	0,001418163
13	0,78609	-0,000691837
14	0,78627	-0,000871837
15	0,78573	-0,000331837
16	0,78675	-0,001351837
17	0,78506	0,000338163
18	0,78684	-0,001441837
19	0,78626	-0,000861837
20	0,78863	-0,003231837
21	0,78194	0,003458163
22	0,78493	0,000468163
23	0,78286	0,002538163
24	0,78424	0,001158163
25		
26	Mittelwert:	0,7853085
27		

Abb. 7-2 Testergebnisse zur Monte-Carlo-Methode

Wie das aussehen kann, zeigt die Abbildung 7.2. Die Zufallszahlen sollten natürlich immer andere sein.

Dem Anwendungsbeispiel Blechteil soll die exakte Berechnung hinzugefügt werden. Dazu ist der Flächeninhalt eines Kreisabschnitts zu berechnen.

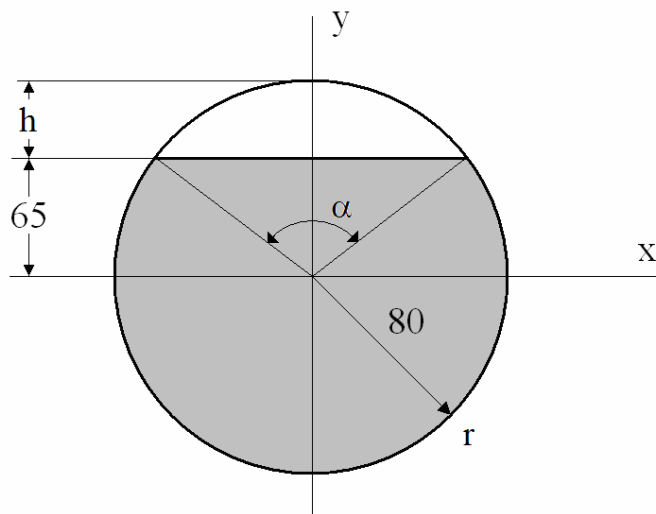


Abb. 7-3 Kreisabschnitt

Der Mittelpunktswinkel α errechnet sich aus der Formel

$$\cos(\alpha) = \frac{r-h}{r}. \quad (7.2.1)$$

Der gesuchte Flächeninhalt ist die Differenz aus Kreisfläche abzüglich der Kreissegmentfläche und folgt aus

$$A = \frac{r^2}{2} (\alpha - \sin(\alpha)). \quad (7.2.2)$$

Die neue Prozedur finden Sie in der Codeliste 7-3. Das Ergebnis der Berechnung zeigt Abbild 7-4.

Code 7-3 Flächenberechnung des Blechteils nach Abbildung 7-3

```

Option Explicit
Sub MonteCarlo_Blechteil()
    Dim x, y, z, e, a, r, h, s As Double
    Dim Pi, Aq, alpha, Ak As Double
    Dim n, m As Long
    Dim i, j As Long
    Dim c As String

    ThisWorkbook.Worksheets("Blechteil").Cells.Clear
    n = InputBox("Anzahl Durchläufe:")
    Cells(1, 1) = "Iterationen:"
    Cells(1, 2) = n
    Randomize
    Pi = Atn(1) * 4
    Aq = 160 * 160
    z = Timer
    e = 1E-308
    r = 80
    h = 80 - 65
    x = (r - h) / r

    ' Arkuskosinus(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
    alpha = 2 * (Atn(-x / Sqr(-x * x + 1)) + 2 * Atn(1))
    Ak = r * r * (Pi - 1 / 2 * (alpha - Sin(alpha)))
    Cells(2, 1) = "Fläche:"
    Cells(2, 2) = Ak
    Cells(4, 1) = "Berechnungen:"
    Cells(4, 2) = "Differenzen:"
    For i = 1 To 20
        m = 0
        For j = 1 To n
            x = (160 + e) * Rnd(z) - 80
            y = (160 + e) * Rnd(z) - 80
            If Sqr(x * x + y * y) <= 80 Then
                If y <= 65 Then
                    m = m + 1
                End If
            End If
        Next j
        Cells(i + 4, 1) = Str(Aq * m / n)
        Cells(i + 4, 2) = Abs(Cells(2, 2) - Cells(i + 4, 1))
    Next i
    'Mittelwert
    s = 0
    For i = 5 To 24

```

```

    s = s + Cells(i, 1)
Next i
Cells(26, 1) = "Mittelwert:"
Cells(26, 2) = s / 20
'Minimum
s = Abs(Cells(5, 2))
j = 5
For i = 6 To 24
    If Abs(Cells(i, 2)) < s Then
        s = Abs(Cells(i, 2))
        j = i
    End If
Next i
c = "A" + LTrim(Str(j)) + ":B" + LTrim(Str(j))
Range(c).Cells.Activate
End Sub

