1 Grundlagen

Die folgenden Simulationen sollen auf dem Bildschirm visualisiert werden (z.B. durch ein zweidimensionales Koordinatensystem).

1.1 Modelle

Der Mensch kann sich von einem Objekt der "Realität" (z.B. elektrischer Widerstand) ein Abbild machen.

	Objekt	Modell
Beispiel 1:	Widerstand	$R = U \cdot I$
Beispiel 2:	Massenbeschleunigung	$F = m \cdot a$

In den Naturwissenschaften sagt man zu diesem Abbild **Modell**. Dieses Modell wird mit formal-operationalen, mathematischen Methoden beschrieben. Um herauszufinden, wie gut dieses Modell das Objekt abbildet, werden an dem Objekt praktische und am Modell theoretische (mathematische Berechnungen) Experimente gemacht.
Das Modell ist desto besser, je mehr sich diese entsprechen. Entsprechen sich die gemachten Experimente nur wenig, dann muss das Modell verändert werden.
Die (theoretischen) Experimente am Modell nennt man auch **Simulationen** (des Objekts).
Simulationen sind besonders wichtig bei Situationen, in denen praktische Experimente am Objekt nicht gemacht werden können oder sollten (z.B. Menschenversuche, Flugzeugabsturz, Versuche mit dem Wetter).
Simulationen von Zufallsexperimenten (das Objekt Zufall steckt auf der untersten Ebene in der Materie drin) durch Zufallsziffern nennt man **Monte-Carlo-Methoden**.

Es gibt verschiedene Modelle um die "Realität" nachzubilden (simulieren):

1.1.1 Spieltheoretisches Modell (chemische Reaktionsspiele)

Zwei Reaktionspartner (Moleküle bzw. Atome) müssen einander begegnen, denn sonst kann keine Reaktion stattfinden. Alle chemischen Reaktionskräfte haben aber nur sehr kurze Reichweiten, deswegen kann diese Begegnung nicht dadurch veranlasst werden.
Die Moleküle bzw. Atome sind in ständiger ungeordneter Bewegung (Brownsche Molekularbewegung) und irren regellos umher (random walk).
Dadurch überstreicht jeder Reaktionspartner im Laufe der Zeit ein bestimmtes Areal, dessen radiale Ausdehnung mit der Quadratwurzel - wie Einstein zum ersten Mal zeigte - aus der Zeit anwächst.
Nicht jeder Zusammenstoß zweier Moleküle bzw. Atome führt zu einer Reaktion. Der Zusammenstoß muß so heftig sein, damit die Moleküle bzw. Atome hinreichend aktiviert werden. (Aktivierungsenergie).

Eine chemische Reaktion kann man durch ein Spiel simulieren:

Wirklichkeit	Modell
Reaktionsgefäß	Urne
Molekül	Kugel
Molekülsorte	Kugelfarbe
Konzentration eines Stoffes	Zahl der Kugeln einer Farbe
Brownsche Bewegung	Durchmischung
Aktivierung durch Stoß	lotterieartige Ziehung
Elementarreaktion	Kugelaustausch
Reaktionsdauer	Zahl der Ziehungen
Aktivierungsenergie	Wahrscheinlichkeitswürfel zur Entscheidung, ob eine gezogene Kugel hinreichend aktiviert ist.

1.1.2 analoges Modell

Mit Hilfe von Differentialgleichungen kann man die Abhängigkeit der Konzentrationen der verschiedenen Stoffe angeben.

2 Simulationstypen

2.1 Biologische Simulationen

Der Biologe D'Ancona beobachtete von 1910 bis 1923 die Fischbestände in der Adria und stellet dabei beachtliche Schwankungen in der Zahl der Nutzfische und der Raubfische fest. D'Anconas Beobachtungen führten Alfred Lotka (amerikanischer Biophysiker) und Vito Volterra (italienischer Mathematiker) dazu ein Modell für solche Systeme aufzustellen.

2.1.1 Räuber - Beute - Modell (Lotka - Volterra)

In einem abgeschlossenen System kommen zwei Tierarten vor: Räuber und Beute.
Dabei soll gelten:
1) Je mehr Beutetiere es hat, desto stärker vermehren sie sich.
Konkrete Modellierung:
Die Zunahme der Beutetiere ist proportional zur Anzahl der Beutetiere.
Formal: (k_1 ist eine Konstante mit Maßeinheit Beutetiere pro Jahr)
$x\_zu = k_1 \cdot x$

2) Je mehr Beutetiere es hat, desto leichter sind sie zu fangen. Je mehr Räuber es hat, desto mehr Beutetiere werden gefangen.
Konkrete Modellierung:
Die Abnahme der Beutetiere ist proportional zur Anzahl der Beutetiere und zur Anzahl der Räuber.
Formal: (k_2 ist eine Konstante mit Maßeinheit Beutetiere pro Jahr pro Räuber)
$x\_ab = k_2 \cdot x \cdot y$

3) Je mehr Räuber und Beutetiere (Futter) es hat, desto stärker vermehren sich die Räuber.
Konkrete Modellierung:
Die Zunahme der Räuber ist proportional zur Anzahl der Beutetiere und zur Anzahl der Räuber.
Formal: (k_3 ist eine Konstante mit Maßeinheit Räuber pro Jahr pro Beutetier)
$y\_zu = k_3 \cdot x \cdot y$

4) Je mehr Räuber es gibt, desto mehr Räuber sterben auch.
Die Abnahme der Räuber ist proportional zur Anzahl der Räuber.
Formal: (k_4 ist eine Konstante mit Maßeinheit Räuber pro Jahr)
$y\_ab = k_4 \cdot y$

5) Anfangsdaten des Modells (empirische Daten):
a) Anfangsbestand der Räuber = 300
b) Anfangsbestand der Beutetiere = 400
c) durchschnittliche Vermehrungsrate k_1 (in Beutetiere / Jahr) = 10
d) durchschnittliche Sterberate k_2 (in Beutetiere pro Jahr und Räuber) = 0,03
e) durchschnittliche Vermehrungsrate k_3 (der Räuber pro Jahr pro Beutetier) = 0,02
f) durchschnittliche Sterberate k_4 (der Räuber / Jahr) = 7

Bezeichnungen:
x(t): Anzahl der Beutetiere
y(t): Anzahl der Räuber.

Damit gilt dann:

$$x'(t) = k_1 \cdot x(t) - k_2 \cdot x(t) \cdot y(t)$$
$$y'(t) = k_3 \cdot x(t) \cdot y(t) - k_4 \cdot y(t)$$

Mit dt wird ein kleiner Zeitabschnitt (z.B. 0.01 Jahre) bezeichnet wird. Wenn man annimmt, dass sich während dieses Zeitabschnitts $x(t_n)$ und $y(t_n)$ nicht ändert, dann gilt die Näherung:

$$dx(t_n) = dt \cdot (k_1 \cdot x(t_n) - k_2 \cdot x(t_n) \cdot y(t_n))$$
$$dy(t_n) = dt \cdot (k_3 \cdot x(t_n) \cdot y(t_n) - k_4 \cdot y(t_n))$$

und damit:

$$x(t_{n+1}) = x(t_n) + dt \cdot (k_1 \cdot x(t_n) - k_2 \cdot x(t_n) \cdot y(t_n))$$
$$y(t_{n+1}) = y(t_n) dt \cdot (k_3 \cdot x(t_n) \cdot y(t_n) - k_4 \cdot y(t_n))$$

in Programmcode übersetzt:

```
dx = dt * (k1 * x  -  k2 * x * y)
dy = dt * (k3 * x * y  -  k4 * y)
x = x + dx
y = y + dy
```

2.1.1.1 Fixpunkte

In diesen Punkten ändern sich die x-Koordinate und die y-Koordinate nicht mehr.
Es gilt also: x'(t) = 0 und y'(t) = 0
also:

$$\begin{aligned} 0 &= k_1 \cdot x(t) - k_2 \cdot x(t) \cdot y(t) \\ 0 &= k_3 \cdot x(t) \cdot y(t) - k_4 \cdot y(t) \end{aligned} \quad \Longrightarrow x_F = \frac{k_4}{k_3} \wedge y_F = \frac{k_1}{k_2} \text{ oder } x_F = 0 \wedge y_F = 0$$

also

$F_1(\frac{k_4}{k_3} \mid \frac{k_1}{k_2})$ und $F_2(0 \mid 0)$

Beim Testen des Simulationsprogramms wird empfohlen die Anfangswerte der x- und y-Koordinate zuerst in der Nähe des nichttrivialen Fixpunkts F_1 zu wählen

2.1.2 Modifiziertes Räuber - Beute - Modell

Um zu experimentieren, könnte man in dem herkömmlichen Räuber - Beute- Modell noch die Jäger (Menschen) integrieren.
Im Frühjahr, wenn die Tiere trächtig sind, oder Nachwuchs haben, dürfen keine Tiere geschossen werden. Im Laufe des Jahres dürfen dann immer mehr geschossen werden, bis dann wieder beim nächsten Frühjahr keine Tiere mehr geschossen werden dürfen.
Dies könnte man durch eine Sinuskurve modellieren.

Damit gilt dann:

$$x'(t) = k_1 \cdot x(t) - k_2 \cdot x(t) \cdot y(t) - k_5(1 + \sin(k_6 \cdot t)$$

$$y'(t) = k_3 \cdot x(t) \cdot y(t) - k_4 \cdot y(t) - k_7(1 + \sin(k_8 \cdot t)$$

Bem:
Da der Sinus minimal -1 werden kann, wurde hier noch 1 dazuaddiert.
Der Jäger kann ja nur eine positive Anzahl Tiere schießen.

Aufgabe:
1) Erstellen Sie ein Programm, das die Punkte P(x(t), y(t)) in Abhängigkeit von der Zeit in ein 2D-Koordinatensystem einträgt.

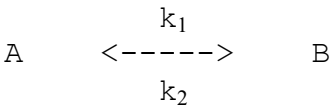
2) Das Programm soll zusätzlich noch die zu x(t) und y(t) zugehörigen Kurven in das gleiche Koordinatensystem eintragen.

2.2 Chemische Simulationen

2.2.1 Modelle

2.2.1.1 Analoges (kontinuierliches) Modell

Ein Stoff A wird mit einer Geschwindigkeit v_1, die proportional zur Konzentration [A] des Stoffes A ist, in den Stoff B umgewandelt.
Der Stoff B wird mit der Geschwindigkeit v_2, die proportional zur Konzentration [B] des Stoffes B ist, in den Stoff A umgewandelt.



$v_1 = k_1$ [A]
$v_2 = k_2$ [B]
k1, k2 sind Geschwindigkeitskonstanten.
[A], [B] in: g/cm³
v_1, v_2 in : g/cm³/s

2.2.1.2 Diskretes Modell

Man legt einen Zeitabschnitt Δt fest, z.B. Δt = 1Sekunde.
Der Wert einer bestimmte Größe zum Zeitpunkt n (n Zeitabschnitte) wie z.B. z (n) berechnet sich aus der Größe von z nach einem vorhergehenden Zeitabschnitt.
Beispiel:
z(0) = 3
z(n) = 2 * z(n-1)

2.2.1.3 Spieltheoretisches Modell (Kugelspiel)

Erwischt man aus der lotterieartigen Ziehung aus einer Urne eine Kugel A, dann wird diese mit einer bestimmten Wahrscheinlichkeit (z.B. wenn mit einem Würfel 6 gewürfelt wird) gegen eine Kugel B ausgetauscht.
Erwischt man eine Kugel B, dann wird diese mit einer bestimmten Wahrscheinlichkeit (z.B. wenn mit einer Münze "Kopf" gewürfelt wird), gegen eine Kugel A ausgetauscht.
Die durchschnittliche Anzahl der Umwandlungen einer Kugel A in eine Kugel B (und umgekehrt) hängt von der Konzentration der Kugel A und von der Wahrscheinlichkeit k_1 ab.

Nach einer bestimmten Anzahl Ziehungen wird das Spiel abgebrochen. Jede Ziehung - ob erfolgreich oder nicht - zählt als eine Zeiteinheit.

Konkrete Werte:
k1 = 1
k2 = 0,5
Anzahl der Kugeln A zum Zeitpunkt 0: A(0) = 150
Anzahl der Kugeln B zum Zeitpunkt 0: B(0) = 0

2.2.2 Lotka-Volterra-Modell

2.2.2.1 Analoges Modell

$$A+X \xrightarrow{k_1} 2X$$
$$X+Y \xrightarrow{k_2} 2Y$$
$$Y \xrightarrow{k_3} E$$

Dabei sind k_1,k_2,k_3 die Geschwindigkeitskonstanten der entsprechenden Prozesse. Die Konzentration von A wird konstant gehalten.

Aus dem Modell folgen die entsprechenden kinetischen Gesetze:

$$\dot{X} = k_1 \cdot A \cdot X - k_2 \cdot X \cdot Y$$
$$\dot{Y} = k_2 \cdot X \cdot Y - k_3 \cdot Y$$

Für den stationären Zustand $S(x_s|y_s)$ gilt:

$$x_s = \frac{k_3}{k_2} \qquad y_s = \frac{k_1}{k_2} \cdot A$$

Bei kleinen Abweichungen (von X und Y) vom stationären Zustand gilt für die Schwingungsdauer T (um den stationären Zustand):

$$T = \frac{2\pi}{\sqrt{k_1 \cdot k_3 \cdot A}}$$

2.2.3 Das Brüsselator-Modell:

2.2.3.1 Analoges Modell

$$A \xrightarrow{k_1} X$$
$$2X+Y \xrightarrow{k_2} 3X$$
$$B+X \xrightarrow{k_3} Y+D$$
$$X \xrightarrow{k_4} E$$

Die Konzentrationen von A,B,D,E werden wieder konstant gehalten.

Aus dem Modell folgen die entsprechenden kinetischen Gesetze:
$dX/dt = k1 \cdot A + k2 \cdot X \cdot X \cdot Y - k3 \cdot B \cdot X - k4 \cdot X$
$dY/dt = k3 \cdot B \cdot X - k2 \cdot X \cdot X \cdot Y$

Spezielle Simulationen

3 Billardspiel

Es soll der Verlauf einer Billardkugel visualisiert werden.
An einer Bande wird die Kugel reflektiert (Einfallswinkel = Ausfallswinkel)

3.1 Bestimmung des Ausfallswinkels

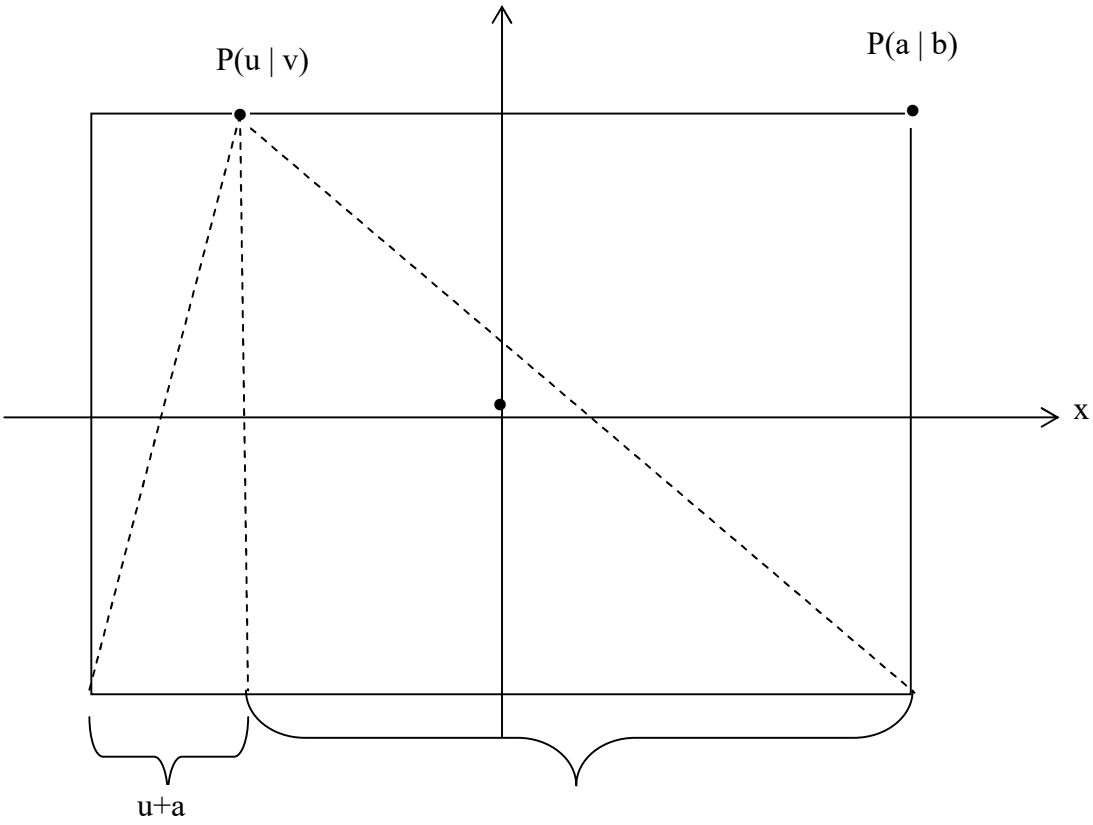
Fall1: Kugel trifft genau in eine der 4 Ecken
Wir setzen dann voraus, dass die reflektierte Kugel in die die gleiche, aber umgekehrte Richtung zurückkommt

Fall2: Kugel trifft nicht genau in eine der 4 Ecken
Unterfall 1: einfallende Kugel fällt nicht senkrecht auf Bande
Steigung der reflektierten Kugel = - Steigung der einfallenden Kugel

Unterfall 2: sonst
reflektierte Kugel kommt senkrecht zurück.

3.2 Mögliche Fälle des Ausfallswinkels

3.2.1 Fall 1: Kugel wird an der oberen Bande reflektiert.



Fall1.1: m $\in$ (-$\frac{2b}{a-u}$, 0] ==> uNeu = a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

Fall1.2: m = -$\frac{2b}{a-u}$ ==> uNeu = a $\wedge$ vNeu = -b $\wedge$ mNeu = m

Fall1.3: m $\in$ (-00, -$\frac{2b}{a-u}$) ==> vNeu = -b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = -m

Fall1.4: m = -00 ==> uNeu = u $\wedge$ vNeu = -b $\wedge$ mNeu = m

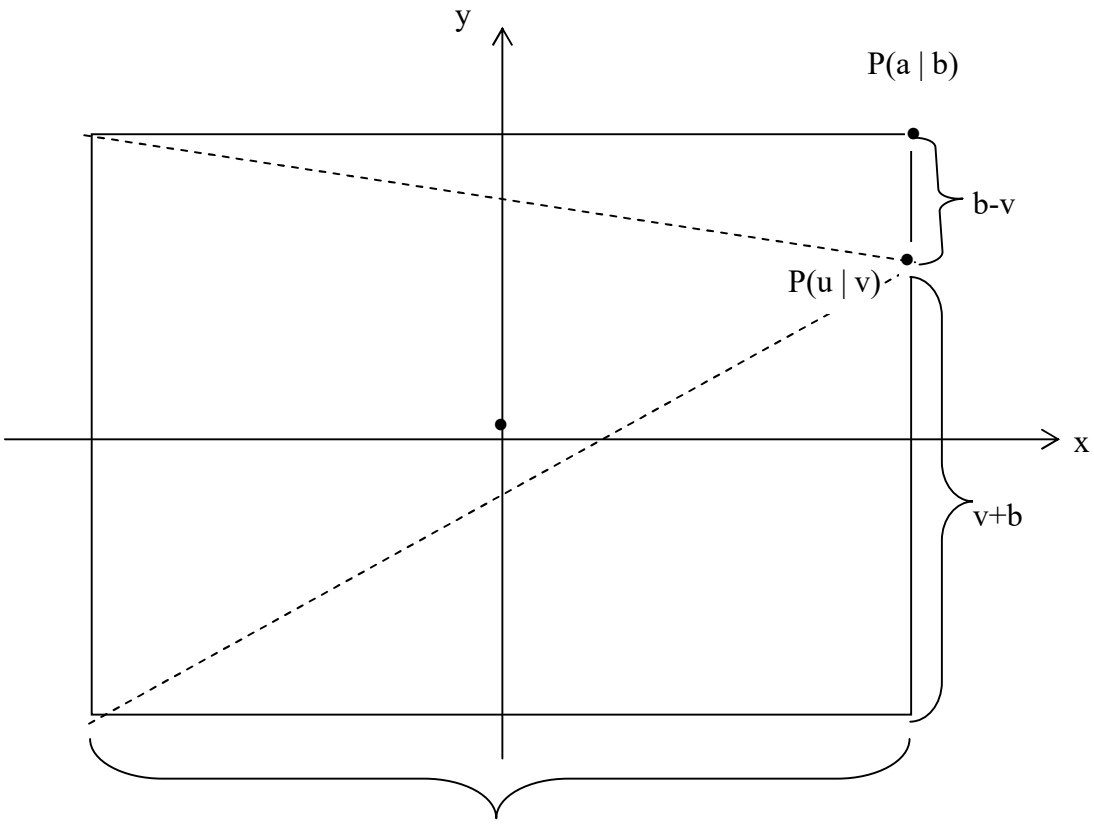
Fall1.5: m $\in$ ($\frac{2b}{u+a}$, 00) ==> vNeu = -b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = -m

Fall1.6: m = $\frac{2b}{u+a}$ ==> uNeu = -a $\wedge$ vNeu = -b $\wedge$ mNeu = m

Fall1.7: m $\in$ (0, $\frac{2b}{u+a}$) ==> uNeu = -a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

3.2.2 Fall 2: Kugel wird an der rechten Bande reflektiert.

Fall2:



Fall2.1: m $\in (\frac{v+b}{2a}, 00]$ ==> vNeu = -b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = -m

Fall2.2: m = 00==> uNeu = u $\wedge$ vNeu = -b $\wedge$ mNeu = m

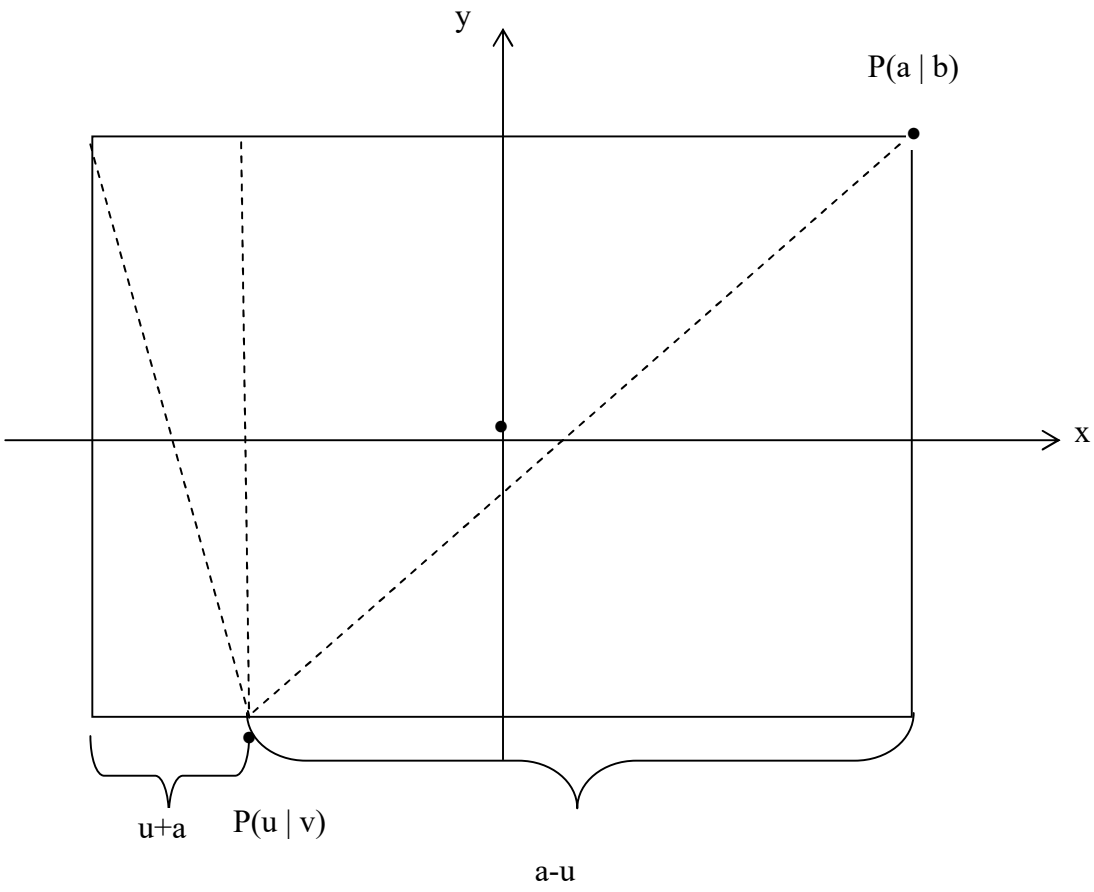
Fall2.3: m = $\frac{v+b}{2a}$ ==> vNeu = -b $\wedge$ uNeu = -a $\wedge$ mNeu = m