```

	A	B
1	Iterationen:	1000
2	Fläche:	19154,4328
3		
4	Berechnungen:	Differenzen:
5	19353,6	199,167152
6	18841,6	312,832848
7	18739,2	415,232848
8	19097,6	56,8328485
9	19712	557,567152
10	18764,8	389,632848
11	18713,6	440,832848
12	19532,8	378,367152
13	19251,2	96,7671515
14	19276,8	122,367152
15	18969,6	184,832848
16	20019,2	864,767152
17	18688	466,432848
18	19532,8	378,367152
19	18739,2	415,232848
20	19276,8	122,367152
21	18867,2	287,232848
22	19123,2	31,2328485
23	18432	722,432848
24	18944	210,432848
25		
26	Mittelwert:	19093,76
27		

Abb. 7-4 Ergebnisse zur Blechteilberechnung

7.3

Simulation

Eigentlich hätte ich in meinem Buch über Algorithmen der Simulation ein eigenes Kapitel geben müssen, denn es kommt ihr eine immer größere Bedeutung zu. So möchte ich an dieser Stelle etwas ausführlicher darauf eingehen und jeweils ein klassisches Beispiel der unterschiedlichen Methoden zeigen. Neben den traditionellen Methoden der Theorie und des Experiments, stellt die Simulation eine neue Disziplin dar. Diese Erkenntnis ist zwar neueren Datums und dennoch verglichen mit der Entwicklung der Computer relativ alt. Schon in den 50er Jahren hat John von Neumann ein erstes Simulationsverfahren entwickelt.

Wie man den Abhandlungen der heutigen Zeit entnehmen kann, ist die angewandte Mathematik eine Schlüsselressource heutiger Wissenschaften und mit ihr auch die Simulation. Mathematik, Informatik und Disziplinwissenschaft bilden die Grundlage jeder Simulation. Dies gilt in besonderem Maße auch für die Ingenieurwissenschaften. Hier unterscheiden wir hauptsächlich deterministische, stochastische und quanten-mechanische Simulationen. Aber auch andere Methoden finden ihre Anwendung, und dies wird in der nächsten Zeit in erheblichem Maße fortschreiten.

Simulation und Computerentwicklung sind bis heute eng miteinander verbunden. Auch heute noch treibt die Simulation mit ihrem großen Bedarf an Speicherkapazität und Rechenleistung die Weiterentwicklung der Computer und die Entwicklung neuer Methoden an. Bezeichnend für die Simulation ist, dass oft Erkunden und Lernen eng miteinander verbunden sind.

7.3.1

Deterministische Simulationen

Weit verbreitet sind deterministische Simulationen. Ein solches Simulationsmodell besteht aus diskreten Gleichungssystemen, die den Zustand des Modells zum Startbeginn wiedergeben (Zustandsgrößen), die Randbedingungen des Modells sowie die

Anfangsbedingungen zur Initialisierung beschreiben.

Zudem wird dieses Simulationsmodell durch Parameter beeinflusst, die ihren Wert während der Simulation nicht verändern. Doch sie lassen sich für jeden Simulationslauf neu einstellen und ermöglichen so die Anpassung der Simulation.

Allen Gleichungstypen voran werden partielle Differentialgleichungen zur Simulation sich entwickelnder, räumlicher und zeitlicher Systeme in den Naturwissenschaften und Ingenieurwissenschaften eingesetzt. Die deterministische Simulation wird als numerische Lösung eines Differentialgleichungssystems, für das aufgrund seiner Komplexität keine analytische Lösung bekannt ist, gerne verwendet. Sie ist aber lediglich eine Approximation der exakten Lösung.

Bereits John von Neumann hat diesen Typ von Simulationen entwickelt, um die Stagnation in der analytischen Behandlung zahlreicher Probleme der Wissenschaften zu überwinden. 1945 zeigte er erstmals in der numerischen Hydrodynamik neue Wege auf, indem er vorschlug, die analytischen Methoden durch numerische zu ersetzen. Zusammen mit Athur Burks und Herman Goldstine konzipierte er in dieser Zeit die nach ihm benannte Rechnerarchitektur, die dem Computer zum Durchbruch verhalf.

Beispiel: Die Keplerschen Gesetze

Gerne verwende ich bei meinen Vorträgen dieses Beispiel, da es auf anschauliche Weise die deterministische Simulation zeigt und sich gleichzeitig, quasi nebenher, die Keplerschen Gesetze ableiten lassen.

Betrachten wir zuerst das Kepler-Problem.

Es ist das Problem der Bewegung eines Massenpunktes unter dem Einfluss einer Zentralkraft, die dem Quadrat des Abstandes vom Kraftzentrum umgekehrt proportional ist.

$$F = -\frac{\alpha}{r^2} \quad 7.3.1$$

Darin ist α eine Konstante. Ist $\alpha < 0$, so wird der Punkt vom Zentrum abgestoßen (Coulombsches Wechselwirkung gleichnamiger Punktladungen). Ist $\alpha > 0$, so wird der Punkt vom Zentrum angezogen (Gravitationswechselwirkung von Massenpunkt und Coulombsche Wechselwirkung ungleichnamiger Punktladungen).