Fall2.4: m $\in (-\frac{b-v}{2a}, \frac{v+b}{2a})$ ==> uNeu = -a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

Fall2.5: m = $-\frac{b-v}{2a}$ ==> vNeu = -b $\wedge$ uNeu = -a $\wedge$ mNeu = m

Fall2.6: m $\in (-00, -\frac{b-v}{2a}]$ ==> vNeu = b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = - m

3.2.3 Fall 3: Kugel wird an der unteren Bande reflektiert.



Fall3.1: $m \in (-\frac{2b}{u+a}, 0]$ ==> uNeu = -a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

Fall3.2: $m \in (-00, -\frac{2b}{u+a}]$ ==> vNeu = b $\wedge$ uNeu = $\frac{vNeu - c}{m}$ $\wedge$ mNeu = -m

Fall3.3: m = -00 ==> uNeu = u $\wedge$ vNeu = b $\wedge$ mNeu = m

Fall3.4: $m = -\frac{2b}{u+a}$ ==> uNeu = -a $\wedge$ vNeu = -b $\wedge$ mNeu = m

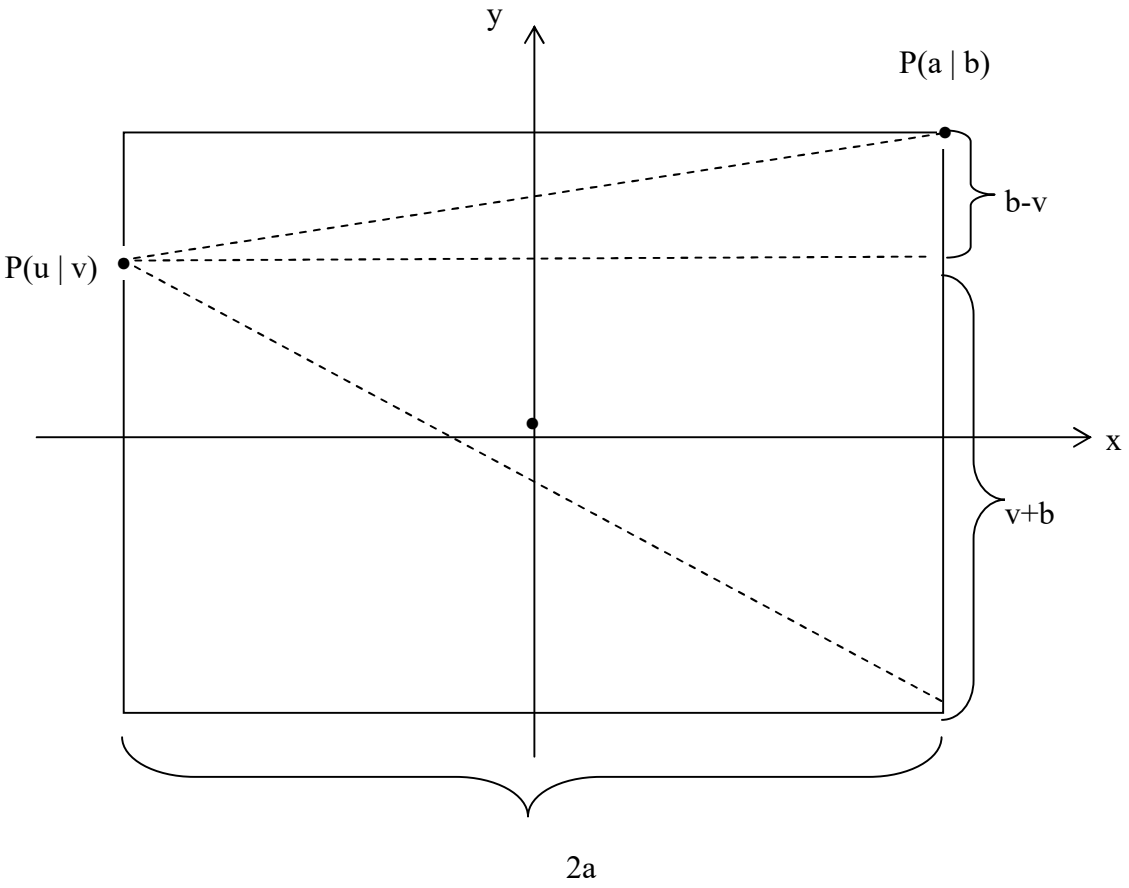
Fall3.5: $m \in (\frac{2b}{a-u}, 00)$ ==> vNeu = b $\wedge$ uNeu = $\frac{vNeu - c}{m}$ $\wedge$ mNeu = -m

x

Fall3.6: $m = \frac{2b}{a-u}$ ==> uNeu = a $\wedge$ vNeu = b $\wedge$ mNeu = m

Fall3.7: $m \in (0, \frac{2b}{a-u})$ ==> uNeu = a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

3.2.4 Fall 4: Kugel wird an der linken Bande reflektiert.



Fall4.1: m $\in$ (-00, $-\frac{v+b}{2a}$) ==> vNeu = -b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = -m

Fall4.2: m = -00 ==> uNeu = u $\wedge$ vNeu = -b $\wedge$ mNeu = m

Fall4.3: m = $-\frac{v+b}{2a}$ ==> vNeu = -b $\wedge$ uNeu = a $\wedge$ mNeu = m

Fall4.4: m $\in$ ($-\frac{v+b}{2a}$, $\frac{b-v}{2a}$) ==> uNeu = a $\wedge$ vNeu = m·xNeu+c $\wedge$ mNeu = -m

Fall4.5: m = $\frac{b-v}{2a}$ ==> vNeu = b $\wedge$ uNeu = a $\wedge$ mNeu = m

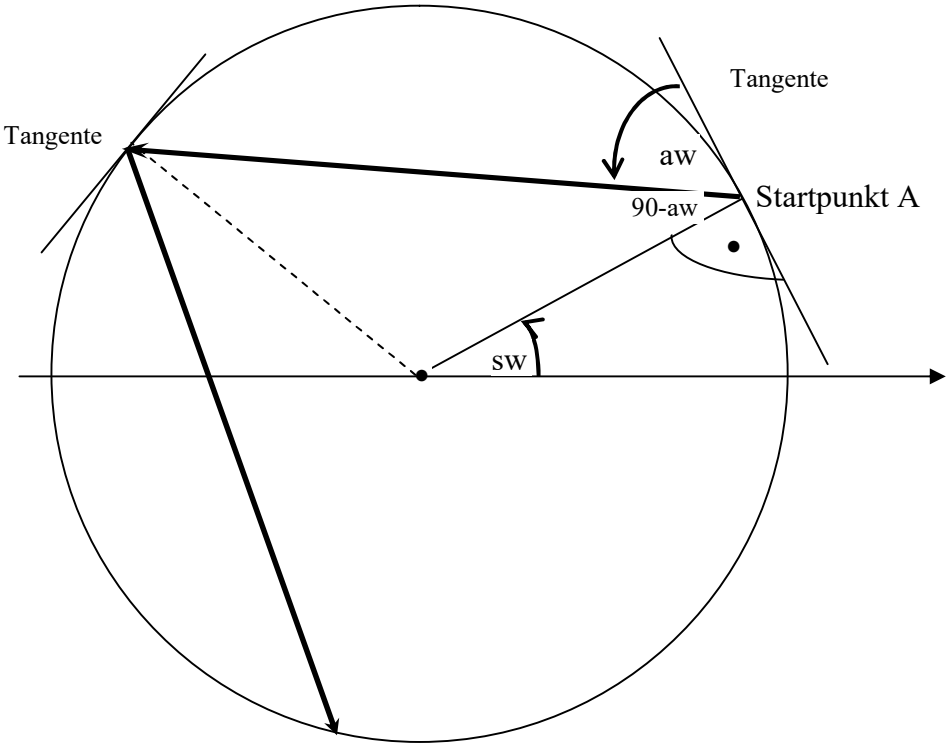
Fall4.6: m $\in$ ($\frac{b-v}{2a}$,00) ==> vNeu = b $\wedge$ uNeu = $\frac{vNeu-c}{m}$ $\wedge$ mNeu = - m

4 Billard im Kreis oder der Weg des Lichts im verspiegelten Kreis

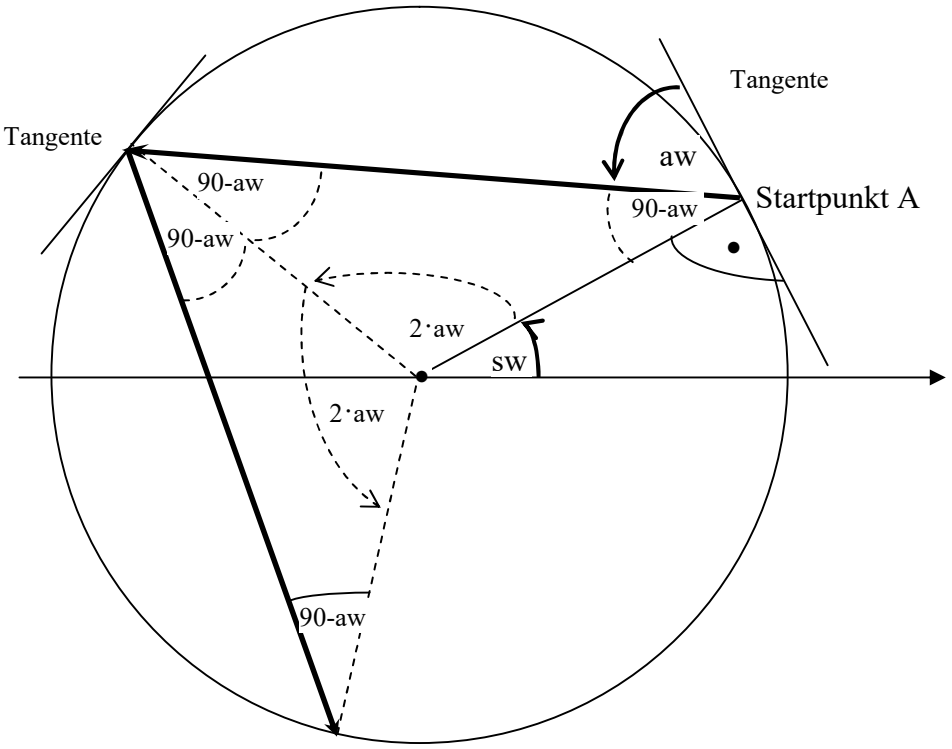
4.1 Aufgabe

Schreiben Sie ein Programm, das den Verlauf des Lichtstrahls (siehe Zeichnung unten) grafisch darstellt.

Ein Kreis mit dem Radius R ist innen verspiegelt. Sein Mittelpunkt M liegt im Ursprung O(0|0) eines rechtwinkligen Koordinatensystems.
Auf der Kreislinie wird durch die Wahl des Startwinkels sw der Startpunkt A bestimmt.
Von diesem Startpunkt A aus wird ein Lichtstrahl unter dem Abschusswinkel aw (bzgl. der Tangente) in den verspiegelten Kreis "abgeschossen" und immer wieder in dem verspiegelten Kreis reflektiert.



4.2 Lösung

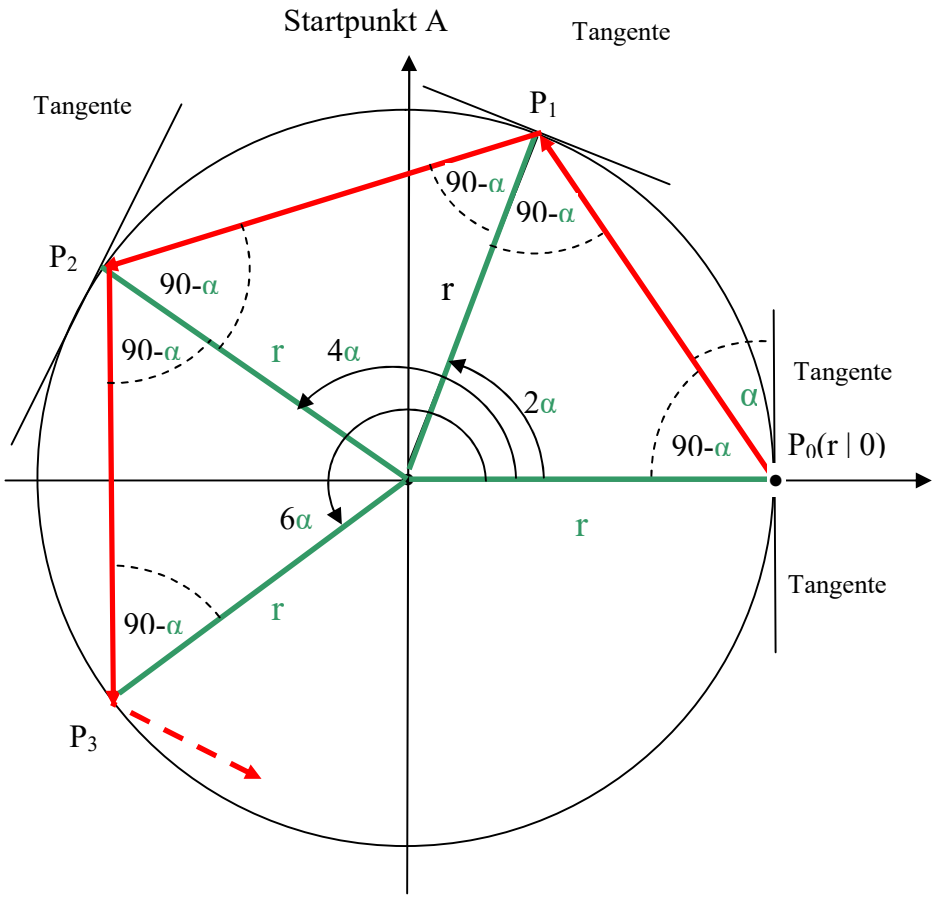


Die Folge der Punkte die der Lichtstrahl im Kreis berührt, lassen sich durch seine Koordinaten bzw. den Winkel w (von der positiven x-Achse aus gemnessen) darstellen.
Für die Folge der Punkte Pn(xn|yn) die der Lichtstrahl auf der Kreislinie berührt gilt:

$w_n = sw + 2n \cdot aw \qquad n=0,1,2,3,...$
$x_n = R \cdot \cos(w_n)$
$y_n = R \cdot \sin(w_n)$

Bemerkung:
Als weitere Aufgabe kann statt des Kreises auch eine andere geometrische Figur (Rechteck, Ellipse, usw.) benutzt werden.

speziell (vereinfachte) Annahme



Startwinkel: α
Unter diesem Winkel tritt der Lichtstrahl am Punkt $P_0(r \mid 0)$ in den verspiegelten Kreis ein.

Nach n Reflexionen ergibt dies den Punkt $P_n(x_n \mid y_n)$ mit:

$w_n = 2n \cdot \alpha$

$x_n = r \cdot \cos(w_n)$

$y_n = r \cdot \sin(w_n)$

5 Die Bäckertransformation

Mit der Bäckertransformation soll das Kneten eines Teigs simuliert werden.
Zuerst drückt der Bäcker den Teig in die Breite. Dadurch wird die y-Koordinate einer Rosine halbiert und die x-Koordinate verdoppelt.
Dann wird der Teig gefaltet .

Konkreter:
Ein Quadrat der Länge a=1 wird durch folgende Konstruktion wieder in ein Quadrat transformiert.

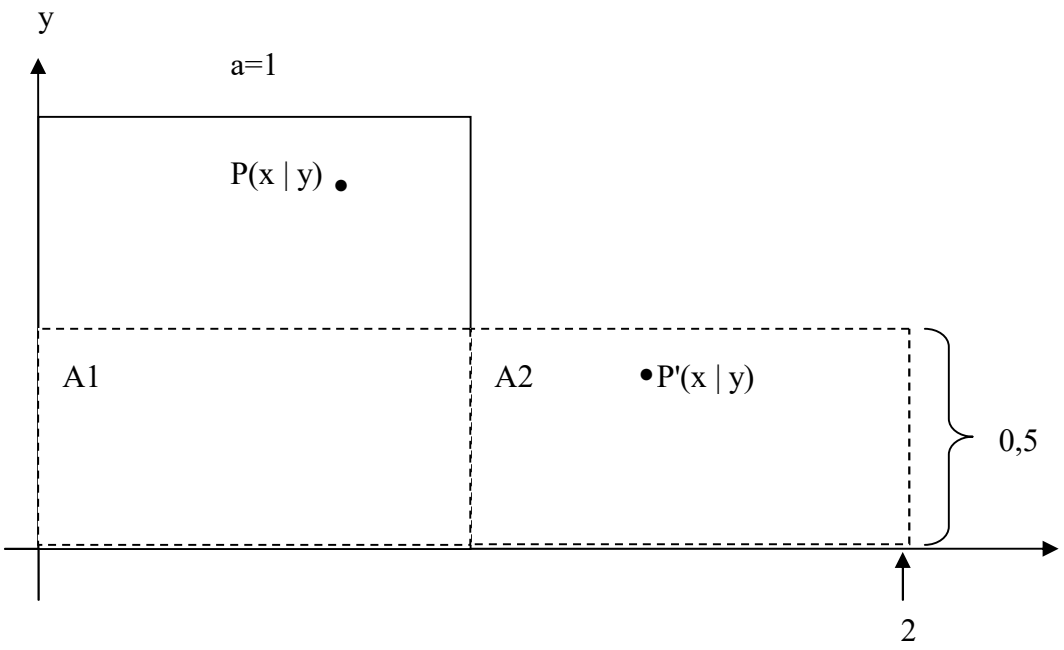
T1) Das Quadrat wird in y-Richtung um den Faktor $\frac{a}{2}$ gestaucht und in x-Richtung um den Faktor 2a gestreckt. Dadurch wird das Quadrat in 2 Teile A1 und A2 aufgeteilt.
T2) Dann wird A2 über A1 verschoben.

Die Punkte innerhalb des Quadrats werden durch die Hintereinanderausführung dieser zweier Transformationen innerhalb des vorgegeben Ausgangsquadrates verschoben.

Alles Folgende wird für **a = 1** realisiert.

5.1 T1 im Koordinatensystem

Die x-Koordinate einer Rosine P(x | y) im Teig wird verdoppelt, die y-Koordinate halbiert.
P(x | y) wird nach P'(x' | y') verschoben.



x' = 2 x

y' = $\frac{1}{2}$ y

x

5.1.1 Beispiel $P(\frac{1}{3} \mid \frac{1}{4})$

$x' = 2 \cdot x = 2 \cdot \frac{1}{3} = \frac{2}{3}$

$y' = \frac{1}{2} \cdot y = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$

5.1.2 Beispiel $P(\frac{2}{3} \mid \frac{3}{4})$

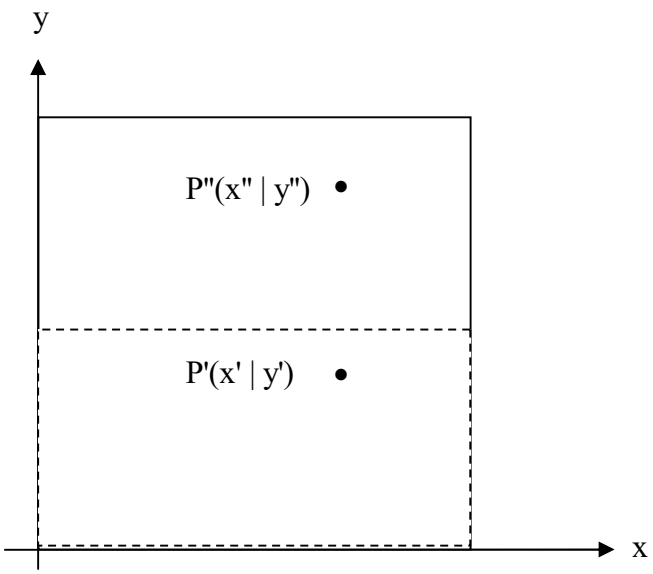
$x' = 2\ x = 2 \cdot \frac{2}{3} = \frac{4}{3}$

$y' = \frac{1}{2}\ y = \frac{1}{2} \cdot \frac{3}{4} = \frac{3}{8}$

5.2 T2 im Koordinatensystem

Wenn eine Rosine (durch die vorige Transformation T1) außerhalb des Quadrates verschoben wurde, wird sie wieder nach innen verschoben und die y-Koordinate um ½ vergrößert (d.h. nach oben geschoben).
Wenn sich die Rosine aber - nach der Verschiebung - innerhalb des Quadrates befindet, wird nichts mehr gemacht, weil nur der rechte Teiglappen nach oben geschoben wird.

5.2.1 Fall $x' \leq 1$



$x'' = x'$
$y'' = y'$

5.2.1.1 Beispiel P' ($\frac{1}{4}$ | $\frac{1}{3}$)

$x'' = x' = \frac{1}{4}$

$y'' = y' = \frac{1}{3}$

5.2.2 Fall x' > 1

$x'' = x' - 1$

$y'' = y' + \frac{1}{2}$

5.2.2.1 Beispiel P' ($\frac{3}{2}$ | $\frac{1}{4}$)

$x'' = x' - 1 = \frac{3}{2} - 1 = \frac{1}{2}$

$y'' = y' + \frac{1}{2} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$

5.3 Die gesamte Transformation

5.3.1 Fall $x \leq \frac{1}{2}$

$y' = \frac{1}{2}\ y$ $\qquad$ $y'' = y'$

$x' = 2\ x$ $\qquad$ $x'' = x'$

also:

$y'' = y' = \frac{1}{2} y$

$x'' = x' = 2x$

also:

$y'' = \frac{1}{2}\ y$ und $x'' = 2x$

5.3.2 Fall $x > \frac{1}{2}$

$y' = \frac{1}{2}\ y$ $\qquad$ $y'' = y' + \frac{1}{2}$

$x' = 2\ x$ $\qquad$ $x'' = x' - 1$

also:

$y'' = y' + \frac{1}{2} = \frac{1}{2} y + \frac{1}{2}$

$x'' = x'-1 = 2\ x - 1$

also:

$y'' = \frac{1}{2} y + \frac{1}{2}$ und $x'' = 2x - 1$

Insgesamt wandert der Punkt P(x,y) nach P"(x" | y") bzw. durch die Wanderung ist folgende Abbildung f definiert:

P(x | y) ---> P"(2x | 0,5 y) falls $x \leq 0{,}5$
P(x | y) ---> P"(2x-1 | 0,5 y + 0,5y) falls $x > 0{,}5$

5.3.3 Beispiel $P(\frac{1}{3} \mid \frac{1}{4})$

Da $x \leq \frac{1}{2}$ gilt:

$x'' = 2 \cdot \frac{1}{3} = \frac{2}{3}$ und $y'' = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$

und damit

$P''(\frac{2}{3} \mid \frac{1}{8})$

5.3.4 Beispiel $P(\frac{2}{3} \mid \frac{3}{4})$

Da $x > \frac{1}{2}$ gilt:

$x'' = 2x - 1 = 2 \cdot \frac{2}{3} - 1 = \frac{1}{3}$

$y'' = \frac{1}{2}y + \frac{1}{2} = \frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} = \frac{7}{8}$

und damit

$P''(\frac{1}{3} \mid \frac{7}{8})$

6 Die Bernoulli-Verschiebung

Man wähle eine Zahl z mit folgender Eigenschaft: $0 < z < 1$
Multipliziere die Zahl mit 2.
Wenn das Ergebnis > 1 ist, wird 1 subtrahiert, sonst wird sie wieder mit 2 multipliziert.
Das geschieht immer wieder.
Formaler:

$z_0 = z$

$z_{n+1} = 2 \cdot z_n$ falls $2 \cdot z_n \leq 1$

$z_{n+1} = 2 \cdot z_n - 1$ falls $2 \cdot z_n > 1$

Beispiel:

$z_0 = 0{,}13$
$z_1 = 2 \cdot 0{,}13 = 0{,}26$
$z_2 = 2 \cdot 0{,}26 = 0{,}52$
$z_3 = 2 \cdot 0{,}52 - 1 = 0{,}04$
$z_4 = 2 \cdot 0{,}04 = 0{,}08$
$z_5 = 2 \cdot 0{,}08 = 0{,}16$
$z_6 = 2 \cdot 0{,}16 = 0{,}32$
...

7 Simulationen des Chaos (Chaos-Theorie)

In einem Kinderheim sind die Masern ausgebrochen. Jeden Tag wird sich die Anzahl der kranken Kinder erhöhen, weil nicht zu vermeiden ist, dass kranke und gesunde Kinder miteinander Kontakt haben.
Das folgende mathematische Modell soll versuchen, diese Krankheit zu simulieren, um das Verhalten und die Gesetzmäßigkeiten eines solchen Systems zu verstehen:

Die Anzahl der kranken Kinder am Tag i+1 (mit a_{i+1} bezeichnet) hängt einerseits von dem Prozentsatz der bereits kranken Kinder am Vortag i (mit a_i bezeichnet) ab. Je höher der Prozentsatz kranker Kinder am Tag i ist, desto höher ist einen Tag später der Prozentsatz kranker Kinder.
Der Prozentsatz kranker Kinder am Tag i+1 hängt auch von dem Prozentsatz gesunden Kinder am Vortag i ab, denn es ist kein Zuwachs mehr möglich, wenn alle Kinder krank im Bett liegen.
Wenn 30% (=0,3) krank sind, sind noch 100%- 30% = 1 -0,3 gesund.
Allgemein:
Wenn p% (= p) Kinder krank sind, sind 100% - p (= 1 - p) gesund-
Da sich die Kinder sicher nicht alle treffen und auch nicht jeder Kontakt zu einer Ansteckung führt, taucht in der Formel für den Prozentsatz der kranken Kinder am Tag i+1 auch noch ein Ansteckungsfaktor k auf.
Berücksichtigt man alle diese Überlegungen für eine Gesamtformel, so gilt:

$a_{i+1} = a_i \cdot (1-a_i) \cdot k$ wobei $0 \le a_i \le 1$ und $k \ge 0$

Für jeweils einen festen Ansteckungsfaktor k kann man ausgehend von einem Startwert a_0 den Verlauf der Krankheit berechnen.

7.1.1 Beispiel

$a_0 = 0,25$
$k = 2$

i	a_i
0	0,25
1	0,375
2	0,46875
3	0,49999237
...	...

7.1.2 Aufgaben

1) Berechnen Sie für den festen Startwert $a_0 = 0,3$ für die folgenden k-Werte
0,5 1 1,5 2 2,5 3
jeweils in einer Tabelle den Krankheitsverlauf mit einem Taschenrechner.

2) Machen Sie das gleiche mit einem Tabellenkalkulationsprogramm:
Wählen Sie einen festen Startwert a_0 im Bereich [0, 1], d.h. zwischen 0 und 1.
Auf der x-Achse werden die Werte von k im Bereich [0, 4] aufgetragen.
Auf der y-Achse werden die Werte von a_n im "eingeschwungenen" Zustand, z.B. im Bereich zwischen a_{100} und a_{200} aufgetragen.

3) Schreiben Sie ein Programm.

7.1.3 Ergebnis

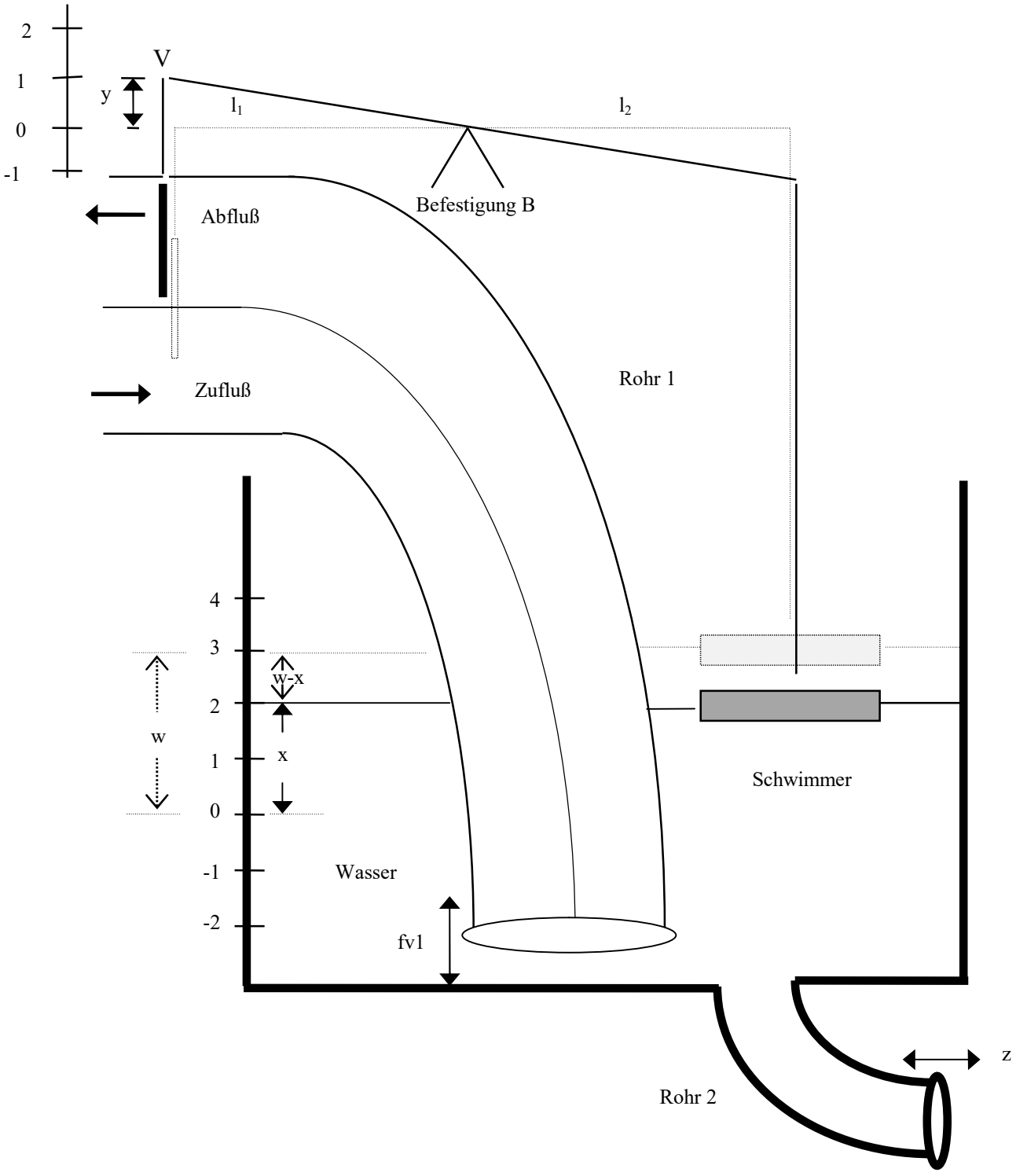
Wenn k zwischen 0 und 3 liegt, streben die a_n für große n gegen **einen** festen Wert.
Wenn k zwischen 3 und $1+\sqrt{6} \approx 3{,}5$ liegt, "pendeln" die a_n für große n zwischen 2 Werten
Man sagt a_n hat die Periode 2.
Wenn k größer wird, wird auch die Periode größer. Ab dem $k \approx 3{,}5699$ springen die Werte a_n wild durcheinander. Vorhersagen sind nicht mehr erkennbar.
Es findet ein Übergang von der Ordnung ins Chaos statt.

Wegen der starken Abhängigkeit der Anzahl a_n von k - kleinste Änderungen von k können völlig andere Ausgangswerte und Szenarien ergeben- sagt man, dass ein **deterministisches Chaos** vorliegt.
Das mathematische Modell, das das Wetter beschreibt, ist auch ein deterministisches Chaos. Weil alle von Wetterstationen gemessene Systemparameter nur einen bestimmten Messfehler haben, ist es möglich, dass der Wert des Parameters gerade einen kritischen Wert hat und deswegen an dieser Stelle nicht mehr vorhersagbar ist !

8 Kybernetik (Regelung)

8.1 Zeichnung

(siehe auch Skript Regelungstechnik.doc)



8.2 Beschreibung

Abkürzungen:

y: Stellgröße (m) = Abstand des Punktes V von seiner „Normallage“
yv: Geschwindigkeit (m/s) des Punktes V
ya: Beschleunigung (m/s²) des Punktes V
x: Regelgröße (m) = Wasserhöhe des Wasserpegels (über dem Boden des Wasserbehälters)
(= Höhe des Schwimmers über dem Boden des Wasserbehälters)
v: Geschwindigkeit (m/s) des Wasserpegels = Geschwindigkeit des Schwimmers (= $\dot{x}(t)$)
a: Beschleunigung (m/s²) des Wasserpegels = Beschleunigung des Schwimmers (= $\ddot{x}(t)$)
z: Störgröße (m/s) = Geschwindigkeit des Flußes (Zufluß bzw. Abfluß) in Rohr 2 (wird auch zv genannt)
za: Beschleunigung (m/s²) des Flußes in Rohr 2
w: Führungsgröße (m)

fv1: Geschwindigkeit (m/s) des Flußes (Flußgeschwindigkeit) in Rohr 1
fv2: Geschwindigkeit (m/s) des Flußes (Flußgeschwindigkeit) in Rohr 2 = z = zv
fa1: Beschleunigung (m/s²) des Flußes (Flußbeschleunigung) in Rohr 1
fa2: Beschleunigung (m/s²) des Flußes (Flußbeschleunigung) in Rohr 2 = za

Bei der Regelung der Wasserhöhe (Wasserstand, Höhe des Wasserpegels) in einem Wasserbehälter wird das Ventil im Rohr 1 über einen Hebel von einem Schwimmer verstellt. Rohr 1 hat einen quadratischen Querschnitt und besteht intern aus einem Zufluß und einem Abfluß.
Wenn sich das Ventil mehr im Zufluß als im Abfluß befindet, versperrt es mehr vom Zufluß als vom Abfluß, so daß insgesamt Wasser aus dem Wasserbehälter über Rohr 1 abfließt.
Wenn sich das Ventil mehr im Abfluß als im Zufluß befindet, versperrt es mehr vom Abfluß als vom Zufluß, so daß insgesamt Wasser in den Wasserbehälter über Rohr 1 zufließt.

Der Fluß in Rohr 1 (bzw. Rohr 2) ist positiv wenn aus Rohr 1 (bzw. Rohr 2) Wasser in den Wasserbehälter zufließt oder negativ wenn aus Rohr 1 (bzw. Rohr 2) Wasser aus dem Wasserbehälter abfließt.

8.2.1 Messung des Flußes in Rohr 1 bzw. Rohr 2

Der Fluß in Rohr 1 (bzw. Rohr 2) wird oft in Volumeneinheiten pro Zeiteinheit gemessen, wie z.B. 10 l /s
Wir messen ihn hier anders:
Der Fluß in Rohr 1 veranlaßt eine Veränderung der Wasserhöhe im Wasserbehälter. Wir messen deshalb den Fluß aus Rohr 1 als die Wasserhöhenänderung pro Zeiteinheit (falls Rohr 2 verschlossen wäre), wie z.B. 0,002 m/s .
Genauso wird der Fluß in Rohr 2 als die Wasserhöhenänderung pro Zeiteinheit gemessen (falls Rohr 1 verschlossen wäre). Der Fluß stellt also eine Geschwindigkeit dar.
Diese Art der Messung kennt man aus der Wetterkunde:
Dort wird die Niederschlagsmenge in cm (Wasserhöhe) pro Jahr angegeben.

Steigt die Wasserhöhe, dann bewegt sich das Ventil nach unten. Dann wird der Fluß kleiner. Das heißt, der Zufluß wird kleiner und der Abfluß größer.

Sinkt die Wasserhöhe, dann bewegt sich das Ventil nach oben. Dann wird der Fluß größer. Das heißt, der Zufluß wird größer und der Abfluß kleiner.

8.2.2 Regelungstechnische Begriffe in diesem Regelkreis

Die Regelgröße x (in m) ist die Wasserhöhe des Wasserpegels über dem Boden des Wasserbehälters.
Die Stellgröße y (in m) ist der Abstand des Punktes V von seiner „Normallage“ (in der Normallage ist der Fluß gleich 0) . In der Zeichnung ist die Normallage gestrichelt gekennzeichnet.
Die Störgröße z (in m/s) ist die Geschwindigkeit des Flußes (Zufluß oder Abfluß) in Rohr 2.
Die Führungsgröße w (in m) ist die Wasserhöhe, bei welcher der Fluß in Rohr 1 gleich 0 wird, d.h. bei welchem Wert der Regelgröße x der Wert der Stellgröße y gleich 0 wird. Das heißt, man kann die Führungsgröße w aus der Zeichnung ablesen, indem man gedanklich die Stellgröße y auf 0 bringt (in seine „Normallage“) und die zugehörige Wasserhöhe x abliest. Dies ist dann die Führungsgröße w.
Die Führungsgröße wird durch die Höhe der Befestigung B bestimmt.

8.2.3 Die Gleichung des Reglers

Der Regler ist ein Proportionalregler (ohne Totzeit), das heißt die Abweichung $x_d = w - x$ von der Führungsgröße w ist proportional der Stellgröße y.
Die Proportionalitätskonstante k_{PR} ist das Verhältnis von den Längen l_1 und l_2 des Hebels. $k_{PR} = l_1 / l_2$, also:

Regler:

$$(R\,1) \quad y(t) = k_{PR}(w - x(t))$$

Beispiel (mit Dimensionen):
Annahme: $w=1m,\ k_{PR}=0,01,\ x=3m$
dann gilt für die Stellgröße y:
$y = 0,01 * (1m - 3m) = 0,01 * -2m = -0,02m$

8.2.4 Die Gleichung der Regelstrecke

Die Geschwindigkeit des Wasserpegels (= Geschwindigkeit des Schwimmers) ergibt sich aus der Summe der Geschwindigkeiten der Flüße (= fv1 + fv2) aus dem Schwimmbecken, ist also die Summe vom Fluß in Rohr 1 und dem Fluß in Rohr 2.

Da man empirisch (durch praktische Messungen) feststellen kann (oder durch einfache mathematische Überlegungen, auf die hier nicht eingegangen werden sollen), daß der Fluß in Rohr 1 gleich einer Proportionalitätskonstanten k_{IS} multipliziert mit der Stellgröße y ist und der Fluß in Rohr 2 gleich der Störgröße z ist, gilt für die Geschwindigkeit des Wasserpegels (außerdem ist die Anfangswasserhöhe gleich xanf):

Regelstrecke:

$$(S\,1) \quad v(t) = k_{IS}\ y(t) + z(t)$$

$$x(0) = xanf$$

8.3 Der Wert von v(t) nach der Zeit t

Aus den Gleichungen für den Regler und der Regelstrecke folgt die Gleichung für den *Regelkreis:*

$$\text{(K 1)} \quad v(t) = k_{IS} * k_{PR} * (w - x(t))$$

$$x(0) = xanf$$

8.4 Der Wert von v_n nach dem n-ten Zeitabschnitt Δt

Wir wollen die Wasserhöhe nach 1 Zeitabschnitt, nach 2 Zeitabschnitten, nach 3 Zeitabschnitten, allgemein die Wasserhöhe x_n nach n Zeitabschnitten Δt (Δt ist fest vorgegeben und man kann z.B. für einen Zeitabschnitt Δt eine Sekunde wählen) berechnen. Man definiert v_n als Geschwindigkeit des Wasserpegels (mit der Wasserhöhe x) nach n Zeitabschnitten Δt

$$x_n := x(n*\Delta t)$$

$$v_n := v(n*\Delta t)$$

Für t muß nun $n*\Delta t$ in die Gleichung (K 1) des Regelkreises eingesetzt werden:

$$v(n*\Delta t) = k_{IS} * k_{PR} * (w - x(n*\Delta t))$$

das ergibt:

$$\text{(G 1)} \quad v_n = k_{IS} * k_{PR} * (w - x_n)$$

8.5 Der angenäherte Wert der Regelgröße x_n nach dem n-ten Zeitabschnitt Δt

Die Geschwindigkeit v_n läßt sich annähern durch:

$$v_n \approx (x_{n+1} - x_n)/\Delta t$$

eingesetzt in Gleichung (G 1) ergibt:

$$(x_{n+1} - x_n)/\Delta t \approx k_{IS} * k_{PR} * (w - x_n)$$

oder:

Regelkreis:

(K 1)	$x_{n+1} \approx x_n + k_{IS} * k_{PR} * \Delta t * (w - x_n)$ für $n>=1$
(K 2)	$x_0 = xanf$

Das heißt: aus der Wasserhöhe x_n (n>=1), läßt sich die Wasserhöhe x_{n+1} berechnen.

Es gilt:
x_0 läßt sich mit (K 2) berechnen,
x_{n+1} (n >=1) mit (K 1).

kurz:

x_0		mit (K 2)
x_0	$\rightarrow x_1$	mit (K 1)
x_1	$\rightarrow x_2$	mit (K 1)
x_2	$\rightarrow x_3$	mit (K 1)
...		

Das heißt man kann die Wasserhöhe nach einem beliebigen Zeitabschnitt t_n berechnen !!!

Aufgaben:
1) Erstellen Sie ein Programm, das in einem Koordinatensystem den zeitlichen Verlauf von x_n darstellt (z.B. $\Delta t=0,01$ $w = 2$ $k_{PR}=2$ $k_{IS}=0,5$ $x_0=xanf=0$)

2) Verändern Sie die Werte von w, k_{PR} , k_{IS} usw. und betrachten Sie das dazugehörige Diagramm.
Wählen Sie k_{PP} und k_{PS} so, daß:
2.1) $k_{PR} * k_{IS} > 0$
2.2) $k_{PR} * k_{IS} = 0$
2.3) $k_{PR} * k_{IS} < 0$

und betrachten Sie den Verlauf von x_n.

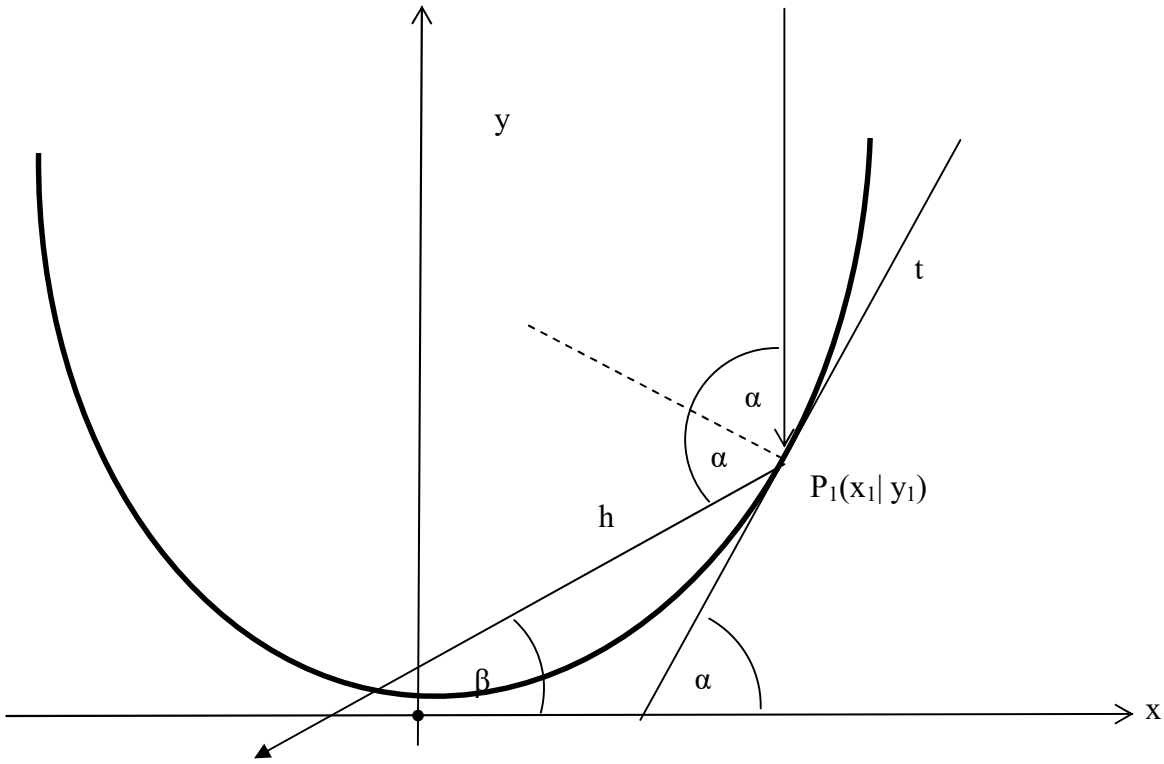
Bestätigen Sie, daß für
2.4) $k_{PR} * k_{IS} > 0$ der Wert von x_n sich dem Wert g = w nähert (für „große“ n).
2.5) $k_{PR} * k_{IS} = 0$ der Wert von x_n = xanf ist.
2.6) $k_{PR} * k_{IS} < 0$ der Wert von x_n gegen $\pm\infty$ geht.

3) Wählen Sie die Werte von w und x_0 so, daß sich die Regelgröße x_n möglichst schnell der Führungsgröße w nähert und ändern Sie dann (ab dem n, ab dem sich x_n genügend w genähert hat) sprungartig den Wert von w um +1.
Wie verändert sich die Regelgröße x (Sprungantwort) ?