Die Bahnkurven des Punktes sind Kegelschnitte, deren Gleichungen in Polarkoordinaten r und $\psi = \varphi - C_2$ folgende Form haben:

$$r = \frac{p}{1 + e \cdot \cos \psi}, (\alpha > 0)$$

$$r = \frac{p}{-1 + e \cdot \cos \psi}, (\alpha < 0)$$

$$p = \frac{L^2}{m \cdot |\alpha|} \quad 7.3.2$$

$$e = \sqrt{1 + \frac{2 \cdot W \cdot L^2}{m \cdot \alpha^2}}$$

p ist der Bahnparameter, e die Exentrität der Bahn, W ist die Gesamtenergie des Punktes und L ist sein Drehimpuls in Bezug auf das Zentrum.

Für $\alpha < 0$ liegt das Kraftzentrum im Brennpunkt der Bahn des Massenpunktes. Sein Abstand zum nächsten Bahnpunkt wird als Perihel bezeichnet und ist

$$r_{\min} = \frac{p}{1 + e}. \quad 7.3.3$$

Ist $W > 0$, dann ist $e > 1$ und die Bahn ist eine Hyperbel. Ist $W = 0$, dann ist $e = 1$ und die Bahn ist eine Parabel. Ist schließlich $W < 0$, dann ist $e < 1$ und die Bahn ist eine Ellipse, deren Achsen durch

$$a = \frac{p}{1-e^2} = \frac{\alpha}{2 \cdot |W|} \quad 7.3.4$$

$$b = \frac{p}{\sqrt{1-e^2}} = \frac{L}{\sqrt{2 \cdot m \cdot |W|}}$$

gegeben sind. Kreist der Punkt auf einer elliptischen Bahn, so gilt für die Bahnperiode

$$T = 2 \cdot \pi \cdot \sqrt{\frac{m \cdot a^3}{\alpha}} \quad 7.3.5$$

(drittes Keplersches Gesetz).

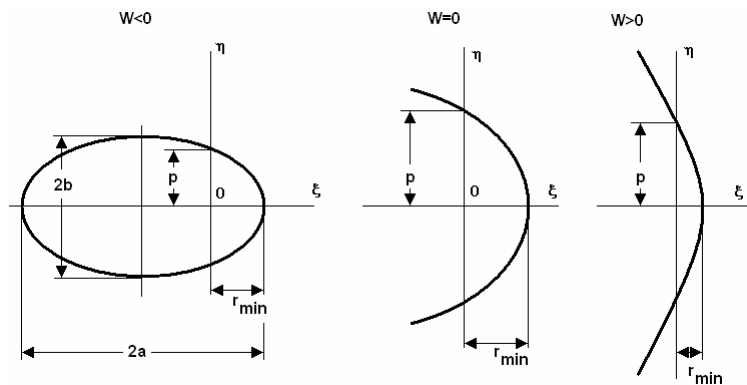


Abb. 7-5 Bahnformen

Kommen wir nun zum eigentlichen Algorithmus.

Zur numerischen Bestimmung eines Bewegungsablaufs bedarf es eines exakten endlichen Algorithmus. Zu diesem gelangt man durch die Betrachtung zweier Zustände des Systems, die um die Zeitdifferenz Δt auseinander liegen. Sodann versucht man, die Zustandsänderungen durch Gesetzmäßigkeiten in Bezug auf den ersten Zustand vor der Veränderung auszudrücken.

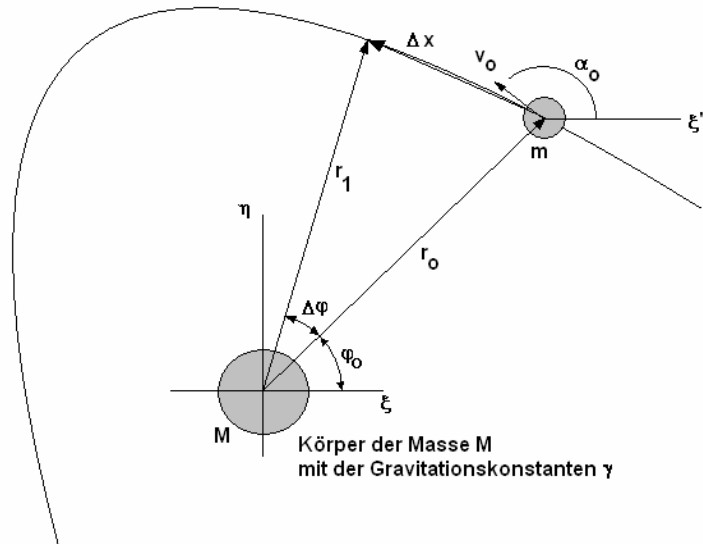


Abb. 7-6 Bahnbewegung

Unter Betrachtung von Abbildung 7-6 erhalten wir aus Massenanziehung, Trägheitskraft und Gravitationskonstante γ