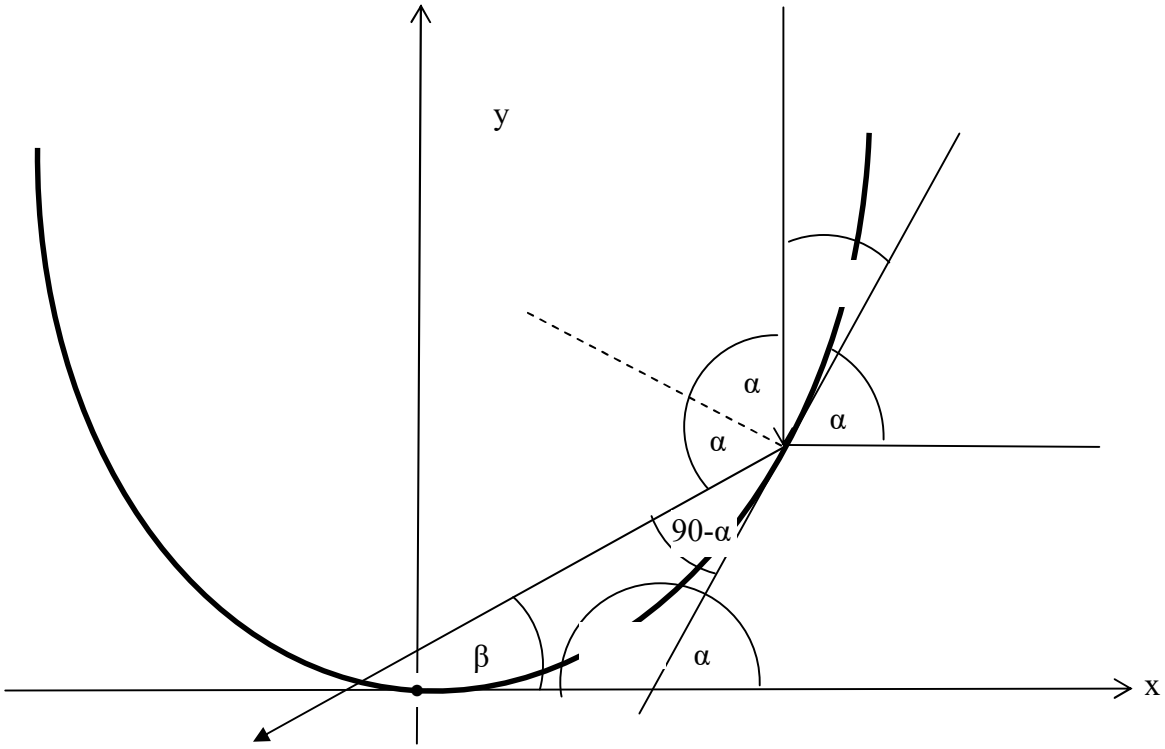
9 Problem eines Autozulieferers

9.1 Motivation

Ein Autozulieferer muss sich überlegen, wo er die Glühlampe in einem verspiegelten Autoscheinwerfer anbringen muss, so dass die reflektierenden Strahlen parallel den Schweinwerfer verlassen.
Man kann also ein Programm entwickeln, das senkrecht einfallende Strahlen in einen Scheinwerfer simuliert. Dann kann man visuell nachprüfend, ob sich die reflektierenden Strahlen in einem Punkt schneiden.
In den Scheinwerfer, dessen zugehörige Kurve durch die Funktion f beschrieben wird, fallen parallel zur y-Achse senkrecht einfallende Lichtstrahlen.
Ein senkrecht einfallender Lichtstrahl geht durch den Punkt $P_1(x_1|y_1)$, wird reflektiert und bildet die Gerade h.



9.2 Mathematik



1)
Es gilt:
$\beta + 180 - \alpha + 90 - \alpha = 180$
also:
$\beta = 2\,\alpha - 90°$

2)
Die Steigung m_1 der Tangenten t im Punkt $P_1(x_1|y_1)$ beträgt: $f'(x_1)$
Die zugehörige Funktionsgleichung der Tangenten ist:
$m_1 = \tan(\alpha)$
Für die Funktionsgleichung der Tangenten t gilt also:

$$\frac{y - y_1}{x - x_1} = f'(x_1) \implies y = f'(x_1)(x - x_1) + y_1$$

Für den Winkel α gilt also:
$\alpha = \tan^{-1}(f'(x_1))$

3)
Die Steigung m_2 von h beträgt:
$m_2 = \tan(\beta) = \tan(2\,\alpha - 90°) = \tan(2 \cdot \tan^{-1}(f'(x_1)) - 90°)$
Für die Funktionsgleichung der Geraden h gilt:

$$\frac{y - y_1}{x - x_1} = \tan(2 \cdot \tan^{-1}(f'(x_1)) - 90°) \implies$$

$y = (x - x_1) \cdot \tan(2 \cdot \tan^{-1}(f'(x_1)) - 90°) + y_1$

10 Verfolgungsprobleme im 2D

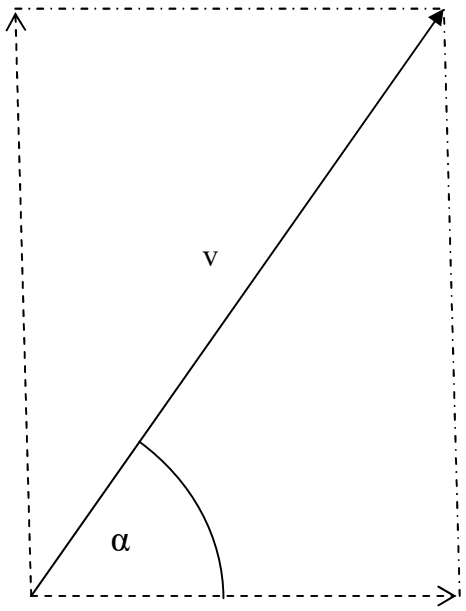
10.1 Motivation

Ein Hund P_1 verfolgt einen flüchtenden Hasen P_2.
Wie soll sich der Hase optimal verhalten?

10.2 Mathematische Grundlagen

Ein Punkt bewegt sich mit der Geschwindigkeit v relativ zum Winkel α.

Wie groß sind die Vektorkoordinaten des Vektors $\overrightarrow{v}$?



$\cos(\alpha) \cdot v$

also:

$$\vec{v} = \begin{pmatrix} \cos(\alpha) \cdot v \\ \sin(\alpha) \cdot v \end{pmatrix}$$

10.3 Modellierungen

Gegeben ist ein konstanter Zeitabschnitt dt (z.B. 1 Sekunde). Nach jedem Zeitabschnitt dt wird jeweils P_1 und P_2 betrachtet.
Jeder Punkt kann seine Richtung nach Belieben ändern.
Dies ist ein so genanntes **Differentialspiel**.
Frage:
Gibt es eine Strategie für P_2, dass er von P_1 nicht gefangen wird bzw.
die Fangzeit (Zeit bis P_2 er eingefangen wird) möglichst groß wird ?

$x_1(n)$: x-Koordinate von P_1 zum Zeitpunkt n
$y_1(n)$: y-Koordinate von P_1 zum Zeitpunkt n
$x_2(n)$: x-Koordinate von P_2 zum Zeitpunkt n
$y_2(n)$: y-Koordinate von P_2 zum Zeitpunkt n
$w_1(n)$: Richtung (Winkel bzgl. x-Achse) von P_1 zum Zeitpunkt n
$w_2(n)$: Richtung (Winkel bzgl. x-Achse) von P_2 zum Zeitpunkt n

Abkürzung:

w_P_1(n-1)_ P_2(n-1) = Winkel des Richtungsvektors $\overrightarrow{P_1(n-1)P_2(n-1)}$

= w_P1(1)_P2(1)) = Winkel des Richtungsvektors $\overrightarrow{P_1(1)P_2(1)}$

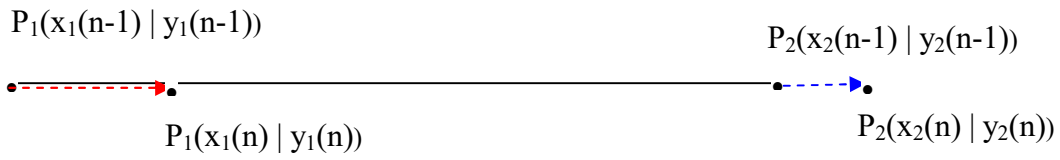
10.3.1 Modellierung 1

10.3.1.1 Beschreibung

Ein Punkt $P_1(x_1 \mid y_1)$ mit der konstanten Geschwindigkeit v_1 verfolgt einen anderen Punkt $P_2(x_2 \mid y_2)$, der die konstante Geschwindigkeit v_2 besitzt.
Die Strategie von P_1 ist es, immer seine Bewegungsrichtung direkt auf P_2 zu halten.
P_2 flüchtet immer in die gleiche Bewegungsrichtung (in direkter Linie weg von P_1), d.h. P_1 und P_2 sind auf einer Geraden.

10.3.1.2 mathematische Modellierung

n = 1:
$x_1(1)$: anfängliche x-Koordinate von P_1
$y_1(1)$: anfängliche y-Koordinate von P_1
$w_1(1)$: Anfangsrichtung von P_1, z.B. $w_1(1) = w\_P1(1)\_P2(1))$
$x_2(1)$: anfängliche x-Koordinate von P_2
$y_2(1)$: anfängliche y-Koordinate von P_2
$w_2(1)$: Anfangsrichtung von P_2, z.B. $w_2(1) = w\_P1(1)\_P2(1))$



n > 1:

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$w_1(n) = w\_P_1(n)\_ P_2(n)$

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta t$$

$w_2(n) = w\_P_1(n)\_ P_2(n)$

10.3.1.3 alternative mathematische Modellierung

$$\overrightarrow{P_1P_2(n)} = \begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}$$

$$r(n) := |\overrightarrow{P_1P_2(n)}| = \sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}$$

Für den Einheitsvektor $\overrightarrow{e_0}(n)$ von $\overrightarrow{P_1P_2(n)}$ gilt:

$$\overrightarrow{e_0}(n) = \frac{\overrightarrow{P_1P_2(n)}}{r(n)} = \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{r(n)} = \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{\sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}}$$

Es gilt:

$$\overrightarrow{OP_1}(n+1) = \overrightarrow{OP_1}(n) + v_1 \cdot \Delta t \cdot \overrightarrow{e_0}(n)$$

wobei e_0 ein Einheitsvektor ist

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}(n) = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}(n-1) + v_1 \cdot \Delta t \cdot \frac{\begin{pmatrix} x_2(n-1) - x_1(n-1) \\ y_2(n-1) - y_1(n-1) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(n) = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(n-1) + v_2 \cdot \Delta t \cdot \frac{\begin{pmatrix} x_2(n-1) - x_1(n-1) \\ y_2(n-1) - y_1(n-1) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

10.3.2 Modellierung 2

10.3.2.1 Beschreibung

Ein Punkt $P_1(x_1 \mid y_1)$ mit der konstanten Geschwindigkeit v_1 verfolgt einen anderen Punkt $P_2(x_2 \mid y_2)$, der die konstante Geschwindigkeit v_2 besitzt.
Die Strategie von P_1 ist es, immer seine Bewegungsrichtung direkt auf P_2 zu halten.
P_2 flüchtet immer in senkrechte Richtung zur gedachten Linie zwischen P_1 und P_2

10.3.2.2 mathematische Modellierung

n = 1:
$x_1(1)$: anfängliche x-Koordinate von P_1
$y_1(1)$: anfängliche y-Koordinate von P_1
$w_1(1)$: Anfangsrichtung von P_1, z.B. $w_1(1)$ = w_P1(1)_P2(1))
$x_2(1)$: anfängliche x-Koordinate von P_2
$y_2(1)$: anfängliche y-Koordinate von P_2
$w_2(1)$: Anfangsrichtung von P_2, z.B. $w_2(1)$ = w_P1(1)_P2(1)) - 90°

n > 1:
$x_1(n) = x_1(n-1) + \cos(w_1(n-1)) \cdot v_1 \cdot \Delta t$
$y_1(n) = y_1(n-1) + \sin(w_1(n-1)) \cdot v_1 \cdot \Delta t$
$x_2(n) = x_2(n-1) + \cos(w_2(n-1)) \cdot v_2 \cdot \Delta t$
$y_2(n) = y_2(n-1) + \sin(w_2(n-1)) \cdot v_2 \cdot \Delta t$

andere Schreibweise wie oben:

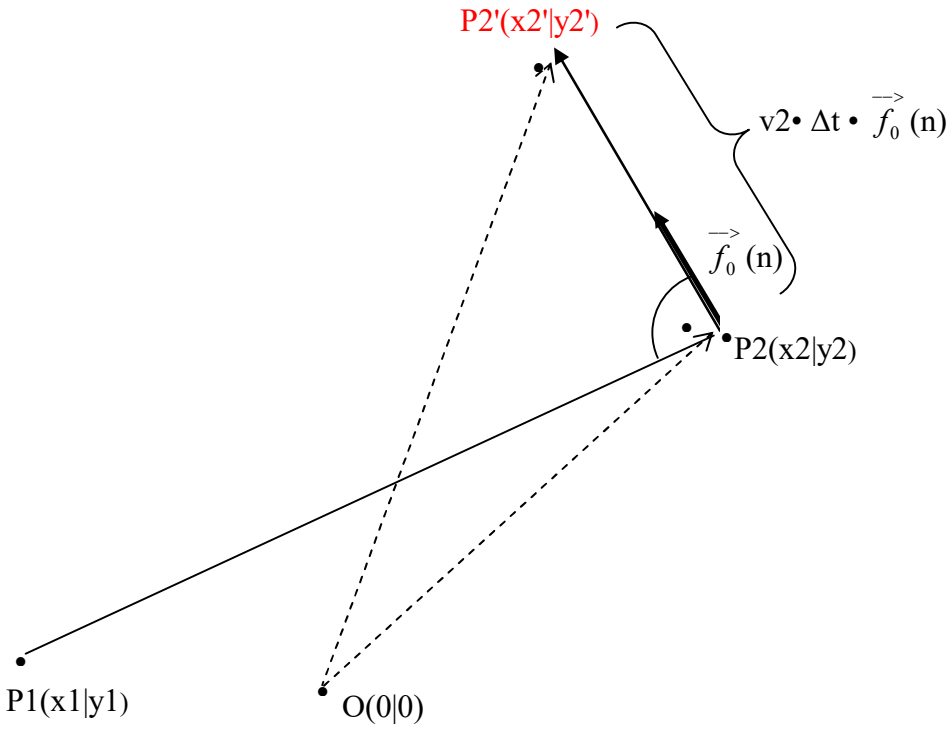
$$\overrightarrow{OP_1}(n) = \overrightarrow{OP_1}(n-1) + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$w_1(n)$ = w_P1(n)_ P2(n)

$$\overrightarrow{OP_2}(n) = \overrightarrow{OP_2}(n-1) + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta t$$

$w_2(n)$ = w_P1(n)_ P2(n) + deltaW2

10.3.2.3 alternative mathematische Modellierung



$\vec{OP}_2$ (n) = $\vec{OP}_2$ (n-1) + $v_1 \cdot \Delta t \cdot \vec{f}_0$ (n-1)

wobei f_0 ein Einheitsvektor ist

1)

Allgemein: Bestimmung eines zu $\begin{pmatrix} a \\ b \end{pmatrix}$ senkrechten Vektors:

$\begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} c \\ d \end{pmatrix} = 0$ ==> ac + bd = 0 ==> Es gibt unendlich viele Lösungen.

Eine Lösung ist: c = -b und d = a, also $\begin{pmatrix} -b \\ a \end{pmatrix}$

Eine andere Lösung ist: c = b und d = -a, also $\begin{pmatrix} b \\ -a \end{pmatrix}$

Wodurch unterscheiden sich diese 2 Vektoren? In der Richtung:

$\begin{pmatrix} -b \\ a \end{pmatrix} = - \begin{pmatrix} b \\ -a \end{pmatrix}$

2)

Bestimmung eines zu $\vec{e}_0$ (n-1) senkrechten Einheitsvektors $\vec{f}_0$ (n-1):

$$\vec{e}_0\,(\text{n-1}) = \frac{\overrightarrow{P_1P_2}(n-1)}{r(n-1)} =$$

$$\frac{\begin{pmatrix} x_2(n-1) - x_1(n-1) \\ y_2(n-1) - y_1(n-1) \end{pmatrix}}{r(n-1)} = \frac{\begin{pmatrix} x_2(n-1) - x_1(n-1) \\ y_2(n-1) - y_1(n-1) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

==>

$$\vec{f}_0\,(\text{n-1}) = \frac{\begin{pmatrix} -(y_2(n-1) - y_1(n-1)) \\ x_2(n-1) - x_1(n-1) \end{pmatrix}}{r(n-1)} = \frac{\begin{pmatrix} -(y_2(n-1) - y_1(n-1)) \\ x_2(n-1) - x_1(n-1) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

Also:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\text{n}) = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\text{n-1}) + \text{sv}_2 \cdot \Delta\text{t} \cdot \frac{\begin{pmatrix} -(y_2(n-1) - y_1(n-1)) \\ x_2(n-1) - x_1(n-1) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

oder:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\text{n}) = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\text{n-1}) + \text{v}_2 \cdot \Delta\text{t} \cdot \frac{\begin{pmatrix} y_2(n-1) - y_1(n-1) \\ -(x_2(n-1) - x_1(n-1)) \end{pmatrix}}{\sqrt{(x_2(n-1) - x_1(n-1))^2 + (y_2(n-1) - y_1(n-1))^2}}$$

Bemerkung:
Um das Modell mehr der Realität anzupassen und um die Verfolgung interessanter werden zu lassen, verpa0t man dem Verfolger in den folgenden Modellierungen verschiedene Handicaps (z.B. Totzeit, Verzögerungszeit, usw.), die ihn bei der Verfolgung behindern.

10.3.3 Modellierung 3

10.3.3.1 Beschreibung

Handicap (Nachteil) Ozeandampfer
Der Verfolger P_1 kann seine Richtung nicht sofort in Richtung des neuen Ortes von P_2 ändern, sondern er rennt wegen seiner Trägheit (ähnlich der Trägheit eines Ozeandampfers, der erst noch kilometerweit geradeaus schwimmt) erst noch eine Weile z.B. T = 5 Zeitabschnitte in seine bisherige Richtung. Erst danach ändert er seine Richtung in Richtung auf P_2, bevor er wiederum T = 5 Zeitabschnitte in diese Richtung weiter schwimmt, usw.

Zusätzlich soll sich P_2 etwas flexibler bewegen können:
Der Winkel zu dem sich P_2 relativ zu P1_P2 bewegt kann beliebig durch die Konstante deltaW2 vorgegeben werden (er muß nicht notwendig 90° betragen).

10.3.3.2 mathematische Modellierung

n = 1:
$x_1(1)$: anfängliche x-Koordinate von P_1
$y_1(1)$: anfängliche y-Koordinate von P_1
$w_1(1)$: Anfangsrichtung von P_1, z.B. $w_1(1)$ = w_P1(1)_P2(1))
$x_2(1)$: anfängliche x-Koordinate von P_2
$y_2(1)$: anfängliche y-Koordinate von P_2
$w_2(1)$: Anfangsrichtung von P_2, z.B. $w_2(1)$ = w_P1(1)_P2(1)) - 90°
T sei die Anzahl der Zeitpunkte, in denen sich P_1 in die gleiche Richtung bewegt, z.B. T = 10
anz(1) : Anzahl der Zeitpunkte, in denen P1 in die gleiche Richtung fährt, z.B. anz(1) = 0

n > 1:
Fall: anz(n-1) = 0
anz(n) = T-1

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$w_1(n)$ = w_P_1(n)_ P_2(n)

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta t$$

$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

Fall: sonst
anz(n) = anz(n-1)-1

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$w_1(n) = w_1(n-1)$

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta t$$

$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

10.3.4 Modellierung 4

10.3.4.1 Beschreibung

Handicap (Nachteil) Rennfahrer
Der Verfolger P_1 kann seine Richtung nicht sofort in Richtung des neuen Ortes von P_2 ändern, sondern er kann bei jedem Schritt seine Richtung nur höchstens um den maximalen Winkel "deltaW1Absolut" ≥ 0 verändern (in positiver bzw. negativer Uhrzeigerrichtung).

10.3.4.2 mathematische Modellierung

n = 1:
$x_1(1)$: anfängliche x-Koordinate von P_1
$y_1(1)$: anfängliche y-Koordinate von P_1
$w_1(1)$: Anfangsrichtung von P_1, z.B. $w_1(1)$ = w_P1(1)_P2(1))
$x_2(1)$: anfängliche x-Koordinate von P_2
$y_2(1)$: anfängliche y-Koordinate von P_2
$w_2(1)$: Anfangsrichtung von P_2, z.B. $w_2(1)$ = w_P1(1)_P2(1)) - deltaW2

n > 1:
Fall: $| w_1(n-1) - w\_P_1(n-1)\_ P_2(n-1) | \leq$ deltaW1Absolut

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta$$

$w_1(n)$ = w_P_1(n)_ P_2(n)
$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

Fall: sonst
Unterfall: wd(w_1(n-1) + deltaW1Absolut)) < wd(w_1(n-1) - deltaW1Absolut))

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n-1)) \\ \sin(w_1(n-1)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta$$

$w_1(n) = w_1(n-1)$ + deltaW1Absolut
$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

Unterfall : sonst

$$\overrightarrow{OP_1(n)} = \overrightarrow{OP_1(n-1)} + \begin{pmatrix} \cos(w_1(n)) \\ \sin(w_1(n)) \end{pmatrix} \cdot v_1 \cdot \Delta t$$

$$\overrightarrow{OP_2(n)} = \overrightarrow{OP_2(n-1)} + \begin{pmatrix} \cos(w_2(n-1)) \\ \sin(w_2(n-1)) \end{pmatrix} \cdot v_2 \cdot \Delta$$

$w_1(n) = w_1(n-1)$ - deltaW1Absolut
$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

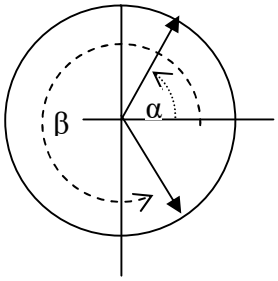
Bem:
Das Problem:

Wenn sich der Vektor $\overrightarrow{v\alpha}$ um einen bestimmten Winkel auf $\overrightarrow{v\beta}$ zu bewegen soll, kann er sich in oder gegen die Uhrzeigerrichtung bewegen. Er soll sich in die Uhrzeigerrichtung bewegen, so daß sich die Winkeldifferenz zwischen den Vektoren kleiner wird.

Wenn sich im Beispiel (siehe Zeichnung) $\overrightarrow{v\alpha}$ um 10° bewegen soll, so muß das gegen die Uhrzeigerrichtung geschehen, damit sich die Vektoren möglichst "nahe" kommen.

Wenn man den Winkel zwischen 2 Vektoren $\overrightarrow{v\alpha}$ und $\overrightarrow{v\beta}$ berechen soll, gibt es immer zwei Winkel (siehe Zeichnung). Es soll der kleinere Winkel (Winkeldifferenz wd) bestimmt werden:

Beispiel:
α = 60° , β = 300°
| α - β | = | 60° - 300° | = 240°
also:
wd(α , β) = wd(60° , 300°) = 360° - 240° = 120°



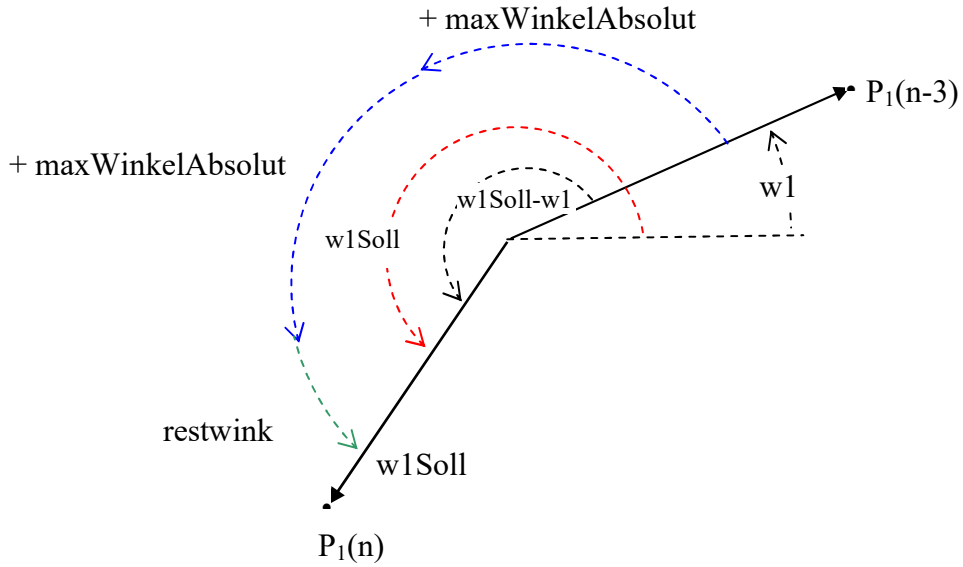
wd(α , β) = | α - β |, falls | α - β | < 180°
= | 360° - |α - β| |, sonst

10.3.5 Modellierung 5

10.3.5.1 Beschreibung

Handicap (Nachteil) Luxusdampfer
Der Verfolger kann seine Richtung nicht sofort in Richtung des neuen Ortes von P2 ändern, sondern er kann bei jedem Schritt seine Richtung (in oder gegen die Uhrzeigerrichtung) nur höchstens um den maximalen Winkel "maxWinkelAbsolut" ≥ 0 in Richtung eines vorgegebenen Sollwerts (Sollwinkel) verändern und zwar so lange bis dieser Sollwert erreicht wurde. Erst dann kann er sich wieder in Richtung P2 bewegen (wobei diese Richtung der neue Sollwert ist). Hat sich P_1 dann in diese Richtung bewegt, muß diese Richtung beibehalten werden, bis der Sollwert erreicht wird.

Beispiel:
P1(n-3) erreicht nicht in einem, sondern in 3 Schritten den Sollwinkel w1Soll.



10.3.5.2 mathematische Modellierung

n = 1:
$x_1(1)$: anfängliche x-Koordinate von P_1
$y_1(1)$: anfängliche y-Koordinate von P_1
$w_1(1)$: Anfangsrichtung von P_1, z.B. $w_1(1)$ = w_P1(1)_P2(1))
w1Soll(1) = Sollrichtung des Punktes P1(1), z.B. w1Soll(1) = w_P1(1)_P2(1))
$x_2(1)$: anfängliche x-Koordinate von P_2
$y_2(1)$: anfängliche y-Koordinate von P_2
$w_2(1)$: Anfangsrichtung von P_2, z.B. $w_2(1)$ = w_P1(1)_P2(1)) + deltaW2

n > 1:
Fall: | w1Soll(n-1)-w1(n-1) | < maxWinkelAbsolut d.h: Sollwert in einem Schritt erreichbar

$$\overrightarrow{OP_1}(n) = \overrightarrow{OP_1}(n-1) + \begin{pmatrix}\cos(w_1(n-1))\\ \sin(w_1(n-1))\end{pmatrix} \cdot v_1 \cdot \Delta t$$

$$\overrightarrow{OP_2}(n) = \overrightarrow{OP_2}(n-1) + \begin{pmatrix}\cos(w_2(n-1))\\ \sin(w_2(n-1))\end{pmatrix} \cdot v_2 \cdot \Delta$$

$w_1(n)$ = w1Soll(n-1)
w1Soll(n) = berechneWinkel des Vektors P1(n)_P2(n)
$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

Fall: sonst, d.h: Sollwert nicht in einem Schritt erreichbar
maxWinkel(n) = berechneMaxwinkel(w_1(n-1), w1Soll(n-1))

$$\overrightarrow{OP_1}(n) = \overrightarrow{OP_1}(n-1) + \begin{pmatrix}\cos(w_1(n-1))\\ \sin(w_1(n-1))\end{pmatrix} \cdot v_1 \cdot \Delta t$$

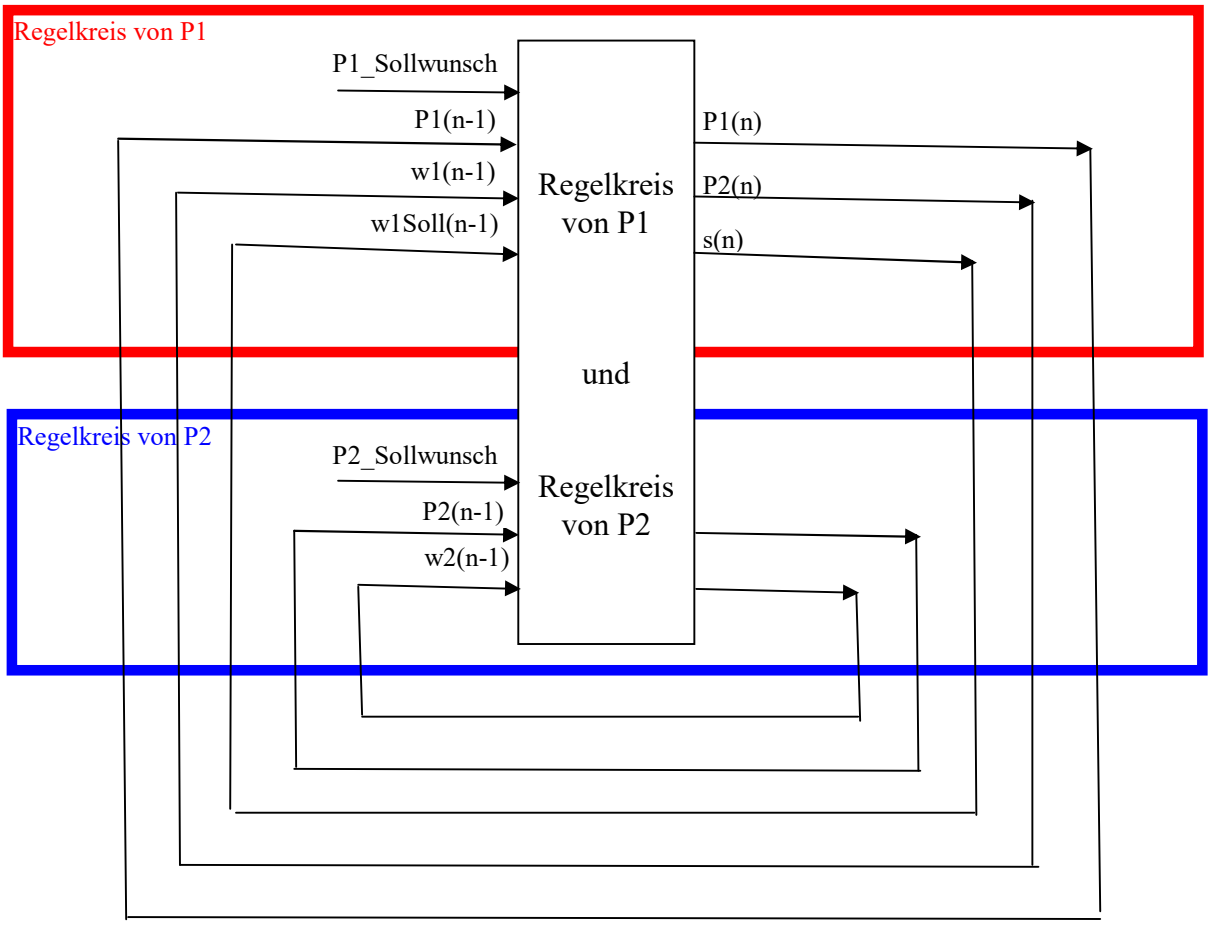
$$\overrightarrow{OP_2}(n) = \overrightarrow{OP_2}(n-1) + \begin{pmatrix}\cos(w_2(n-1))\\ \sin(w_2(n-1))\end{pmatrix} \cdot v_2 \cdot \Delta$$

$w_1(n) = w_1(n-1)$ + maxWinkel(n)
w1Soll(n) = w1Soll(n-1)
$w_2(n)$ = w_P_1(n)_ P_2(n) + deltaW2

Bemerkung:

```
berechneMaxwinkel(w1, w1Soll) {
  if (w1Soll < w1)
    maxWinkel = -maxWinkelAbsolut;
  else
    maxWinkel = maxWinkelAbsolut;
}
```

10.3.5.3 Kybernetische Darstellungen



Beschreibung:
1)
P1 Sollwunsch: P1 = P2
P2 Sollwunsch: P1 ≠ P2

2)
Beim "Durchschalten" liegt am Input $P_1(n-1)$, $w_1(n-1)$, usw. an und es wird am Output $P_1(n)$, $w_1(n)$, usw. erzeugt.
Nach dem Durchschalten liegt am Input und am Output der gleiche Wert $P_1(n)$, $w_1(n)$, usw. an (die "Leitungen" sind verbunden).

10.3.6 Weitere mögliche Modellierungen (angedeutete Vorschläge)

1)
Man könnte den maximalen Winkel, um den sich P1 drehen kann, von der Geschwindigkeit (die dann - wie bisher - dann keine Konstante mehr ist), abhängig machen: bei einer hohen Geschwindigkeit wird der maximalen Winkel kleiner (wenn sie zu hoch wird, ist sie 0)
Man könnte dann eine Regelung dafür implementieren (z.B. solange Geschwindigkeit erhöhen, solange Abstand kleiner wird).
D.h. P1 könnte, falls seine Geschwindigkeit zu groß ist (um in einem Schritt die Richtung auf P2 ändern zu können) im nächsten Schritt seine Richtung nur um den "maxWinkelAbsolut" ändern und dann seine Geschwindigkeit weniger werden lassen.
P1 kann also verschiedene Kombinationen von "maxWinkelAbsolut" und Geschwindigkeit v_1 austesten, und diese optimieren.

2)
P2 verändert seine Richtung (verändert seine Richtung nicht konstant um einen bestimmten Winkel), sondern dieser Winkel kann sich z.B. in Abhängigkeit von der Entfernung zwischen P_1 und P_2 ändern.

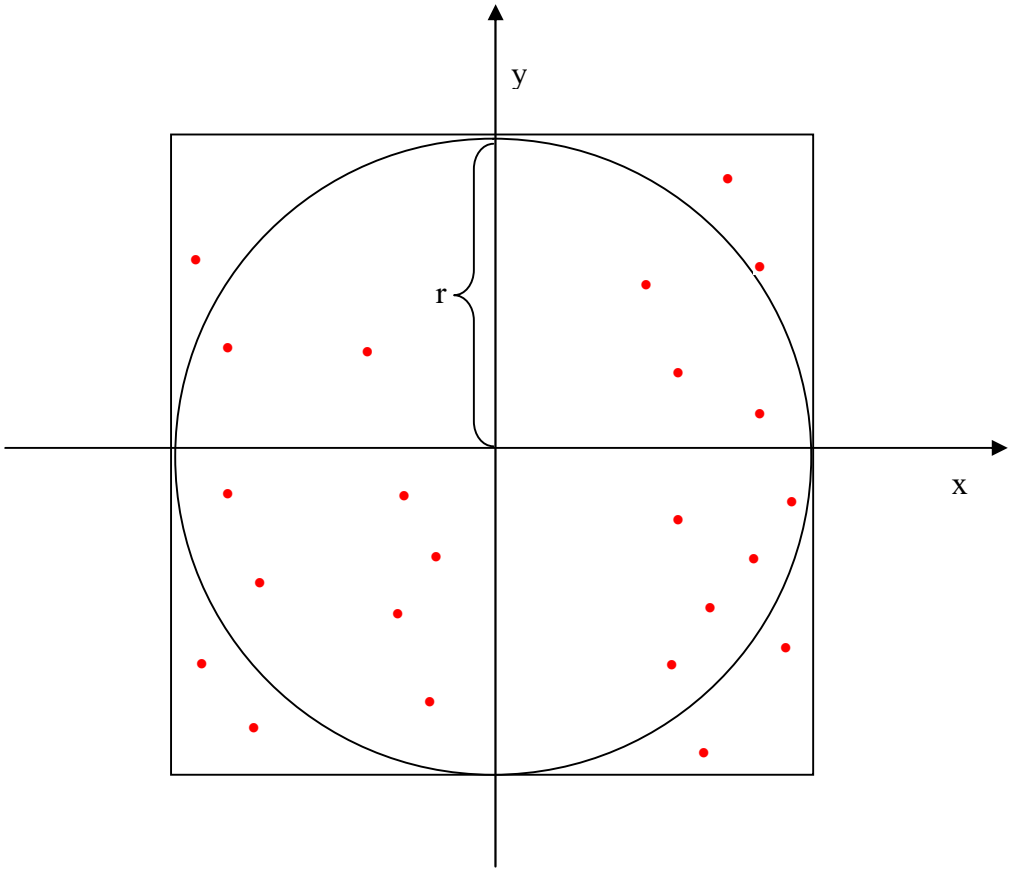
3)
Hohe Wellen lassen P_1 immer nur zu bestimmten Zeitpunkten die Position von P_2 erkennen. Zwischen diesen Zeitpunkten muß P1 deshalb eine "Winkelvorhersage" berechnen, wie z.B. der Durchschnitt der letzten 20 Positionen von P2.

4)
Wenn einer der Punkte aus dem Fenster wandert, werden diese wieder in das Fenster transformiert, so daß die Verfolgung weiter beobachtet werden kann.

5)
Verfolgungsspiele mit mehr als 2 Teilnehmern.

11 Flächenberechnung mit Hilfe des Zufalls

Dies nennt man auch eine Monte-Carlo-Methode.
Wenn man eine Fläche nicht mehr exakt mathematisch berechnen kann oder - was realistischer ist - man nicht mehr die zugehörige Formel kennt, kann man dies auch eine Näherungslösung mit Hilfe eines Zufallsgenerators bestimmen.



Lösungsidee:
Mit verbundenen Augen wirft man "blindwütig" sehr oft (gesamtanzahl-mal) mit Darts-Pfeilen auf das Quadrat, das auf der obigen Zeichnung eingezeichnet ist. Dabei trifft man n-mal die Kreisfläche.
Bautechnisch ist garantiert, daß man immer das Quadrat trifft (z.B. befindet sich das Quadrat am Ende eines das Quadrat umschließenden Tunnels).

Wenn man die Kreisfläche mit A_K , die Fläche des Quadrats mit A_Q , die Anzahl der Treffer mit n und die Gesamtanzahl der Würfe mit "gesamtanzahl" bezeichnet, dann gilt für sehr grosse "gesamtanzahl":

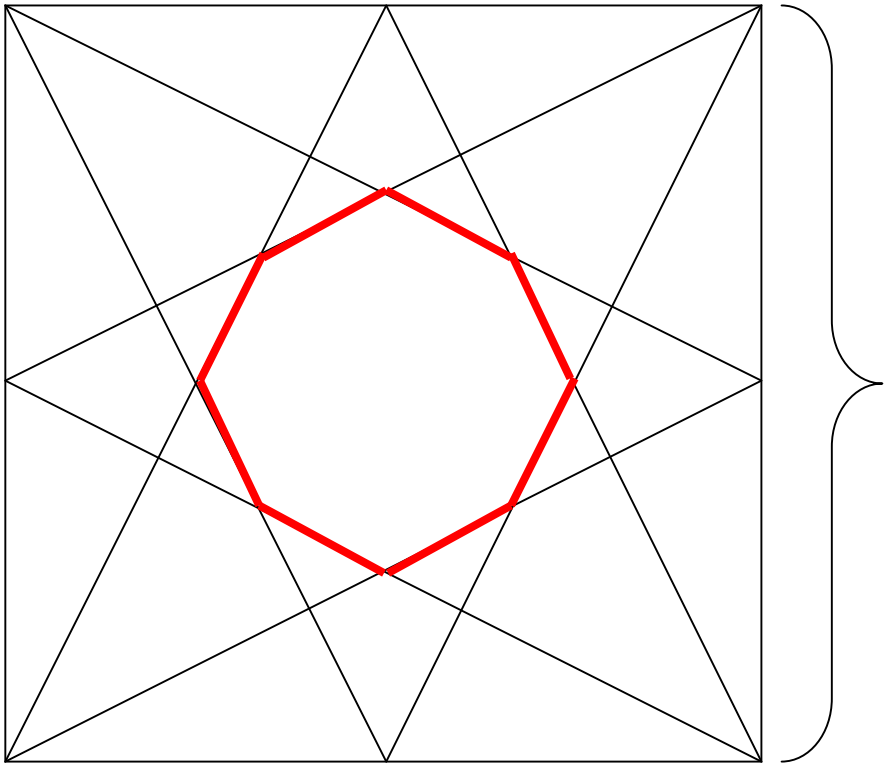
$$\frac{A_K}{A_Q} \approx \frac{n}{gesam\,\tan\,zahl}$$

Da hier $A_Q = 4r^2$, gilt:

$$A_K \approx \frac{n}{gesam\,\tan\,zahl} \cdot 4r^2$$

11.1.1 weitere Aufgabe

Verbindet man jede Ecke eines Quadrates jeweils mit den Mittelpunkten der beiden nicht angrenzenden Seiten, so bilden diese Verbindungslinien im Inneren des Quadrates ein gleichseitiges Achteck. Wie groß ist der Flächeninhalt dieses Achtecks ?



Simulation Beschleunigung und Gravitation

11.2 physikalische Grundlagen

1) Die Geschwindigkeit v(t) zur Zeit t bzw. die danach zurücklegte Strecke s(t) einer mit der konstanten Bescleunigung a bewegren Masse beträgt:
$v(t) = a \cdot t$
$$s(t) = \frac{1}{2} a^2$$

2) Die Kraft, die man benötigt, um eine Masse m mit der Beschleunigung a zu bewegen, beträgt:
$F = m \cdot a$

3) Die Kraft zwischen zwei Massen m_1 und m_2, die den Abstand s (bzgl. ihrer Massenmittelpunkte) haben, beträgt:
$$F = \frac{m_1 \cdot m_2 \cdot \kappa}{s^2}$$

11.3 grundlegende Begriffe zur Simulation

Alle Δt Zeiteinheiten (z.B. Sekunden) hat ein Massenpunkt eine bestimmte Strecke (zurückgelegt), bzw. eine bestimmte Geschwindigkeit und eine Beschleunigung.
Die Zeitpunkte t_n nach denen die zurückgelegte Strecke gemessen werden soll ist also:

$$t_n = n\ \Delta t,\quad n=0,1,2,3,...$$

Die zurückgelegte Strecke zum Zeitpunkt t_n wird mit $x(n) := x(t_n)$ bezeichnet.
Die zu berechnende Geschwindigkeit zum Zeitpunkt t_n wird mit $v(n) := v(t_n)$ bezeichnet.
Die zu berechnende Beschleunigung zum Zeitpunkt t_n wird mit $a(n) := a(t_n)$ bezeichnet.

Trick:
Man nimm (fälschlicherweise) an, dass während eines Zeitabschnitts Δt die Beschleunigung und die Geschwindigkeit konstant sind (sich also nicht ändern).
Bei konstanter Geschwindigkeit kann man die zurückgelegte Strecke des nächsten Zeitpunktes n+1 berechnen.
Genauso kann man bei konstanter Beschleunigung die Geschwindigkeit des nächsten Zeitpunktes n+1 berechnen.
Wenn man den Zeitabschnitt Δt hinreichend klein macht, wird der Fehler, den man macht auch (hoffentlich) nicht allzu groß.

11.4 1D-Simulation einer konstanten Beschleunigung

gegeben:
a : konstante Beschleunigung des bewegten Massenpunktes,
s(0) : anfängliche x-Koordinate des Massenpunktes P.
v(0) : Anfangsgeschwindigkeit des Massenpunktes P.

gesucht:
Berechnen Sie die s(n) und die Geschwindigkeit v(n) nach n Zeiteinheiten

Rechnung:
$v(n+1) \approx v(n) + a \cdot \Delta t$
$s(n+1) \approx s(n) + v(n) \cdot \Delta t$

11.5 1-D-Simulation eines beweglichen mit einem unbeweglichen Punkt.

11.5.1 Eine Möglichkeit

gegeben:
x(0) : anfängliche x-Koordinate des 1. Massenpunktes P_1.
v(0) : Anfangsgeschwindigkeit des 1. Massenpunktes P_1.
Der 2. Massenpunktes P_2 ist fest und befindet sich im Ursprung.
Der 1. Massenpunktes P_1 befindet sich rechts des Ursprungs.

gesucht:
Berechnen Sie die x-Koordinate x(n) und die Geschwindigkeit v(n) nach n Zeiteinheiten

Rechnung:
$F(n+1) = m_1 \cdot m_2 \cdot k / x(n+1)^2$
also beträgt die Beschleunigung des bewegten Massenpunktes m_1 :
$a(n) = F(n) / m_1 = m_2 \cdot k / x()^2 := k_1 / (x(n)^2$

damit:
$a(n) = -k_1 / (x(n)^2$
$v(n+1) \approx v(n) + \frac{1}{2} a(n) \cdot \Delta t$
$x(n+1) \approx x(n) + v(n) \cdot \Delta t + \frac{1}{2} a(n) \cdot \Delta t$

11.5.2 andere Möglichkeit der Simulation (siehe Skript Regelungstechnik):

$v(n) \approx (x(n+1) - x(n)) / \Delta t$
$a(n) \approx (x(n+2) - 2x(n+1) + x(n)) / \Delta t^2$

Es gilt:
$a(n) = -k_1 / (x(n)^2$
angenähert ergibt dies:
$(x(n+2) - 2x(n+1) + x(n))/\Delta t^2 \approx -k_1 / (x(n)^2$
$x(n+2) \approx 2x(n+1) - x(n) - k_1 \cdot \Delta t^2 / (x(n)^2$

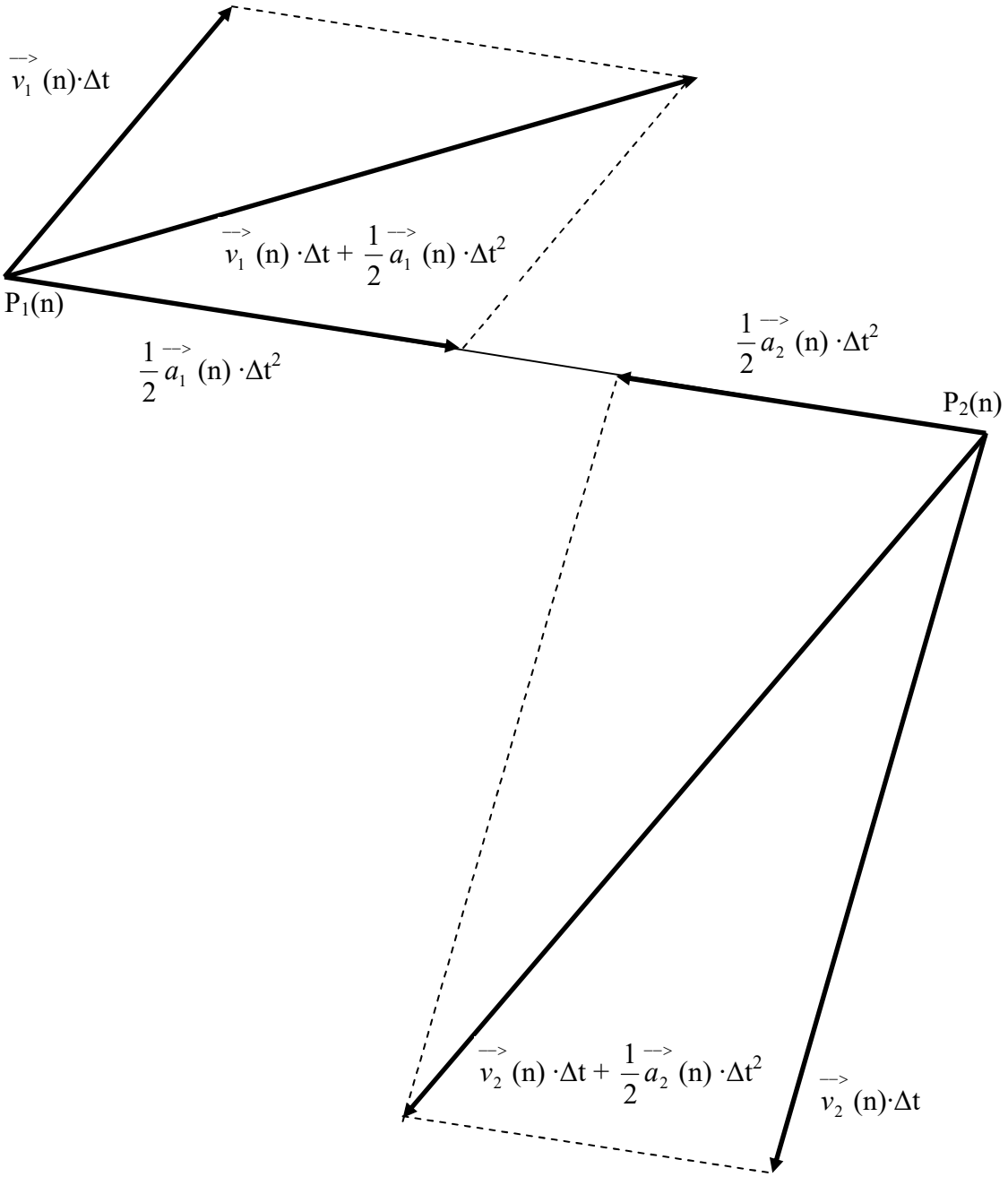
Außerdem läßt sich x_1 wie folgt berechnen:
$v(0) \approx (x(1) - x(0)) / \Delta t$
daraus folgt:
$x(1) \approx v(0) * \Delta t + x(0)$

also:
$x(1) \approx v(0) * \Delta t + x(0)$
$x(n+2) \approx 2x(n+1) - x(n) - k_1 \cdot \Delta t^2 / (x(n)^2$

11.6 2-D-Simulation zweier beweglicher Punkte

Ein Punkt $P_1(x_1 \mid y_1)$ mit der Masse m_1 hat die momentane Geschwindigkeit $\vec{v}_1$.