$$F = \gamma \cdot \frac{M \cdot m}{r^2} = m \cdot a \quad 7.3.6$$

und

$$a = \frac{\gamma \cdot M}{r^2} = \frac{dv}{dt} \quad 7.3.7$$

Drücken wir nun den für die Numerische Mathematik unbrauchbaren Differentialquotient durch den näherungsweise Differenzenquotienten aus, so erhalten wir

$$\frac{dv}{dt} = \frac{\Delta v}{\Delta t}, \text{ für sehr kleines } \Delta t \quad 7.3.8$$

und damit

$$\Delta v = \frac{\gamma \cdot M}{r^2} \cdot \Delta t \quad 7.3.9$$

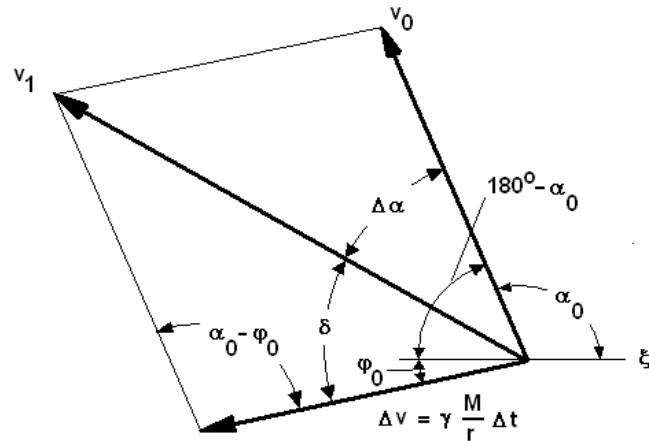


Abb. 7-7 Bahngeschwindigkeiten

Für unsere Geschwindigkeiten ergeben sich unter Betrachtung von Abbildung 7-7 folgende geometrischen Beziehungen.

Die Geschwindigkeit v_1 ergibt sich durch den Kosinussatz

$$v_1 = \sqrt{v_0^2 + \Delta v^2 - 2 \cdot v_0 \cdot \Delta v \cdot \cos(\alpha_0 - \varphi_0)} \quad 7.3.10$$

Den Winkel δ erhalten wir aus dem Sinussatz

$$\Delta \alpha = \arcsin\left(\frac{\Delta v}{v_1} \sin(\alpha_0 - \varphi_0)\right). \quad 7.3.11$$

Durch erneute Anwendung des Kosinussatzes ergibt sich damit auch die neue Entfernung

$$r_1 = \sqrt{r_0^2 + \Delta x^2 - 2 \cdot r_0 \cdot \Delta x \cdot \cos(180^\circ - \alpha_1 + \varphi_0)} \quad 7.3.12$$

Und letztlich ergibt eine nochmalige Anwendung des Sinussatzes den bei der Bewegung überstrichenen Winkel

$$\Delta \varphi = \arcsin\left(\frac{\Delta x}{r_1} \sin(180^\circ - \alpha_1 + \varphi_0)\right) \quad 7.3.13$$

Fassen wir die so errechneten Werte als alte Werte auf und beginnen wir diesen Berechnungsgang von vorn, so erhalten wir in groben Zügen den gesuchten Algorithmus. Zur besseren Anschauung noch einmal als Struktogramm in Tabelle 7.5.

Tab. 7-5 Der Algorithmus des Kepler-Problems

Eingabe $\gamma, M, r_0, \varphi_0, \alpha_0, \Delta t, t_0, t_{\max}, v_0$
Solange $t < t_{\max}$
$\Delta v = \frac{\gamma \cdot M}{r_0^2} \cdot \Delta t$
$v_1 = \sqrt{v_0^2 + \Delta v^2 - 2 \cdot v_0 \cdot \Delta v \cdot \cos(\alpha_0 - \varphi_0)}$
$\Delta \alpha = \arcsin\left(\frac{\Delta v}{v_1} \sin(\alpha_0 - \varphi_0)\right)$
$\alpha_1 = \alpha_0 + \Delta \alpha$
$r_1 = \sqrt{r_0^2 + \Delta x^2 - 2 \cdot r_0 \cdot \Delta x \cdot \cos(180^\circ - \alpha_1 + \varphi_0)}$
$\varphi_1 = \varphi_0 + \Delta \varphi$
$t_1 = t_0 + \Delta t$
Ausgabe von v_1, r_1, φ_1, t_1
Umsetzung alte Werte = neue Werte

Wie immer beginnen wir mit einem Formblatt und Testdaten. Danach erfolgt die Auswertung.

Code 7-4 Das Kepler-Problem

Option Explicit

```
Sub Satellit_Formblatt()
    ThisWorkbook.Worksheets("Satellit").Cells.Clear
    Cells(1, 1) = ChrW(947) & " [m" & ChrW(179) & "/(kg s" &
        ChrW(178) & "]"
    Cells(2, 1) = "M [kg]"
    Cells(3, 1) = "r [m]"
    Cells(4, 1) = ChrW(966) & " [grad]"
    Cells(5, 1) = ChrW(945) & " [grad]"
End Sub
```

```
Cells(6, 1) = "dt [s]"
Cells(7, 1) = "v [m/s]"
Cells(8, 1) = "tmax [s]"
Cells(1, 4) = "t [s]"
Cells(1, 5) = "v [m/s]"
Cells(1, 6) = "r [m]"
Cells(1, 7) = ChrW(945) & " [grd]"
Cells(1, 8) = ChrW(966) & " [grd]"
Range("A1:A8").Select
Selection.Font.Bold = True
Selection.Font.Italic = True
Range("D1:H1").Select
Selection.Font.Bold = True
Selection.Font.Italic = True
End Sub

Sub Satellit_Daten_1()
Cells(1, 2) = 0.0000000000667
Cells(2, 2) = 5.975E+24
Cells(3, 2) = 12471000#
Cells(4, 2) = 0
Cells(5, 2) = 90
Cells(6, 2) = 120
Cells(7, 2) = 5675.83 'v für Kreisbahn
Cells(8, 2) = 15120
End Sub

Sub Satellit_Daten_2()
Cells(1, 2) = 0.0000000000667
Cells(2, 2) = 5.975E+24
Cells(3, 2) = 12471000#
Cells(4, 2) = 0
Cells(5, 2) = 90
Cells(6, 2) = 120
Cells(7, 2) = 7000 'v für Elliptische Bahn
Cells(8, 2) = 43680
End Sub

Sub Satellit_Auswertung()
Dim g, M, r0, r1 As Double
Dim phi0, phi1, dphi, alpha0, alpha1, dalpha As Double
Dim t0, t1, dt, tmax As Double
Dim dv, v0, v1 As Double
Dim dx, x, wg, wb, pi As Double
Dim i As Integer
```

```
Call Satellit_Formblatt
Call Satellit_Daten

g = Cells(1, 2)
M = Cells(2, 2)
r0 = Cells(3, 2)
phi0 = Cells(4, 2)
alpha0 = Cells(5, 2)
dt = Cells(6, 2)
v0 = Cells(7, 2)
tmax = Cells(8, 2)

pi = 4 * Atn(1)
phi0 = phi0 / 180 * pi
alpha0 = alpha0 / 180 * pi
t0 = 0
i = 1
Do
  dv = g * M / r0 ^ 2 * dt
  x = alpha0 - phi0
  v1 = Sqr(v0 ^ 2 + dv ^ 2 - 2 * v0 * dv * Cos(x))
  x = dv / v1 * Sin(x)
  dalpha = Atn(x / Sqr(-x * x + 1))
  alpha1 = alpha0 + dalpha
  dx = v1 * dt
  x = pi - alpha1 + phi0
  r1 = Sqr(r0 ^ 2 + dx ^ 2 - 2 * r0 * dx * Cos(x))
  x = dx / r1 * Sin(x)
  dphi = Atn(x / Sqr(-x * x + 1))
  phi1 = phi0 + dphi
  t1 = t0 + dt
'Ausgabe
  i = i + 1
  Cells(i, 4) = t1
  Cells(i, 5) = v1
  Cells(i, 6) = r1
  Cells(i, 7) = alpha1 / pi * 180
  Cells(i, 8) = phi1 / pi * 180
'neu-alt-Verschiebung
  t0 = t1
  v0 = v1
  r0 = r1
  phi0 = phi1
  alpha0 = alpha1
Loop While t1 < tmax
End Sub
```

Wir betrachten nun einen Erdsatelliten, der sich mit konstanter Winkelgeschwindigkeit ω auf einer Bahn um die Erde bewegt. Der Satellit besitzt die Zentripetalbeschleunigung

$$b = -\omega^2 \cdot \bar{r} \quad 7.3.14$$

Darin ist r der Ortsvektor vom Erdmittelpunkt zum Satelliten. Nach dem Grundgesetz der Mechanik in Verbindung mit dem Gravitationsgesetz ist die Beschleunigung andererseits

$$b = -\frac{\gamma \cdot M}{r^3} \cdot \bar{r} \quad 7.3.15$$

Daraus erhält man

$$\omega^2 \cdot r^3 = \gamma \cdot M \quad 7.3.16$$

Ist v die Bahngeschwindigkeit des Satelliten, so wird

$$v = \omega \cdot r \quad 7.3.17$$

Für r setzen wir $R + h$, mit R als Erdradius und h als die Flughöhe des Satelliten über der Erde. Dann ergibt sich

$$v = \sqrt{\frac{\gamma \cdot M}{R + h}} \quad 7.3.18$$

Die Auswertung liefert nachfolgende Daten.