Ein Punkt $P_2(x_2 \mid y_2)$ mit der Masse m_2 hat die momentane Geschwindigkeit $\vec{v}_2$.
Die zwischen den 2 Punkten P_1 und P_2 herrschende Kraft verursacht während der Zeit Δt die Streckenänderung $\frac{1}{2}\vec{a}_1\,(n)\cdot\Delta t^2$ von P_1 in Richtung P_2 . Gleichzeitig wird P_1 um die Streckenändeung $\vec{v}_1\,(n)\cdot\Delta t$ in Richtung $\vec{v}_1$ bewegt. Analoges gilt umgekehrt.



gegeben:
$x_1(0)$: anfängliche x-Koordinate des Massenpunktes P_1.
$y_1(0)$: anfängliche y-Koordinate des Massenpunktes P_1.
$x_2(0)$: anfängliche x-Koordinate des Massenpunktes P_2.
$y_2(0)$: anfängliche y-Koordinate des Massenpunktes P_2.
$v_{1x}(0)$: Anfangsgeschwindigkeit des Massenpunktes P_1 in x - Richtung.
$v_{1y}(0)$: Anfangsgeschwindigkeit des Massenpunktes P_1 in y - Richtung.
$v_{2x}(0)$: Anfangsgeschwindigkeit des Massenpunktes P_2 in x - Richtung.
$v_{2y}(0)$: Anfangsgeschwindigkeit des Massenpunktes P_2 in y - Richtung.

$\overrightarrow{F_1}$ (n) : Kraft, mit der der Massenpunktes P_1 nach n Zeitabschnitten von P_2 angezogen wird.

$\overrightarrow{a_1}(n) = \begin{pmatrix} a_{1x} \\ a_{1y} \end{pmatrix}(n) = \begin{pmatrix} a_{1x}(n) \\ a_{1y}(n) \end{pmatrix}$: Beschleunigung des Massenpunktes P_1 nach n Zeitabschnitten.

$\overrightarrow{v_1}(n) = \begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix}(n) = \begin{pmatrix} v_{1x}(n) \\ v_{1y}(n) \end{pmatrix}$: Geschwindigkeit des Massenpunktes P_1 nach n Zeitabschnitten.

$\overrightarrow{OP_1}(n) = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}(n) = \begin{pmatrix} x_1(n) \\ y_1(n) \end{pmatrix}$: Ortsvektor des Massenpunktes P_1 nach n Zeitabschnitten.

$\overrightarrow{F_2}$ (n) : Kraft, mit der der Massenpunktes P_2 nach n Zeitabschnitten von P_1 angezogen wird.

$\overrightarrow{a_2}(n) = \begin{pmatrix} a_{2x} \\ a_{2y} \end{pmatrix}(n) = \begin{pmatrix} a_{2x}(n) \\ a_{2y}(n) \end{pmatrix}$: Beschleunigung des Massenpunktes P_2 nach n Zeitabschnitten.

$\overrightarrow{v_1}(n) = \begin{pmatrix} v_{2x} \\ v_{2y} \end{pmatrix}(n) = \begin{pmatrix} v_{2x}(n) \\ v_{2y}(n) \end{pmatrix}$: Geschwindigkeit des Massenpunktes P_2 nach n Zeitabschnitten.

$\overrightarrow{OP_2}(n) = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(n) = \begin{pmatrix} x_2(n) \\ y_2(n) \end{pmatrix}$: Ortsvektor des Massenpunktes P_2 nach n Zeitabschnitten.

gesucht:
Berechnen Sie die x-Koordinate $x_1(n)$, die y-Koordinate $y_1(n)$ und die Geschwindigkeit $v_1(n)$ des 1. Massenpunktes nach n Zeiteinheiten.
Berechnen Sie die x-Koordinate $x_2(n)$, die y-Koordinate $y_2(n)$ und die Geschwindigkeit $v_2(n)$ des 2. Massenpunktes nach n Zeiteinheiten.

Rechnung:

1)

$$\overrightarrow{P_1(n)P_2(n)} = \begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}$$

$$\mathrm{r(n)} := |\overrightarrow{P_1(n)P_2(n)}| = \sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}$$

Für den Einheitsvektor $\overrightarrow{s_0}$ (n) von $\overrightarrow{P_1(n)P_2(n)}$ gilt:

$$\overrightarrow{s_0}\,(\mathrm{n}) = \frac{\overrightarrow{P_1(n)P_2(n)}}{r(n)} = \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{r(n)} = \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{\sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}}$$

Da der Kraftvektor in die gleiche Richtung zeigt und 1LE der Kraft gleich einer LE der Strecke entspricht, ist der Krafteinheitsvektor $\overrightarrow{F_{1,0}}$ (n) gleich dem Streckeneinheitsvektor $\overrightarrow{s_0}$ (n). Also gilt:

$$\overrightarrow{F_{1,0}}\,(\mathrm{n}) = \overrightarrow{s_0}\,(\mathrm{n}) = \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{r(n)}$$

2) Berechnung des Kraftvektors $\overrightarrow{F_1}$ (n):

$$\mathrm{F_1(n)} = |\overrightarrow{F_1}\,(\mathrm{n})| = \frac{\mathrm{m_1 \cdot m_2 \cdot k}}{r(n)}$$

Es gilt allgemein:

$$\overrightarrow{F_1}\,(\mathrm{n}) = \mathrm{F_1(n)} \cdot \overrightarrow{F_{1,0}}\,(\mathrm{n}) = \mathrm{F_1(n)} \cdot \overrightarrow{s_0}\,(\mathrm{n}) = \frac{\mathrm{m_1 \cdot m_2 \cdot k}}{r(n)^2} \cdot \frac{\begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}}{r(n)^2}$$, also

$$\overrightarrow{F_1}\,(\mathrm{n}) = \frac{\mathrm{m_1 \cdot m_2 \cdot k}}{r(n)^3} \cdot \begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix} =$$

$$\frac{\mathrm{m_1 \cdot m_2 \cdot k}}{\sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}^3} \cdot \begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}$$

Für die Beschleunigung des bewegten Massenpunktes gilt:

$$\overrightarrow{a_1}\,(\mathrm{n}) = \frac{\overrightarrow{F_1}(n)}{m_1} = \frac{\mathrm{m_2 \cdot k}}{\sqrt{(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))^2}^3} \cdot \begin{pmatrix} x_2(n) - x_1(n) \\ y_2(n) - y_1(n) \end{pmatrix}$$

Weiter gilt:

$$\overrightarrow{OP_1}\,(\mathrm{n+1}) \approx \overrightarrow{OP_1}\,(\mathrm{n}) + \overrightarrow{v_1}\,(\mathrm{n}) \cdot \Delta\mathrm{t} + \frac{1}{2}\overrightarrow{a_1}(n) \cdot \Delta t^2$$

Da also $\overrightarrow{OP_1}$ (n) in der Zeit Δt um die Strecke $\overrightarrow{v_1}\,(\mathrm{n}) \cdot \Delta\mathrm{t} + \frac{1}{2}\overrightarrow{a_1}(n) \cdot \Delta t^2$ größer wurde, beträgt die Durchschnittsgeschwindigkeit innerhalb des Zeitintervalls [n , n+1]

$$\frac{\frac{1}{2}\overrightarrow{a_1}(n) \cdot \Delta t^2 + \overrightarrow{v_1}(n) \cdot \Delta t}{\Delta t} = \frac{1}{2}\overrightarrow{a_1}(n) \cdot \Delta t + \overrightarrow{v_1}(n)$$

Damit gilt für die neue Geschwindigkeit:

$$\vec{v}_1\,(n+1) \approx \vec{v}_1\,(n) + \frac{1}{2}\,\vec{a}_1\,(n) \cdot \Delta t$$

Umgekehrt gilt dann:

$$\vec{F}_2\,(n) = -\,\vec{F}_1\,(n) = -\,\frac{m_1 \cdot m_2 \cdot k}{\sqrt{(x_2(n)-x_1(n))^2 + (y_2(n)-y_1(n))^2}^3}$$

Damit:

$$\vec{a}_2\,(n) = -\frac{\vec{F}_1(n)}{m_2} = -\frac{m_1 \cdot k}{\sqrt{(x_2(n)-x_1(n))^2 + (y_2(n)-y_1(n))^2}^3}$$

Und damit:

$$\vec{OP}_2\,(n+1) \approx \vec{OP}_2\,(n) + \vec{v}_2\,(n) \cdot \Delta t + \frac{1}{2}\vec{a}_2(n) \cdot \Delta t^2$$

$$\vec{v}_2\,(n+1) \approx \vec{v}_2\,(n) + \frac{1}{2}\,\vec{a}_2\,(n) \cdot \Delta t$$

also insgesamt:

$$\vec{a}_1\,(n) = \frac{m_2 \cdot k}{\sqrt{(x_2(n)-x_1(n))^2 + (y_2(n)-y_1(n))^2}^3} \cdot \begin{pmatrix} x_2(n)-x_1(n) \\ y_2(n)-y_1(n) \end{pmatrix}$$

$$\vec{OP}_1\,(n+1) \approx \vec{OP}_1\,(n) + \vec{v}_1\,(n) \cdot \Delta t + \frac{1}{2}\vec{a}_1(n) \cdot \Delta t^2$$

$$\vec{v}_1\,(n+1) \approx \vec{v}_1\,(n) + \frac{1}{2}\,\vec{a}_1\,(n) \cdot \Delta t$$

$$\vec{a}_2\,(n) = -\frac{m_1 \cdot k}{\sqrt{(x_2(n)-x_1(n))^2 + (y_2(n)-y_1(n))^2}^3} \cdot \begin{pmatrix} x_2(n)-x_1(n) \\ y_2(n)-y_1(n) \end{pmatrix}$$

$$\vec{OP}_2\,(n+1) \approx \vec{OP}_2\,(n) + \vec{v}_2\,(n) \cdot \Delta t + \frac{1}{2}\vec{a}_2(n) \cdot \Delta t^2$$

$$\vec{v}_2\,(n+1) \approx \vec{v}_2\,(n) + \frac{1}{2}\,\vec{a}_2\,(n) \cdot \Delta t$$

oder anders dargestellt:

$$\begin{pmatrix} a_{1x} \\ a_{1y} \end{pmatrix}(\mathrm{n}) = \begin{pmatrix} \dfrac{[x_2(n)-x_1(n)]\cdot k\cdot m_2}{[(x_2(n)-x_1(n))^2+(y_2(n)-y_1(n))]^{1,5}} \\ \dfrac{[y_2(n)-y_1(n)]\cdot k\cdot m_2}{[(x_2(n)-x_1(n))^2+(y_2(n)-y_1(n))]^{1,5}} \end{pmatrix}$$

$$\begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix}(\mathrm{n+1}) \approx \begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix}(\mathrm{n}) + \frac{1}{2}\begin{pmatrix} a_{1x} \\ a_{1y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t}$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}(\mathrm{n+1}) \approx \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}(\mathrm{n}) + \begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t} \;+\; \frac{1}{2}\cdot\begin{pmatrix} a_{1x} \\ a_{1y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t}^2$$

$$\begin{pmatrix} a_{2x} \\ a_{2y} \end{pmatrix}(\mathrm{n}) = -\begin{pmatrix} \dfrac{[x_2(n)-x_1(n)]\cdot k\cdot m_1}{[(x_2(n)-x_1(n))^2+(y_2(n)-y_1(n))]^{1,5}} \\ \dfrac{[y_2(n)-y_1(n)]\cdot k\cdot m_1}{[(x_2(n)-x_1(n))^2+(y_2(n)-y_1(n))]^{1,5}} \end{pmatrix}$$

$$\begin{pmatrix} v_{2x} \\ v_{2y} \end{pmatrix}(\mathrm{n+1}) \approx \begin{pmatrix} v_{2x} \\ v_{2y} \end{pmatrix}(\mathrm{n}) + \frac{1}{2}\begin{pmatrix} a_{2x} \\ a_{2y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\mathrm{n+1}) \approx \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}(\mathrm{n}) + \begin{pmatrix} v_{2x} \\ v_{2y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t} \;+\; \frac{1}{2}\cdot\begin{pmatrix} a_{2x} \\ a_{2y} \end{pmatrix}(\mathrm{n})\cdot \Delta\mathrm{t}^2$$

oder als Koordinatengleichungen dargestellt:

$$a_{1x}(n) = \frac{[x_2(n) - x_1(n)] \cdot k \cdot m_2}{[(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))]^{1,5}}$$

$$a_{1y}(n) = \frac{[y_2(n) - y_1(n)] \cdot k \cdot m_2}{[(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))]^{1,5}}$$

$$v_{1x}(n+1) \approx v_{1x}(n) + \frac{1}{2} a_{1x}(n) \cdot \Delta t$$

$$v_{1y}(n+1) \approx v_{1y}(n) + \frac{1}{2} a_{1y}(n) \cdot \Delta t$$

$$x_1(n+1) \approx x_1(n) + v_{1x}(n) \cdot \Delta t + \frac{1}{2} \cdot a_{1x}(n) \cdot \Delta t^2$$

$$y_1(n+1) \approx y_1(n) + v_{1y}(n) \cdot \Delta t + \frac{1}{2} \cdot a_{1y}(n) \cdot \Delta t^2$$

$$a_{2x}(n) = -\frac{[x_2(n) - x_1(n)] \cdot k \cdot m_1}{[(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))]^{1,5}}$$

$$a_{2y}(n) = -\frac{[y_2(n) - y_1(n)] \cdot k \cdot m_1}{[(x_2(n) - x_1(n))^2 + (y_2(n) - y_1(n))]^{1,5}}$$

$$v_{2x}(n+1) \approx v_{2x}(n) + \frac{1}{2} a_{2x}(n) \cdot \Delta t$$

$$v_{2y}(n+1) \approx v_{2y}(n) + \frac{1}{2} a_{2y}(n) \cdot \Delta t$$

$$x_2(n+1) \approx x_2(n) + v_{2x}(n) \cdot \Delta t + \frac{1}{2} \cdot a_{2x}(n) \cdot \Delta t^2$$

$$y_2(n+1) \approx y_2(n) + v_{2y}(n) \cdot \Delta t + \frac{1}{2} \cdot a_{2y}(n) \cdot \Delta t^2$$

Bemerkung: Analoge Zusammenhänge bestehen zwischen elektrischen Elementarladung.

Aufgaben

1) Erstellen Sie ein Programm, das von einer sich im Punkt (den gibt der Anwender vor) Punkt $R(x_0 \mid y_0)$ befindlichen elektrischen Elementarladung ausgehend die Feldlinie einzeichnet, an der sich die Elementarladung entlang bewegt.

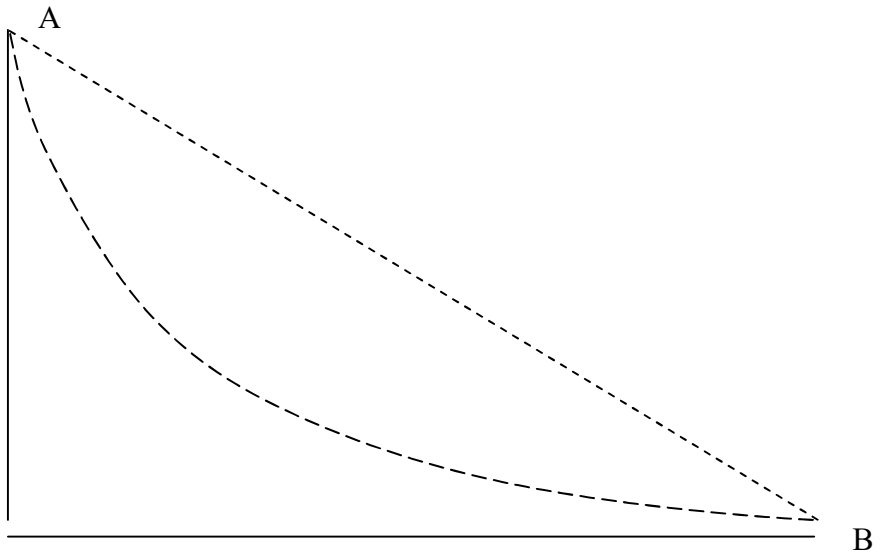
2) Erstellen Sie ein Programm, das von n vorgegebenen elektrischen Teilchen (positive, bzw. negative) die Bewegungen dieser Teilchen (einschließlich dem Auslöschen zweier entgegengestzt geladenen Teilchen beim Zusammenstoss) aufzeichnet.

11.7 Das Brachistochronenproblem

Wie muss eine Bahn beschaffen sein, auf der ein reibungsfreier Körper von einem höher gelegenen Punkt A zu einem niedriger, jedoch nicht senkrecht darunter liegenden Punkt B allein von der Schwerkraft angetrieben die kürzeste mögliche Zeit benötigt?

11.7.1 Aufgabe

Geben Sie beliebige Bahnen vor (z.B. verschiedene Parabeln, die durch A und B gehen) und berechnen Sie die dafür benötigte Zeit.



12 Genetische Algorithmen

12.1 Definition

Ein genetischer Algorithmus versucht die biologischen Evolution nachzuahmen:
Eine Menge (Population) = Elterngeneration) von Lösungskandidaten (Individuen) werden zufällig erzeugt.
Deren Eigenschaften werden leicht verändert (Mutation) und miteinander kombiniert (Rekombination). Die besten dieser Kindergeneration werden ausgewählt (Selektion), um eine neue Population von Lösungskandidaten (eine neue Generation) zu erzeugen.
Auf diese wird dann wiederum die Mutation, Rekombination und Selektion angewendet.
Das wird viele Male wiederholt.
vollständig zu beschreiben, muss die Implementierung folgender fünf Komponenten festgelegt werden:

12.2 Beispiel

Es soll das Minimum der Funktion
f: Z x Z x Z x Z x Z --> Z
berechnet werden, wobei Z die Menge der ganzen Zahlen ist.
mit

$$f(z1, z2, z3, z4, z5) = (z_1 - 1)^2 + (z_2 - 2)^2 + (z_1 - 1)^2 + \sin(z_3) + \cos(z_4) + e^{\frac{z_5^2}{5}}$$

Jede Generation der Population hat die Grösse n

12.2.1 Mutation

Für jede Position
$i \in \{0,1,2,3,4\}$ eines Elternindividuums (Eltern-Genom) eine einfache Addition an dieser Position um eine Zahl $z \in \{-1,0,1\}$.
Diese Mutation komme mit einer Wahrscheinlichkeit von 1% pro Generationswechsel und Position vor.
Beispiel:
GE = (23, -12 , 13 , 7 , -1) ----> GK = (23, -12 , **14** , 7 , -1) sowie i = 3

12.2.2 Rekombination

Aus 2 Eltern-Genomen (zufällig gewählt aus der alten Elterngeneration) wird ein Kind-Genom erzeugt:
Bestimme durch Zufall eine Positon $i \in \{0,1,2,3,4\}$, z.B. i = 2
Die vorderen 2 Elemente des einen Elterngenoms werden mit den 3 hinteren Elementen des anderen Eltergenoms kombiniert.
Beispiel:
G0 = (18, −3, 5 , 9, 8) und G1 = (14, 13, 33, 2, 15) sowie i = 2, dann ist das Kind-Genom GC = (18, −3, 33, 2, 15).

12.2.3 Selektion

Bestimme die n besten Elemente (d.h.die n kleinsten Zahlen) aus der Vereinigung der Elterngeneration und der Kindergeneration.

13 Zelluläre Automaten

13.1 Definition

Ein zellulärer Automat ist ein System von Zellen (z.B. Schachbrett), welche miteinander in Wechselbeziehung stehen. Jede Zelle kann einen von n möglichen Zuständen annehmen, der sich im Lauf der Zeit ändern kann (durch die Wechselbeziehung mit anderen Zellen).

13.2 Beispiel (Game of Life)

Es ist ein quadratisches Brett gegeben (z.B.Schachbrett).
Jede Zelle des Brettes kann mit einem Stein besetzt sein oder nicht. Der Stein wird als Individuum aufgefasst, der auf dieser Zelle "lebt". Die Situation auf der Zelle wird nun nach jeder Generation verändert, wobei für jede Generation alle Zellen gleichzeitig nach folgenden Regeln verändert werden:

1) Ein Stein bleibt stehen ("überlebt die Generation"), wenn auf den (höchstens acht) Nachbarzellen insgesamt genau 2 oder 3 Steine stehen.

2) Ein Stein wird entfernt ("stirbt"), wenn auf den (höchstens acht) Nachbarzellen insgesamt weniger als 2 oder mehr als 3 Steine stehen ("sterben an Isolierung oder Überbevölkerung").

3) Ein Stein wird auf ein bisher leeres Feld gesetzt ("wird geboren"), wenn auf den (höchstens acht) Nachbarzellen insgesamt genau drei Steine stehen.