	A	B	C	D	E	F	G	H
1	γ [$m^3/(kg \cdot s^2)$]	6,67E-11		t [s]	v [m/s]	r [m]	α [gra]	φ [gra]
2	M [kg]	5,975E+24		120	5684,15352	12452740,5	93,1010682	3,13534324
3	r [m]	12471000		240	5692,32947	12434862,3	96,2067742	6,27980742
4	φ [gra]	0		360	5700,33206	12417420,7	99,3170469	9,43322465
5	α [gra]	90		480	5708,13571	12400469,8	102,4318	12,5953967
6	dt [s]	120		600	5715,7152	12384062,6	105,55093	15,7660952
7	v [m/s]	5675,83		720	5723,04569	12368251,1	108,67432	18,9450617
8	tmax [s]	15120		840	5730,10293	12353085,5	111,801835	22,1320081
9				960	5736,86326	12338614,4	114,933327	25,3266174
123				14640	5715,07458	12385447,1	465,024042	375,234626
124				14760	5722,42739	12369582,5	468,147072	378,412894
125				14880	5729,50899	12354359,6	471,274239	381,599168
126				15000	5736,2957	12339827	474,405396	384,793133
127				15120	5742,76456	12326031,8	477,540379	387,994447
128								

Abb. 7-8 Auswertung mit Beispieldaten

Anschaulicher zeigt Abbildung 7-9 den Bahnverlauf. Darin lassen sich die drei Keplerschen Gesetze anschaulich ablesen.

1. Gesetz: Die Planeten bewegen sich auf elliptischen Bahnen, in deren einem Brennpunkt die Sonne steht.

In unserem Fall ist die Sonne die Erde und der Planet ein Satellit. Die Summe der Abstände eines Bahnpunktes einer Ellipse zu den beiden Brennpunkten ist immer konstant. Die grünen Linien zeigen dies.

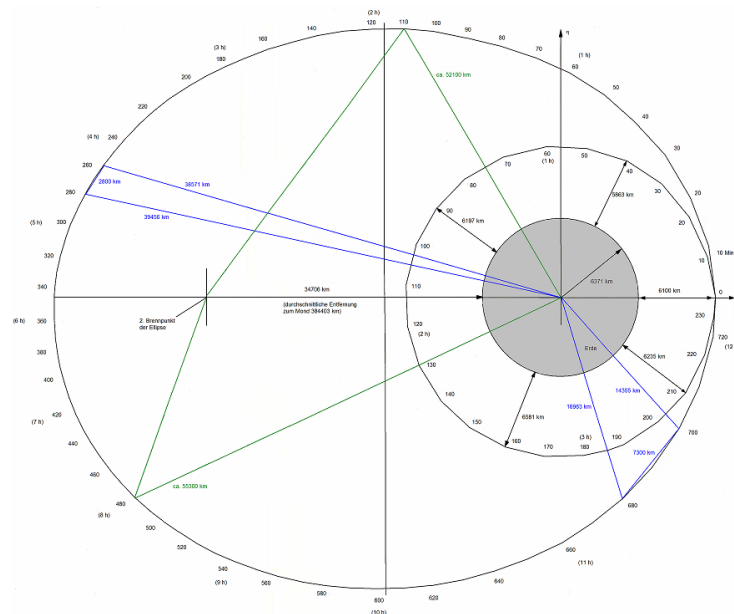


Abb. 7-9 Beispiel-Bahnen

2. Gesetz: Die Flächengeschwindigkeit eines Punktes ist konstant.

Für die blau gezeichneten Flächen ergibt sich aus den Formeln für ein unregelmäßiges Dreieck (Satz des Heron):

Von 260 bis 280 Min.: $A=51785508,51 \text{ km}^2$

Von 680 bis 700 Min.: $A=51889417,39 \text{ km}^2$

3. Gesetz: Das Verhältnis der dritten Potenzen der großen Halbachsen der Bahnen zu den Quadraten der Umlaufzeiten ist konstant.

Also:

$$\frac{T_1^2}{a_1^3} = \frac{T_2^2}{a_2^3}. \quad 7.3.14$$

In unserem Fall:

$$\frac{T_1^2}{a_1^3} = \frac{43680^2 s^2}{26774^3 km^3} = 9,94 \cdot 10^{-5} \frac{s^2}{km^3}$$

$$\frac{T_2^2}{a_2^3} = \frac{14100^2 s^2}{12593^3 km^3} = 9,95 \cdot 10^{-5} \frac{s^2}{km^3}$$

Mit Hilfe dieses Programms lassen sich die Bewegungen von Satelliten im Gravitationsfeld eines Planeten berechnen.

So lässt sich auch zeigen, dass einem Körper die erste kosmische Geschwindigkeit von

$$v = 7,9 \frac{km}{s}$$

erteilt werden muss, damit er zu einem Erdsatelliten mit kreisförmiger Umlaufbahn wird. Und auch, dass es der zweiten kosmischen Geschwindigkeit von

$$v = 11,2 \frac{km}{s}$$

bedarf, damit der Körper die irdische Gravitation überwindet und zu einem Satelliten der Sonne werden kann.