Beispiel:

0. Generation

			O				
			O				
		O		O			
			O				
			O				

1. Generation

		O	O	O			
		O		O			
		O	O	O			

13.3 Staumodell (Nagel-Schreckenberg)

Eine Ringstrasse (wie z.B. eine Autorennstrecke) wird in z.B. 1000 Zellen der Länge 7,5 Meter aufgeteilt.
In jeder Zelle kann sich ein Auto befinden oder nicht (je nachdem wie man die Situation am Anfang der Simulation modelliert).
Ein Auto besitzt zu einem bestimmten Zeitpunkt eine bestimmte Geschwindigkeitsstufe:
0 , 1 , 2 , ..., maxSpeed (z,B. maxSpeed 5), die in Anzahl Zellen pro Simulationsschritt angegeben ist.
Die Anzahl der freien Zellen zwischen 2 aufeinanderfolgenden Autos wird mit Gap bezeichnet.

Pro Runde werden für alle Fahrzeuge folgende vier Schritte durchgeführt:

I) Modellieren Beschleunigung
Wenn die maximale Geschwindigkeit kleiner ist als das Gap (= Abstand zum vorderen Fahrzeug) dann wird die Geschwindigkeit um 1 erhöht (jedoch nicht, wenn das Fahrzeug schon die maximale Geschwindigkeit besitzt).

II) Modellieren Bremsen:
Falls das Gap zum vorderen Fahrzeug kleiner ist als die Geschwindigkeit (in Zellen), wird die Geschwindigkeit des Fahrzeugs auf die Größe der Lücke reduziert. (Kollisionsfreiheit)

III) Modellieren Trödeln:
Die Geschwindigkeit eines Fahrzeugs wird mit der Wahrscheinlichkeit p um eins reduziert, sofern es nicht schon steht.

IV) Alle Fahrzeuge werden ihrer momentanen Geschwindigkeit entsprechend vorwärts bewegt.

Aufgabe
Schreiben Sie ein Programm, in dem die dynamische Entwicklung der Ringstrasse visualisiert wird. Auf dem Bildschirm soll die Ringstrasse nach 0, 1, 2, ... Simulationsschritten ausgegebn werden. Die Geschwindigkeitsstufen gehen von 0 bis 5 und -1 bedeutet eine leere Zelle (kein Auto)

Beispiel: Füllen Sie den 4.-ten und 5.-ten Simulationsschrittselbst aus.

5.)																				
4.)																				
3.)	-1	1	-1	-1	-1	-1	2	-1	-1	2	-1	-1	-1	3	-1	-1	-1	3	-1	1
2.)	1	-1	-1	-1	2	-1	-1	2	-1	-1	2	-1	-1	-1	3	-1	-1	-1	3	-1
1.)	-1	-1	3	-1	-1	1	-1	-1	4	-1	-1	2	-1	-1	-1	5	-1	-1	-1	1

Man kann z.B. ein stehendes Auto (Stau !!) durch ein rotes Pixel, eine leere Zelle (kein Auto) durch ein gelbes Pixel darstellen, usw. usf.

Bemerkung: (Literaturhinweis)
1)
http://www.site2004.de/java/automat.html

14 Schwarmverhalten

14.1 Motivation

Manche Lebewesen (z.B. Fische, Vögel) bewegen sich in Schwärmen.
Wie ist es möglich, daß sich - trotz fehlender zentraler Steuerung - eine geordnete Bewegung ergibt (wenig Kollissionen, gleiche Richtung) ?
Wissenschaftler wie Craig W. Reynolds haben sich darüber Gedanken gemacht.
Anwendung: Comuterspiele, Animationstechniken, Filme (z.B. Batman 2 (Fledermausschwärme, Pinguine), König der Löwen)).

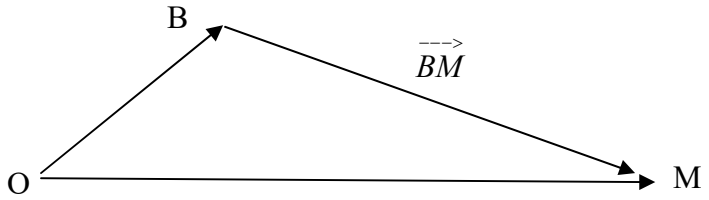
14.2 Eine mögliche Modellierung

Es gibt 3 Regeln, die die einzelnen Individuen (Boids) zu beachten haben:

14.2.1 Regel 1: Anziehung

Sinn: Zusammenhalten des Schwarms
Bewege dich in Richtung des Mittelpunkts derer, die du in deinem Umfeld siehst.

Beispiel: (O ist der Koordinatenursprung)
B ist der Punkt, wo sich das Lebewesen befindet. M ist der von B wahrgenommene Mittelpunkt des Restschwarms (= alle Individuen, außer sich selbst).

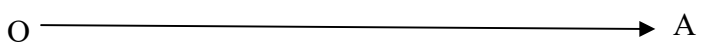


B muss sich also sich also in Richtung des Vektors $\overrightarrow{BM}$ auf den Mittelpunkt M zu bewegen.
$\overrightarrow{BM}$ stellt die Geschwindigkeit (Strecke pro Simulationsschritt) in Richtung M dar.

14.2.2 Regel 2: Kollisionsvermeidung

Sinn: Mitglieder des Schwarms sollen sich nicht verletzen
Bewege dich weg, sobald dir jemand zu nahe kommt.

Beispiel:
B ist der Punkt, wo sich das Lebewesen befindet. Wenn sich B innerhalb eines bestimmten Abstands zu den benachbarten Individuen befindet, wird er von diesen abgestoßen (wie gleichgeladene elektrische Teilchen). Die Resultierende, mit der B von den benachbarten Individuen abgestoßen wird, sei $\overrightarrow{OA}$.



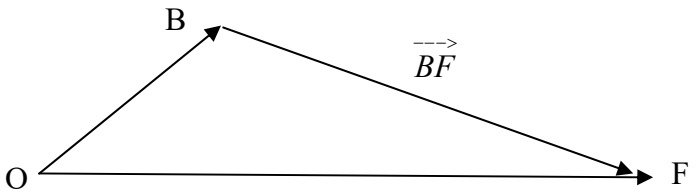
$\overrightarrow{OA}$ stellt eine Geschwindigkeit (Strecke pro Simulationsschritt) dar.

14.2.3 Regel 3: Ausrichten

Sinn: Mitglieder des Schwarms sollen sich in die gleiche Richtung bewegen
Bewege dich in etwa in dieselbe Richtung wie deine Nachbarn.

Beispiel:

B ist der Punkt, wo sich das Lebewesen befindet. $\overrightarrow{OF}$ ist die von B wahrgenommene resultierende Geschwindigkeit des Restschwarms (= alle Individuen, außer sich selbst).



$\overrightarrow{BF}$ stellt eine Geschwindigkeit (Strecke pro Simulationsschritt) in Richtung F dar.

14.3 Eine mögliche Implementierung (Pseudocode)

Bemerkung:
Im folgenden Pseudocode bedeutet sM eine Abkürzung für Schwarmmitglied

14.3.1 Mögliche Implementierung Regel 1

```
regel1(boid b)

    resultierende = 0;
    FOR EACH BOID sM
        IF b != sM THEN
            resultierende = resultierende + sM.position
         END IF
    END

     resultierende = resultierende / (N-1)

     // Geschwindigkeit = Strecke / 100 Simulationsschritte
     RETURN (resultierende - b.position) / 100
END PROCEDURE
```

14.3.2 Mögliche Implementierung Regel 2

```
regel(boid b)
    resultierende = 0;
    FOR EACH BOID sM
        IF b != sM THEN
            IF |b.position - sM.position| < 100 THEN
                resultierende = resultierende + b.position - sM
            END IF
        END IF
    END
    RETURN resultierende
END PROCEDURE
```

14.3.3 Mögliche Implementierung Regel 3

```
PROCEDURE rule3(boid b)
    resultierende = 0;

    FOR EACH BOID sM
        IF b != sM THEN
            resultierende = resultierende + sM.geschwindigkeit
        END IF
    END

    resultierende = resultierende / (N-1)

    RETURN (resultierende - b.velocity) / 8

END PROCEDURE
```

14.3.4 Mögliche Implementierung "berechneNeuePosition"

```
PROCEDURE berechneNeuePositionen()
    Vector v1, v2, v3
    Boid b

    FOR EACH BOID b
        v1 = rule1(b)
        v2 = rule2(b)
        v3 = rule3(b)
        b.v = b.v + v1 + v2 + v3
        b.position = b.position + b.v
    END
END PROCEDURE
```

14.3.5 Mögliche Implementierung Hauptprogramm

```
PROCEDURE main()
    FOR i=0; i<100000; i++
        zeichneAlleBoids()
        berechneNeuePositionen()
    END IF
END PROCEDURE
```

15 Warteschlangen

15.1 Einführung

Einen Großteil unserer Zeit verbringen wir in der Warteschlange: an der Kasse im Supermarkt, im Wartezimmer beim Arzt oder im Verkehrsstau auf dem Weg zur Arbeit. Jeder von uns kennt das Unbehagen, in einer langen Warteschlange zu stehen. Oftmals lässt sich nicht einmal abschätzen, wie lange man noch ausharren muss und ob es sich überhaupt lohnt abzuwarten.
Abgesehen von der psychischen Beeinträchtigung, die von einer langen Warteschlange ausgeht, sind Warteschlangen auch aus volks- bzw. betriebswirtschaftlicher Sicht nicht wünschenswert. Denn die Zeit, die man in der Warteschlange zubringt, ist Untätigkeitszeit, die weder den Kunden noch dem Betreiber des Systems zugute kommt. Den wirtschaftlichen Einfluss von Warteschlangen kann man am besten am Beispiel der Produktion verdeutlichen. Lange Durchlaufzeiten durch die Produktion haben zur Folge, dass man neue Produkte nicht schnell genug auf den Markt bringen kann und der Konkurrenz das Feld überlassen muss. Da lange Durchlaufzeiten mit hohen Beständen korreliert sind, entstehen durch die auf Bearbeitung wartenden Halbfertigfabrikate außerdem hohe Kapitalbindungskosten, die sich negativ auf das Betriebsergebnis auswirken.
Das Phänomen des Wartens wird seit fast einem Jahrhundert wissenschaftlich erforscht. Bereits 1917 publizierte der dänische Ingenieur und Mathematiker A.K. Erlang, der bei einer Kopenhagener Telefongesellschaft beschäftigt war, eine mathematische Formel, mit deren Hilfe man Fernsprechvermittlungsstellen dimensionieren kann. Nach Erlang waren es hauptsächlich Nachrichtentechniker, die mathematische Verfahren benutzten, um den Telefonverkehr durchgängiger und effizienter zu machen. Die Fachleute erzählen sich, dass gerade diejenigen Länder, die Mitte des vergangenen Jahrhunderts über die schlechtesten Telefonsysteme verfügten, zumindest die besten Mathematiker auf dem Gebiet der Warteschlangentheorie hervorgebracht hätten. Mit dem Aufkommen der Datenverarbeitung werden diese Methoden auch zur Konzeption von Rechensystemen verwendet. Ziel der Analysen ist es, bereits im Vorfeld der Planung Engpässe und Schwachstellen zu erkennen. Ganz langsam fängt man an, die Warteschlangentheorie auch auf Fragen der Produktion, des Verkehrs und der Modellierung von Geschäftsprozessen auszudehnen. Inzwischen sind mehr als 10 000 wissenschaftliche Publikationen über Warteschlangenprobleme erschienen, die sich auf die unterschiedlichsten Bereiche unseres täglichen Lebens erstrecken. Angesichts einer so großen Zahl gesicherter Erkenntnisse, lässt sich kaum erklären, warum wir heute noch so oft in der Warteschlange stehen...

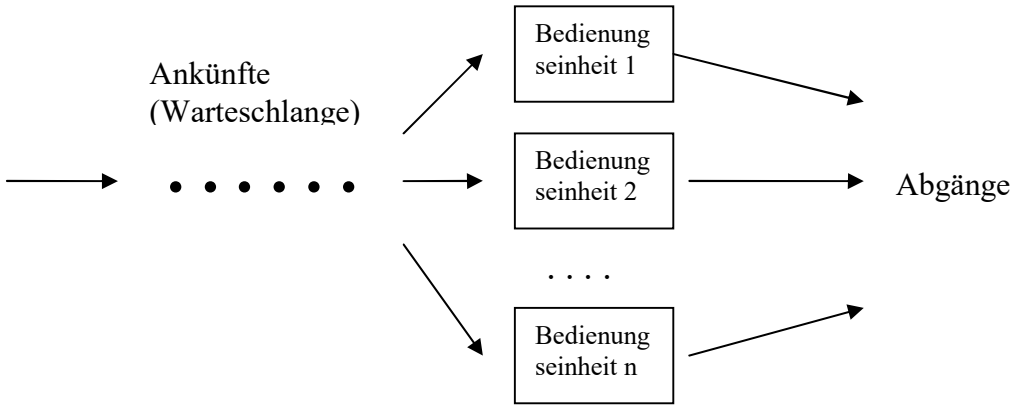
Es sind nicht bloß Menschen, denen im Rahmen der verschiedenen betrieblichen Abläufe das Schicksal des Wartens zuteil wird. Man spricht allgemein von **Jobs** und das können nun sein:
1) Kunden eines Unternehmens, die auf die Erledigung eines Auftrages warten.
2) Produkte, die im Zuge ihrer Produktion auf die Durchführung eines weiteren Verarbeitungsschrittes warten.
3) Nachrichten, Anfragen, Reparatur- und Instandhaltungsaufgaben etc.

15.2 Anwendungsbereiche der Warteschlangentheorie

Warteschlangen lassen sich in vielen Bereichen unseres Alltagslebens beobachten.
Die nachfolgende Tabelle stellt einige dieser Bereiche vor:

Kunde	**Bedienstation**	**Warteschlange**
Zwischenprodukte	Maschine	Puffer vor der Maschine
Skifahrer	Ski-Lift	Ski-Fahrer vor dem Ski-Lift
Flugzeuge	Startbahn	Flugzeuge auf dem Rollfeld
Flugzeuge	Landebahn	Flugzeuge in der Warteschleife
Touristen	Airline-Schalter	Touristen vor dem Schalter
Reisende	Fahrkartenschalter/-automat	Reisende vor dem Schalter
Kunden	Supermarkt-Kasse	Kundenschlange
Fahrzeuge	Tankstelle oder Ampel	Fahrzeugschlange
Patienten	Arzt	Wartezimmer
Bestellungen	Verkaufsabteilung	Lieferverpflichtungen
Druckaufträge	Drucker	Druck-Jobs
Studenten	Mensa	Studenten vor der Essensausgabe
Maschinen	Instandhaltung	Reparaturaufträge
Transportgüter	Transportsystem	Zwischenläger
Lastwagen	Laderampe	Kolonne
Anrufe	Vermittlungsstelle	Anrufwiederholer
Prozesse	CPU	Prozess-Warteschlange

15.3 Unser Modell eines Warteschlangensystems



Die Ankunft erfolge mit der **Ankunftsrate** λ, d.h. im Durchschnitt treffen λ Kunden pro Zeiteinheit ein (und stellen sich in **genau einer Warteschlange** an). Die Bedienung erfolge in n Bedienungseinheiten (z.B. Friseurstuhl), die jede die Bedienungsrate μ hat, d.h. im Durchschnitt μ Kunden pro Zeiteinheit bedient. Dies ist z.B. in einem Frisiersalon der Fall, in welchem die Kunden warten, bis irgendeiner der Friseure frei wird. In Tankstellen und Selbstbedienungsläden hat jede Bedienungseinheit eine eigene Warteschlange.
Wir nehmen in unserem Modell an, daß es genau eine Warteschlange gibt.
Aus theoretischen, hier nicht bewiesenen Überlegungen folgt, dass das System mit einer Warteschlange (auch Queue genannt) jenem mit mehreren Queues überlegen ist!

15.3.1 Notationen

E(W): Erwartungswert (=Mittelwert) der Wartezeit (Zeit vom Eintreffen bis einschließlich der Bedienung eines Kunden) = durchschnittliche Wartezeit

E(L): Erwartungswert (=Mittelwert) der Länge der Warteschlange (einschließlich der Anforderungen, die gerade bedient werden) = durchschnittliche Länge der Warteschlange

λ : Ankunftsrate = Durchschnittliche Anzahl der Ankünfte pro Zeiteinheit.

μ : Bedienungsrate = Durchschnittliche Zahl der von einem Bedienungskanal abgefertigten Jobs pro Zeiteinheit.

n : Anzahl der Bedienungselemente

r : λ/μ

ρ : $= \lambda/(n \cdot \mu)$ = Verkehrsintensität

Bemerkung:
λ ist also ein Maß (sagt also etwas aus) über die **Zugänge** pro Zeiteinheit und μ ein Maß für die **Abgänge** pro Zeiteinheit.

15.4 Mathematik

Seit 1917 hat man unter Einsatz überaus **komplizierter** und **schwieriger** mathematischer Methoden eine ganze Reihe von überraschend einfachen Formeln gefunden.

15.4.1 Fall: Verkehrsdichte > 1

Verkehrsdichte = $\lambda/(n \cdot \mu) > 1$, gleichbedeutend: $\lambda > c\mu$
Es strömen mehr Jobs in das System, als abgefertigt werden können.
Die Länge der Warteschlange wird im Mittel unbeschränkt zunehmen. Das System erweist sich als instabil, man sagt auch, es existiert kein Steady State. Erstaunlicherweise gilt das auch dann, wenn die Verkehrsdichte = 1 ist, also im Mittel pro Zeiteinheit genauso viele Jobs ankommen, als erledigt werden können (es sei denn, Service-und Zwischenankunftszeiten sind konstant und die Ankünfte perfekt auf die Services zeitlich abgestimmt).

15.4.2 Fall: Verkehrsdichte < 1

Fall: Verkehrsdichte = $\lambda/(n \cdot \mu) < 1$
Gleichbedeutend: $\lambda < c\mu$
Es strömen weniger Jobs in das System, als abgefertigt werden können.
Das System bleibt stabil.

15.4.3 Zusammenhang E(L) und E(W) - Formel von Little

$$E(L) = \lambda \cdot E(W)$$

(Little 1961)

15.4.4 Berechnung von E(L) und E(W)

Es gelten bestimmte Voraussetzungen, wie z.B. daß die Warte- und Bedienungszeiten exponentialverteilt sind. Damit ist die Anzahl der innerhalb eines beliebiger Zeitintervall der Laenge T eintreffenden Autos Poisson-verteilt mit Mittelwert lambda·T und kann damit durch eine Binominalverteilung angenähert werden.
Eine gute (diskrete) Simulation durch ein Programm muß dies berücksichtigen.

15.4.4.1 Berechnung von E(W) und E(L) bei n Bedienungseinheiten

r = $\frac{\lambda}{\mu}$ = Ankunftsrate / Bedienungsrate

$$p_0 = \frac{1}{\sum_{i=0}^{n-1} \frac{r^i}{i!} + \frac{r^n}{n!(1-\frac{r}{n})}}$$

$$p_n = p_0 \cdot \frac{r^n}{n!}$$

$$E(L) = \frac{r}{n} \cdot \frac{p_n}{(1-\frac{r}{n})^2}$$

$$E(W) = \frac{E(L)}{\lambda}$$

15.4.4.2 Berechnung von E(W) und E(L) bei einer Bedienungseinheit (n = 1)

$$p_0 = \frac{1}{1+\frac{\lambda/\mu}{1-\lambda/\mu}} = \frac{1}{1+\frac{\lambda/\mu}{(\mu-\lambda)/\mu}} = \frac{1}{1+\frac{\lambda}{\mu-\lambda}} = \frac{1}{\frac{\mu}{\mu-\lambda}} = \frac{\mu-\lambda}{\mu}$$

$$p_n = \frac{\mu-\lambda}{\mu} \cdot \frac{\lambda}{\mu} = \frac{(\mu-\lambda)\lambda}{\mu^2}$$

$$E(L) = \frac{\lambda}{\mu} \cdot \frac{\frac{(\mu-\lambda)\lambda}{\mu^2}}{(1-\frac{\lambda}{\mu})^2} = \frac{\lambda}{\mu} \cdot \frac{\frac{(\mu-\lambda)\lambda}{\mu^2}}{(\frac{\mu-\lambda}{\mu})^2} = \frac{\lambda}{\mu} \cdot \frac{\frac{(\mu-\lambda)\lambda}{\mu^2}}{\frac{(\mu-\lambda)^2}{\mu^2}} = \frac{\lambda}{\mu} \cdot \frac{(\mu-\lambda)\lambda\mu^2}{\mu^2(\mu-\lambda)^2} = \frac{\lambda^2}{\mu(\mu-\lambda)}$$

$$E(W) = \frac{\frac{\lambda^2}{\mu(\mu-\lambda)}}{\lambda} = \frac{\lambda}{\mu(\mu-\lambda)}$$

also:

$$E(L) = \frac{\lambda^2}{\mu(\mu-\lambda)}$$

$$E(W) = \frac{\lambda}{\mu(\mu-\lambda)}$$

15.4.4.3 Berechnung von E(W) und E(L) bei zwei Bedienungseinheiten (n = 2)

$$E(L) = \frac{\lambda^3}{\mu(4\mu^2 - \lambda^2)}$$

$$E(W) = \frac{\lambda^2}{\mu(4\mu^2 - \lambda^2)}$$

15.4.5 Beispiel (Berechnung von E(L) und E(W))

15.4.5.1 Eine Bedienungseinheit

n = 1, λ = 1/3, μ = 1/2

$$E(L) = \frac{\left(\frac{1}{3}\right)^2}{\frac{1}{2}\left(\frac{1}{2} - \frac{1}{3}\right)} = \frac{\frac{1}{9}}{\frac{1}{2} \cdot \frac{1}{6}} = \frac{4}{3}$$

$$E(W) = \frac{E(L)}{\lambda} = \frac{\frac{4}{3}}{\frac{1}{3}} = 4$$

15.4.5.2 Zwei Bedienungseinheiten

n = 2, λ = 1/3, μ = 1/4

$$E(L) = \frac{\left(\frac{1}{3}\right)^3}{\frac{1}{4}\left(4 \cdot \frac{1}{16} - \frac{1}{9}\right)} = \frac{\frac{1}{27}}{\frac{1}{4}\left(\frac{1}{4} - \frac{1}{9}\right)} = \frac{\frac{1}{27}}{\frac{1}{4} \cdot \frac{5}{36}} = \frac{4 \cdot 36}{27 \cdot 5} = \frac{16}{15} \approx 1{,}0666$$

$$E(W) = \frac{\frac{16}{15}}{\frac{1}{3}} = \frac{16 \cdot 3}{15} = \frac{16}{5}$$

ARBEITSBLATT Warteschlangen

A1)
Man bastelt ein würfelähnliches Gebilde mit genau anz Seiten und färbt genau n Seiten davon rot ein.
Man nennt diesen Würfel einen n-anz-Würfel.
1) Wie oft wird **durchschnittlich** bei anz Würfen eine rote Seite gewürfelt?
(d.h. das Ereignis ist eingetreten)
2) Wie groß ist die Wahrscheinlichkeit p bei einem Wurf eine rote Seite zu erwürfeln?

A2)
Die Funktion rand() liefert eine double-Zufallszahl zwischen je einschließlich 0 und RAND_MAX (= 2^{15}-1).
Erstellen Sie mit Hilfe von rand() die Funktion rand_0_1(), die eine Zufallszahl zwischen (je einschließlich) 0 und 1 liefert.