Ähnlich funktioniert die Flugbahn-Simulation einer Rakete, die ich dem Leser überlasse.

7.3.2 Stochastische Simulationen (Monte-Carlo Simulationen)

In stochastischen Simulationen wird der Prozessablauf durch zufällige Ereignisse beeinflusst.

Diese Zufallsabhängigkeiten werden in die Simulation integriert, so dass wiederholte Simulationsläufe - im Gegensatz zur deterministischen Simulation - zu unterschiedlichen Resultaten führen. Das ermöglicht auf Basis der Wahrscheinlichkeitstheorie zufällige Schwankungen, die der Quantifikation durchschnittlicher Verläufe dienen, wie sie Experimentreihen ähnlich sind. Zudem lassen sich Faktoren, die ansonsten im Zuge der Modellvereinfachung eliminiert würden, durch stochastische Variablen berücksichtigen. Nahezu alle physikalischen Größen, die in der Realität als Streuungen vorhanden sind, sind integrierbar.

Stochastische Simulationen sind nützlich, um auf der Grundlage statistischer Daten durch Stichprobenerhebungen aus der Vergangenheit, Entscheidungshilfen für die Zukunft zu prognostizieren, die jedoch immer nur eine mehr oder minder hohe Wahrscheinlichkeiten aufweisen. Es gibt unterschiedliche Arten von stochastischen Simulationsalgorithmen. Der Kern jeder stochastischen Simulation ist jedoch die Erzeugung von Zufallszahlen. Computerbasierte Zufallszahlen-Generatoren können letztendlich nur Pseudo-Zufallszahlen erzeugen und frühe, sehr einfache Generatoren wiesen sich wiederholende Muster auf.

Um die zeitliche Entwicklung von Prozessen stochastisch zu simulieren werden heutzutage meist Algorithmen der Markov-Chain-Monte-Carlo-Simulation verwendet, die auch in der statistischen Analyse von Bilddaten zum Einsatz kommen. Beliebte Anwendungsbereiche sind die Analyse, Planung, Steuerung und Optimierung von Fertigungsprozessen, Auslastungsbetrachtungen, Warteschlangenprobleme, Fahrzeugsimulationen, Stichprobenanalysen und anderes mehr.

Beispiel: Tankstelle

Ein immer wieder gern dargestelltes Modell ist der Betrieb an einer Tankstelle. An diesem einfachen Beispiel lässt sich sehr schön der Aufbau einer stochastischen Simulation zeigen.

An einer Tankstelle kommen zufällig Autos an und nehmen wir an, dass es dort zwei Zapfsäulen gibt. Sind diese besetzt, müssen die Autos warten. So bilden sich Schlangen vor den Zapfsäulen. Dieses Beispiel wird auch als einfaches Warteschlangenproblem bezeichnet.

Die Ankunfts Wahrscheinlichkeit wurde zuvor durch eine Zählung ermittelt (z. B. 12 Autos je Stunde). Ebenso wurde die durchschnittliche Auftankzeit bestimmt (z. B. 6 Minuten). Mit Hilfe einer Simulation soll nun die durchschnittliche und maximale Aufenthaltsdauer eines Autos in der Tankstelle bestimmt werden.

Betrachten wir dieses Modell nun im Minutentakt. Es gibt keine festen Regeln für die Größe des Intervalls. Es muss einerseits klein genug sein für eine vernünftige Simulation und darf andererseits nicht zu viel Rechenzeit kosten.

Die Wahrscheinlichkeit, mit der ein Auto pro Minute ankommt, wird durch das Verhältnis

$$w = \frac{12 \text{ Autos}}{60 \text{ Minuten}} = \frac{2}{10} = 0,2$$

ausgedrückt. Diese Wahrscheinlichkeit wird durch Pseudozufallszahlen simuliert. Wir nutzen dabei die Gleichverteilung dieser Zahlen auf dem Intervall $[0,1)$, siehe Kapitel 7.1. Die pro Zeiteinheit (1 Minute) erzeugte Zahl generiert ein ankommendes Auto, wenn ihr Wert kleiner oder gleich 0,2 ist.

Damit sind wir nun in der Lage, die Simulation einer Tankstelle als Rechnermodell durchzuführen. Dazu benötigen wir die Zapfsäulenbelegungen und eine Warteschlange.

Tab. 7-6 Simulation einer Tankstelle

Eingabe t_{\max}		
$t=1, 1, t_{\max}$		
Erzeugung einer Zufallszahl x		
Ist $x \leq 0,2$		
Ja		Nein
Warteschlange $W=W+1$./.
Ist $Z1 > 0$		
Ja		Nein
Ist $W > 0$		
$Z1=Z1-1$		Ja
		Nein
		./.
$Z1=6$./.
$W=W-1$		
Ist $Z2 > 0$		
Ja		Nein
Ist $W > 0$		
$Z2=Z2-1$		Ja
		Nein
		./.
$Z2=6$./.
$W=W-1$		
Ausgabe $t, Z1, Z2, W$		

Für den Prozedurstart benötigen wir lediglich den Betrachtungszeitraum.

Code 7-5 Tankstellen-Simulation

Option Explicit

```

Sub Tankstellen_Simulation()
    Dim x, z As Double
    Dim t, tmax, Z1, Z2, W As Long

    ThisWorkbook.Worksheets("Tankstelle").Cells.Clear
    tmax = InputBox("Zeitintervall in Min.:")

    Randomize
    z = Timer

```

```
For t = 1 To tmax
  'Ankunftswahrscheinlichkeit
  x = Rnd(z)
  If x <= 0.2 Then
    W = W + 1
  End If

  'Zapfsäule 1
  If Z1 > 0 Then
    Z1 = Z1 - 1
  Else
    If W > 0 Then
      Z1 = 6
      W = W - 1
    End If
  End If

  'Zapfsäule 2
  If Z2 > 0 Then
    Z2 = Z2 - 1
  Else
    If W > 0 Then
      Z2 = 6
      W = W - 1
    End If
  End If

  'Ausgabe
  Cells(t, 1) = t
  Cells(t, 2) = Z1
  Cells(t, 3) = Z2
  Cells(t, 4) = W
Next t
End Sub
```

Die Simulation zeigt, dass in der Warteschlange bis zu 7 Autos stehen können (Abbildung 7-10). Allerdings wird die Warteschlange immer wieder abgebaut. Mit höherer Wahrscheinlichkeit wird dies dann nicht mehr der Fall sein und eine dritte Zapfsäule muss eingesetzt werden. Diese und ähnliche Änderungen überlasse ich dem Leser. Ebenso die Veränderungen zu den Beispielen Maschinenwartung und Lebensdauer im Buch.

	A	B	C	D
1	1			0
2	2			0
3	3			0
4	4			0
5	5	6		0
6	6	5		0
7	7	4		0
8	8	3	6	0
9	9	2	5	0
10	10	1	4	1
11	11	0	3	1
12	12	6	2	0
381	381	3	6	6
382	382	2	5	6
383	383	1	4	7
384	384	0	3	7
385	385	6	2	7
386	386	5	1	7
387	387	4	0	7
388	388	3	6	6
389	389	2	5	6
390	390	1	4	7
391	391	0	3	7
392	392	6	2	6
393	393	5	1	6
394	394	4	0	6
395	395	3	6	5
3590	3590	6	1	0
3591	3591	5	0	1
3592	3592	4	6	1
3593	3593	3	5	2
3594	3594	2	4	3
3595	3595	1	3	3
3596	3596	0	2	4
3597	3597	6	1	4
3598	3598	5	0	4
3599	3599	4	6	4
3600	3600	3	5	4

Abb. 7-10 Beispieldaten der Tankstellen-Simulation

7.3.3 Quantenmechanische Simulationen

Quantenmechanische Simulationen erfordern aufgrund der Komplexität der Gleichungen extrem hohe Rechnerkapazitäten. Ebenso muss für jeden Anwendungsfall eine Anpassung erfolgen.

Das Herzstück der Quantenmechanik ist die Schrödinger-Gleichung deren Lösung eine Wellenfunktion ist, die das Verhalten subatomarer Materie als Reaktion auf die wirkenden Wechselwirkungen beschreibt. Sie gibt Einblick in die Struktur und Eigenschaft der Materie sowie der chemischen Eigenschaften auf atomarer Ebene. Elektromagnetische Wechselwirkungen werden mathematisch als potentielle Energien geschrieben, die zur Berechnung in die Wellenfunktion eingesetzt werden. Sie ergeben sich aus der Addition der elektromagnetischen Abstoßung jedes Elektrons durch jedes andere Elektron mit der Anziehung durch den positiven Kern. Genau hier liegt die Komplexität quantenmechanischer Berechnungen.

Beispiele aus diesem Bereich übersteigen den Rahmen dieses Buches.

7.3.4 Begrenzung der computerbasierten Simulationen

Trotz aller Verschiedenheit darf nicht vergessen werden, dass die hier vorgestellten Simulationen allesamt computerbasiert sind. Das bedeutet, dass sie den grundlegenden Kriterien der Rechnertechnik unterliegen. Selbst stochastische Simulationen sind prinzipiell determiniert, da künstlicher Zufall immer regelbasiert ist, wie in Kapitel 7.1 beschrieben. Ein Modell zu algorithmisieren bedeutet immer, entscheidbare Anweisungen und unterscheidbare Systemzustände zu beschreiben, selbst bei hohen Komplexitäts- und Freiheitsgraden im Modell oder im Simulationsverfahren. Die Endlichkeit der Rechner - Ressourcen zwingt dabei zu effektiven Berechnungsverfahren. Der dabei benötigte hohe Zeitaufwand für die Datenerfassung und die Datenaufbereitung darf nicht unerwähnt bleiben.