A3)
1) Mit einer Wahrscheinlichkeit von p = 1/6 soll die Meldung "GEWONNEN" ausgegeben werden. Erstellen Sie - mit Hilfe obiger Funktion rand_0_1() - den dazu gehörigen Quellcode.
2) Wie oft soll man eine Simulation durchführen, damit eine bestimmte Wahrscheinlichkeit p modelliert wird, d.h. wie kann man relativ sicher sein, daß die
relative Häufigkeit = $\frac{absolute\ \ Häufigkeit}{anz}$ die Wahrscheinlichkeit p annähert?
Bei 1) sollte z.B. bei anz = 6000 Mal ungefähr 1000 Mal die Sechs gewürfelt werden, damit relative Häufigkeit auch ungefähr 1000/6000 = 1/6 wird.
Außerdem muß noch die Differenz relative Häufigkeit -p berechnet werden.
Ausgabe der Werte dieser Variablen auf dem Bildschirm.
Erstellen Sie den Quellcode, der die absolute Häufigkeit, die relative Häufigkeit und relative Häufigkeit - p berechnet und auf dem Bildschirm ausgibt.

A4)
Wie oft (anzahlSimulationen) soll ein n-anz-Würfel geworfen werden, damit die relative Häufigkeit ungefähr gleich p = n / anz wird ?
Diskutieren Sie das an folgenden Würfeln: 1-3-Würfel, 1-6-Würfel, 1-100-Würfel.
Wie (Formel angeben) hängt anzahlSimulationen von p ab ?

A5) (schwer)
Wie kann man das **durchschnittliche** Auftreten von n Ereignissen (z.B. Blitzeinschläge, Ankunft von Autos an einer Tankstelle, usw.) pro Zeiteinheit (z.B. pro Stunde) durch eine Wahrscheinlichkeit p simulieren ? Kein Programm, sondern Lösung in Worten beschreiben.

A6)
Es soll programmtechnisch das **durchschnittliche** Auftreten von n Ereignissen (z.B. Blitzeinschläge, Ankunft von Autos an einer Tankstelle, usw.) pro Zeiteinheit (z.B. pro Stunde) durch eine Wahrscheinlichkeit p simuliert werden, indem die Zeiteinheit in anz Intervalle geteilt wird.
Die relative Häufigkeit des Ereignisses soll bei einer entsprechenden Anzahl von Simulationen (in Abhängigkeit von n und anz berechnen lassen) auch dem Bildschirm ausgegeben werden.
Ebenso die Differenz der relative Häufigkeit und p.
Erstellen Sie den dazugehörigen Quellcode.

A7) Simulation einer Tankstelle mit genau einer Tanksäule

An einer Tankstelle kommen durchschnittlich jede Minuten $\lambda=2$ Autofahrer an und reihen sich in genau einer Warteschlange ein.

Im Durchschnitt tanken (hintereinander) μ Autos (z.B. $\mu=3$) pro Zeiteinheit und verlassen danach die Tankstelle.

Simulieren Sie programmtechnisch dieses Verhalten wie bei A6)

Lösung Arbeitsblatt:
A1)
1) Bei anz - maligen Werfen des n-anz-Würfels.
2) p = n / anz

A2)
```
double rand_0_1(){
  // RAND_MAX = 2^15-1
  double zufall;
  // Zufallszahl im Intervall [0;1]
  zufall = (double)(rand())/RAND_MAX;
  return zufall;
}
```

A3)
```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>

double rand_0_1();

int main(){
  int i = 0;
  int anzSimulationen=6000;
  double relativeHaeufigkeit;
  int absoluteHaeufigkeit=0;
  double p=1.0/6.0;
  double differenz;

  srand((unsigned)time(NULL));

  for(i=0;i<anzSimulationen;i++){
    if(rand_0_1() < p){
      absoluteHaeufigkeit++;
    }
  }
  relativeHaeufigkeit=(double)absoluteHaeufigkeit/
                                        (double)anzSimulationen;
  differenz = p - relativeHaeufigkeit;
  printf("p=%f\n",p);
  printf("relativeHaeufigkeit=%f\n",relativeHaeufigkeit);
  printf("p-relativeHaeufigkeit=%f\n",differenz);
  return(0);
}

double rand_0_1(){
  // RAND_MAX = 2^15-1
  double zufall;
  // Zufallszahl im Intervall [0;1]
  zufall = (double)(rand())/RAND_MAX;
  return zufall;
}
```

A4)
1-3-Würfel ----> 3000 = 1000 / (1/3)
1-6-Würfel ----> 6000 = 1000 / (1/6)
1-100-Würfel ----> 100000 = 1000 / (1/100)
allgemein: k = 1000
n-anz-Würfel ----> anzahlSimulationen = k / p
Je größer man k wählt, desto besser die Annaherung.
Falls k / p eine Fließkommazahl ergibt, schneidet man den Nachkommateil ab, also
anzahlSimulationen = trunc(k / p)

A5)
Man teilt die Zeiteinheit in anz Intervalle auf. Das bedeutet, daß durchschnittlich n Ereignisse in anz Zeitintervallen auftreten müssen.
Dann bastelt man einen n-anz-Würfel und hat damit die Wahrscheinlichkeit
p = n / anz
mit der dieses Ereignis eintritt.

A6)
```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>

double rand_0_1();

int main(){
  int i = 0;
  int k = 1000;
  int n = 1;
  int anz = 6;
  int anzSimulationen;
  double relativeHaeufigkeit;
  int absoluteHaeufigkeit=0;
  double p=(double)n /(double)anz;
  double differenz;
  srand((unsigned)time(NULL));

  anzSimulationen = (int) (k / p);
  for(i=0;i<anzSimulationen;i++){
    if(rand_0_1() < p){
      absoluteHaeufigkeit++;
    }
  }
  relativeHaeufigkeit =(double)absoluteHaeufigkeit /
                                    (double)anzSimulationen;
  differenz = p - relativeHaeufigkeit;
  printf("p=%f\n",p);
  printf("relativeHaeufigkeit=%f\n",relativeHaeufigkeit);
  printf("p-relativeHaeufigkeit=%f\n",differenz);
  return(0);
}

double rand_0_1(){
     // wie oben
}
```

15.5 Konkretes Bespiel (Tankstelle)

15.5.1 Wahrscheinlichkeitstheorie (Wiederholung Arbeitsblatt)

15.5.1.1 Theorie (Basteln eines Polyeders)

Ein würfelähnliches Gebilde (Polyeder, deutsch Vielflach) mit genau anz Seiten, wobei n Seiten davon rot eingefärbt sind, nennen wir hier einen n-anz-Würfel.
Es wird **durchschnittlich** bei anz Würfen n-Mal eine rote Seite gewürfelt
(d.h. das Ereignis ist n-Mal eingetreten)
Die Wahrscheinlichkeit p bei einem Wurf eine rote Seite zu erwürfeln ist p = n / anz.

15.5.1.2 Anzahl der Simulationen berechnen

Wie oft soll man eine Simulation durchführen, damit eine bestimmte Wahrscheinlichkeit p modelliert wird, d.h. wie kann man relativ sicher sein, daß die

relative Häufigkeit = $\frac{absolute\ \ Häufigkeit}{anz}$

die Wahrscheinlichkeit p annähert?
Beispiele:
1-3-Würfel ----> 3000 = 1000 / (1/3)
1-6-Würfel ----> 6000 = 1000 / (1/6)
1-100-Würfel ----> 100000 = 1000 / (1/100)
allgemein:
n-anz-Würfel ----> anzahlSimulationen = trunc(k / p)
Je größer man k wählt, desto besser die Annaherung.

15.5.1.3 Simulation von Ereignissen pro Zeiteinheit

Problem:
Wie kann man das **durchschnittliche** Auftreten von n Ereignissen (z.B. Blitzeinschläge, Ankunft von Autos an einer Tankstelle, usw.) pro Zeiteinheit (z.B. pro Stunde) durch eine Wahrscheinlichkeit simulieren ?

Lösung:
Man teilt die Zeiteinheit in anz Intervalle auf. Das bedeutet, daß durchschnittlich n Ereignisse in anz Zeitintervallen auftreten müssen.
Dann bastelt man einen n-anz-Würfel und hat damit die Wahrscheinlichkeit
p = n / anz
mit der dieses Ereignis eintritt.

15.5.2 Tankstelle mit einer Warteschlange

15.5.2.1 Voraussetzungen

An einer Tankstelle kommen durchschnittlich jede Minuten 2 Autofahrer an und reihen sich in genau einer Warteschlange ein.
Im Durchschnitt gibt es 3 Tankvorgänge pro Minute.

15.5.2.1.1 Frage

Wie groß ist die mittlere Länge der Warteschlange E(L) ?

Bemerkung:
Es gibt mehrere Möglichkeiten die mittlere Länge der Warteschlange zu "berechnen":
a) Mittlere Länge der Warteschlange für eine Simulation
Bei jedem Simulationsschritt wird die aktuelle Länge der Warteschlange (Anzahl der wartenden Kunden) aufsummiert (über alle Simulationsschritte) und beim letzten Simulationsschritt durch die Anzahl der Simulationsschritte dividiert.
Damit hat man die mittlere Länge der Warteschlange für eine Simulation berechnet.

b) Mittlere Länge der Warteschlange für mehrere Simulationen
Nach jeder Simulation wird die mittlere Länge der Warteschlange notiert.
Die Summe dieser Werte wird dann durch die Anzahl der Simulationen dividiert.

15.5.2.1.2 Frage

Wie groß ist die mittlere Wartezeit E(W) ?

Bemerkung:
Die mittlere Wartezeit kann man wie folgt programmtechnisch ermitteln::
Bei jedem Simulationsschritt wird die Zeit, die die Kunden in der Warteschlange in diesem (einen) Simulationsschritt warten müssen, aufsummiert (über alle Simulationsschritte) und beim letzten Simulationsschritt durch die Anzahl aller Kunden dividiert, die jemals an der Tankstelle angekommen sind. Dies wird hier nicht implementiert.

15.5.2.1.3 Frage

Wie groß ist die Wahrscheinlichkeit, dass ein Autofahrer länger als 10 Minuten warten muß ?

15.5.2.1.4 Frage

Was ändert sich, wenn an der Tankstelle noch eine (mehrere) Zapfsäulen installiert werden ?
Wie groß wird E(L) und E(W), bei $\lambda = 0{,}5$ $\mu=0{,}25$ und $n =2$?

15.5.2.2 Wahrscheinlichkeit simulieren

1) Durchschnittliche Zugang
Im Durchschnitt kommen λ Autos pro Zeiteinheit an der Tankstelle an.
Wie kann man dies „auswürfeln“ ?
Wenn man den Würfel anz Mal wirft, dann soll man durchschnittlich λ Mal erfolgreich sein.
Damit bei jedem Wurf nur maximal ein Auto ankommt (und nicht 2, 3, 4 ...) ist es nötig, anz entsprechend hoch zu wählen.
Die Wahrscheinlichkeit beträgt also ungefähr:

$$\text{pAnkunft} \approx \frac{\lambda}{anz}$$

2) Durchschnittliche Abgang
Im Durchschnitt tanken (hintereinander) μ Autos pro Zeiteinheit und verlassen danach die Tankstelle.
Die Wahrscheinlichkeit beträgt also ungefähr:

$$\text{pBedienung} \approx \frac{\mu}{anz}$$

15.5.2.3 Zum Nachdenken

Bemerkungen:
1) Man kann die mittlere Wartezeit wie folgt berechnen:
Wenn die mittlere Länge der Warteschlange konstant ist (und z.B. nicht gegen unendlich geht), dann muß die mittlere Zugangsrate gleich der mittleren Abgangsrate sein.
Da die mittlere Zugangsrate λ ist, ist die mittlere Abgangsrate auch λ.
Das letzte Auto in der Warteschlange der Länge E(L) muß also E(L) Autos warten, bis dieses Auto endlich abgehen kann. Da pro Zeiteinheit λ Autos abgehen, wird also die Zeit E(L) / λ vergehen.

2) Warum ist die mittlere Abgangsrate nicht gleich der Bedienungsrate μ ?
Weil zwar μ Autos pro Sekunde im Fall einer belegten Zapfsäule abgehen, aber 0 Autos im Fall einer unbelegten Zapfsäule!!

3) Weitere Modellierung:
In einer zukünftigen Version könnte man noch eine Rückkopplung modellieren:
Die Ankunftsrate hängt von der Länge der Warteschlange ab, d.h. wenn ein Autofahrer eine lange Warteschlange sieht, fährt er weiter und stellt sich nicht an.

15.5.2.4 Algorithmus

15.5.2.4.1 Tankstelle mit einer Zapfsäule

Der Zustand der Tankstelle wird durch die Länge der Warteschlange (es soll genau eine geben!!) und die Eigenschaft der Zapfsäule (frei / belegt) charakterisiert.
Dieser Zustand wird durch das Tupel (länge, eigenschaft) angegeben.
An der Tankstelle passieren die folgenden Aktionen:
1) Zugang
Im Durchschnitt kommen pro Zeiteinheit eine bestimmte Menge Autos an der Tankstelle an.

2) Abgang
Im Durchschnitt verlassen (nach dem Tanken) pro Zeiteinheit eine bestimmte Menge Autos die Tankstelle.

3) Aufrücken:
Sobald eine Zapfsäule frei ist, wird diese **sofort** von einem Auto in der Warteschlange belegt, wodurch die Warteschlange um ein Auto kleiner wird.

Der Zustand - beschrieben durch das Paar (länge, eigenschaft) - der Tankstelle am Anfang ist: (0, frei)
Anschließend geschehen immer wiederkehrend die folgenden Aktionen:
Zugang, Aufrücken, Abgang, Aufrücken

In Pseudocode:

```
summe=0;
int länge=0;
zustand = (0, frei);
aufrücken(zustand);
for (i=0; i<anzahlSimulationen; i++){
  zugang(zustand);
  aufrücken(zustand);
  abgang(zustand);
  aufrücken(zustand);
  summe=summe+länge;
}
mittelwert = summe/ anzahlSimulationen;
```

Aktion **zugang_bestimmen**

```
if(Autoankunft wird erwürfelt)
  länge++;
```

Aktion **aufrücken**

```
if(Zapfsäule ist frei und länge > 0){
  eigenschaft = belegt;
  länge --;
}
```

Aktion **abgang_bestimmen**

```
if(Zapfsäule ist belegt){
  if(Autoabgang wird erwürfelt)
    eigenschaft = frei;
}
```

15.5.2.4.2 Tankstelle mit mehreren Zapfsäulen

Der Zugang ist genauso groß wie bei einer Zapfsäule.
Das Aufrücken bezieht sich nun auf alle Zapfsäulen.
Der Abgang wird parallel für alle Zapfsäulen gemacht.

In Pseudocode:

```
summe=0;
zustand = (0, frei);
aufrücken(zustand);
for (i=0; i<anzahlSimulatione; i++){
  zugang(zustand);
  aufrücken(zustand);
  abgang(zustand);
  aufrücken(zustand);
  summe=summe+länge;
}
mittelwert = summe/ anzahlSimulationen;
```

Aktion **zugang_bestimmen**

```
if(Autoankunft wird erwürfelt)
  länge++;
```

Aktion **aufrücken**

```
for(i=0; i<anzahlZapfsäulen; i++){
  if(Zapfsäule[i] ist frei und länge > 0){
    eigenschaft[i] = belegt;
    länge --;
  }
}
```

Aktion **abgang_bestimmen**

```
for(i=0; i<anzahlZapfsäulen; i++){
  if(Zapfsäule[i] ist belegt){
    if(Autoabgang wird erwürfelt)
      eigenschaft[i] = frei;
}
```

15.5.2.4.3 Programmierung in C (1 Zapfsäule)

```
#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include "time.h"

double rand_0_1();
double min(double z1, double z2);

int main()
{
  // Ankunftsrate Lamda (Autos/Zeiteinheit) = Input der Warteschlange
  const double ankunftsrate = 2;
  // Bedienungsrate mü (Autos/Zeiteinheit) = Output der Warteschlange
  const double bedienungsrate = 3;
  // mittlere Länge der Warteschlange
  double mittlereLaengeDerWarteschlange;
  // mittlere Wartezeit in der Warteschlange
  double mittlereWartezeitInDerWarteschlange;
  // mittlere mathematisch berechnete Länge der Warteschlange
  double E_L;
  // mittlere mathematisch berechnete Wartezeit in der Warteschlange
  double E_W;
  // ZapfsÃ¤ule frei = 0, belegt = 1
  int zustandZapfsaeule;
  // ZapfsÃ¤ule frei = 0, belegt = 1
  long anzahlSimulationsschritte = 600000;
  long anzahlIntervalleInZeiteinheit=60;
  long simulationsschritt;
  // Anzahl wartender Kuden = Länge der Warteschlange
  long anzahlWartenderKunden = 0;
  // Summe aller Längen der Warteschlangen
  long summeWartenschlangenLaenge = 0;
  // Wahrscheinlichkeit für einen Zugang
  double pAnkunft;
  // Wahrscheinlichkeit für einen Abgang
  double pBedienung;
  // Faktor für Anzahl der Simulationsschnritte
  int k=10000;
  double zufall;
  // srand bitte nur EINMAL aufrufen !!!!!!
  srand((unsigned)time(NULL));
  pAnkunft = (double)ankunftsrate / (double)anzahlIntervalleInZeiteinheit;
  pBedienung = (double)bedienungsrate
                                    /(double)anzahlIntervalleInZeiteinheit;
  // initialisieren des Zustands
  anzahlWartenderKunden = 0;
  // Zapfsäule ist am Anfang frei
  zustandZapfsaeule = 0;
  anzahlSimulationsschritte = k / min(pAnkunft,pBedienung);

  for (simulationsschritt = 0; simulationsschritt <
                        anzahlSimulationsschritte; simulationsschritt++) {
    // Zugang (für Warteschlangensystem)
    zufall=rand_0_1();
    if (zufall < pAnkunft){
      anzahlWartenderKunden++;
    }

    // Aufrücken der Autos
    if(zustandZapfsaeule==0 && anzahlWartenderKunden > 0){
      zustandZapfsaeule=1;
      anzahlWartenderKunden--;
    }
```

```
    // Abgang (für Warteschlangensystem)
    if(zustandZapfsaeule==1){
      zufall=rand_0_1();
      if (zufall < pBedienung){
        zustandZapfsaeule=0;
      }
    }

    // Aufrücken der Autos
    if(zustandZapfsaeule==0 && anzahlWartenderKunden > 0){
      zustandZapfsaeule=1;
      anzahlWartenderKunden--;
    }

    summeWartenschlangenLaenge = summeWartenschlangenLaenge +
                                                    anzahlWartenderKunden;
  }
  mittlereLaengeDerWarteschlange=(double)summeWartenschlangenLaenge/
                                        (double)anzahlSimulationsschritte;
  E_W = ankunftsrate/(bedienungsrate*(bedienungsrate-ankunftsrate));
  E_L = ankunftsrate * E_W;
  printf("summeWartenschlangenLaenge=%d\n",summeWartenschlangenLaenge);
  printf("mittlereLaengeDerWarteschlange=%f\n",
                                          mittlereLaengeDerWarteschlang);
  printf("E_L = %f\n",E_L);
  printf("E_W = %f\n",E_W);
  printf("pAnkunft = %f\n",pAnkunft);
  printf("pBedienung = %f\n",pBedienung);

  char x;
  scanf("%c",&x);
  return 0;
}


double rand_0_1(){
      // RAND_MAX = 2^15-1
      double zufall;
      // Zufallszahl im Intervall [0;1]
      zufall = (double)(rand())/RAND_MAX;
      return zufall;
}

double min(double z1, double z2){
      if(z1<z2)
            return z1;
      else
            return z2;
}
```

16 Wetten

16.1 Beschreibung

Am 28.6.2012 waren die Quoten in der Fuball-Europameisterschaft für das Spiel Deutschland - Italien wie folgt:

Sieg Deutschland:	1,9
Unentschieden:	3,35
Niederlage Deutschland:	4,7

Das bedeutet:
Wenn man z.B.auf eine Niederlage Deutschlands 100 Euro gesetzt hat, bekommt man
$4{,}7 \cdot 100 = 470$ Euro Geld vom Wettbüro mybet.com
Bei einem Einsatz von 100 Euro hat man also einen Gewinn von
470 Euro - 100 Euro = 370 Euro
gemacht.

16.2 Spielstrategien

Kann man eine Wette so gestalten, daß man auf jeden Fall einen Gewinn macht?
Nein, da der Mensch keine "göttlichen Fähigkeiten" der Prognose besitzt, kann er eine Wette verlieren und damit hat man den Einsatz verloren.
Aber man kann mehrere Wetten abschließen und damit eine **Gesamtwette** bilden, die aus mehreren **Teilwetten** besteht.
Kann man eine Gesamtwette so gestalten, dass man auf jeden Fall einen Gesamtgewinn macht?
Ja, falls die Quoten es zulassen (siehe folgendes Besipiel)

Gewinnbringende Gesamtwette

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
10	10	10
100	200	300
= 10*100-(100+200+300) = 400	= 10*200-(100+200+300) = 1400	= 10*300-(100+200+300) = 2400

Minimaler Gesamtgewinn = 2400

Definition:
Eine Teilwette eines Gesamtwette heißt **potentiell gewinnbringend** genau dann wenn bei einem Gewinn dieser Wette der Geamtgewinn > 0 ist.
Man muß also die Einsätze so bilden, daß alle 3 Wetten jeweils potentiell gewinnbringend sind.
Da das Wettbüro die Quoten so berechnet, daß dies nie der Fall sein wird (sonst wäre das Wettbüro pleite), wird die Strategie gewählt, die eine möglichst große Anzahl (also 2) von Teilwetten jeweils potentiell gewinnbringend macht.
Unter diesen wählt man dann die mit dem größten Gewinn aus.

16.3 Eine Spielstrategie

Bei einem vorgegebenen maximalem Verlust, den die Gesamtwette nicht überschreiten darf, werden 2 potentiell gewinnbringende Teilwetten gesucht, so daß deren minimaler potentiell gewinnbringender Gesamtgewinn möglichst groß wird.

Beispiel

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	5
50	200	100
= 2*50-(50+200+100) = -250	= 3*200-(50+200+100) = 250	= 5*100-(50+200+100) = 150

Minimaler Gesamtgewinn = -250
Minimaler potentiell gewinnbringender Gesamtgewinn = 150

Wichtige Frage:
Kann man eine Gesamtwette finden, die einen minimaler Gesamtgewinn nicht unterschreitet (also garantiert, daß ein möglicher Verlust nicht überschritten wird) und der minimale potentielle gewinnbringende Gesamtgewinn möglichst groß werden läßt.
Dies könnte man -sofern man genügend Zeit hat - durch "Probieren" herausfinden.

16.4 Konkrete Aufgabe

Gesucht: Eine Gesamtwette mit
Minimalem Gesamtgewinn = -20 und minimalem potentiell gewinnbringendem Gesamtgewinn.

Suche durch Probieren

Wette 1

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
4	7	9
= 2*4-(4+7+9) = -12	= 3*7-(4+7+9) = 1	= 4*8-(4+7+9) = 12

Minimaler Gesamtgewinn = -12
Minimaler potentiell gewinnbringender Gesamtgewinn = 1

Wette 2

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
4	10	11
= 2*4-(4+10+11) = -17	= 3*10-(4+10+11) = 5	= 4*11-(4+10+11) = 19

Minimaler Gesamtgewinn = -17
Minimaler potentiell gewinnbringender Gesamtgewinn = 5

Wette 3

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
0	11	8
= 2*0-(0+11+8) = -19	= 3*11-(0+11+8) = 14	= 4*8-(0+11+8) = 13

Minimaler Gesamtgewinn = -19
Minimaler potentiell gewinnbringender Gesamtgewinn = 13

Bis jetzt ist Wette 3 am besten.

Aufgabe:
Schreiben Sie ein Programm, das durch "Brute Force" eine optimale Wette bestimmt.