Woher hat das Zebra seine Streifen?

Rainer Huisken[*], Volkhard Nordmeier[°] und H. Joachim Schlichting

Kurzfassung
Es wird ein durch chemische Reaktionen realisierter Mechanismus beschrieben, der auf Alan Turing zurück geht und als Grundlage für die Entstehung von Strukturen auf Tierfellen, Fischen u.ä. angesehen werden kann. Zunächst wird das mit schulischen Mitteln kaum zugängliche Realexperiment beschrieben und erklärt. Anschließend wird eine einfach zu handhabende Computersimulation vorgestellt, mit der die wesentlichen Aspekte der Musterbildungsvorgänge nachgestellt werden können. (Die für die Simulation nötige Software befindet sich inklusive Beschreibung auf der CD im Verzeichnis: /CD_DPG2003/AusUndFuerDenUnterricht/ HuiskenDD5_5/Simulationen/)

1 Turingstrukturen in der CIMA-Reaktion

Der Mathematiker Alan Turing beschäftigte sich Anfang der fünfziger Jahre des 20. Jahrhunderts mit der Morphogenese, dem Studium von der Ausgestaltung und Entwicklung von Organen oder Geweben eines tierischen oder pflanzlichen Organismus. Die zentralen Fragen der Morphogenese sind: Wie entsteht aus einer kugelförmigen befruchteten (Ei)-Zelle ein Lebewesen? Wie entstehen Strukturen auf der Oberfläche von Lebewesen? Oder: Wie kommt es zum Symmetriebruch?
1952 entwarf er in einer Veröffentlichung mit dem Titel: „The Chemical Basis of Morphogenesis“**[1]** ein System hypothetischer chemischer Reaktionen, das spontanen Symmetriebruch hervorbringt und zu stabilen räumlichen Strukturen in einem anfänglich homogenem Gemisch chemischer Komponenten führt. Es handelt sich um einen der einflussreichsten Artikel der theoretischen Biologie ([5], S.79).
Der experimentelle Nachweis solcher Strukturen gelang Castets et al. erst fast 40 Jahre später. 1990 veröffentlichten Sie einen Artikel [6], aus dem auch das folgende Experiment stammt. Die lange Zeit, die zwischen Turings Postulat und der Entdeckung der Strukturen in einem Realexperiment liegt, ist zum einen auf die speziellen Anforderungen, die an die Reaktanten gestellt werden, zurückzuführen (s.u.). Zum anderen erscheinen Turingstrukturen in diesem Experiment nur in einem schmalen Temperatur- und Konzentrationsbereich.

Versuchsbeschreibung
Eine Scheibe aus mit Stärke versetztem Gel (z.B. Polyacrylamid) steht an den beiden gegenüberliegenden Seiten in Kontakt mit einer Lösung A bzw. B. Die beiden Lösungen werden jeweils ständig in einem CSTR (Continiously fed Stirred Tank Reaktor) gut durchmischt, auf der Anfangskonzentration gehalten und auf konstant 7° C gekühlt (vgl. Abb.1).

Die wässrigen Lösungen beinhalten u.a. folgende Ausgangsstoffe:

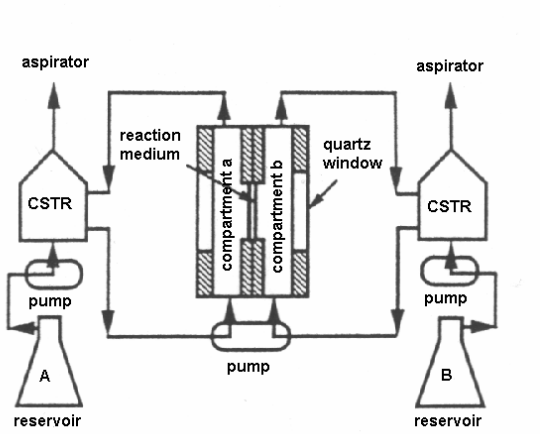

Abb.1: Skizze einer CSTR-Apparatur zum Nachweis von Turingstrukturen. Die Reaktion findet in einer dünnen Schicht (Gelscheibe) zwischen den beiden Kammern a und b in der Bildmitte statt (aus [3]).

Lsg. A: ClO_2^-, I^-
Lsg. B: $CH_2(COOH)_2$, I^-, H^+

Beobachtung
Man beobachtet, dass sich innerhalb von wenigen Stunden auf der Gelscheibe (reaction medium; vgl. Abb.1) stationäre, räumliche Muster ausbilden, die nicht orientiert zu sein scheinen und eine Wellenlänge im 1/10 mm-Bereich aufweisen, was deutlich kleiner als die Ausmaße der Gelscheibe ist. Je nach Reaktionsbedingungen handelt es sich bei den Mustern um Streifen, Hexagone oder Mischformen der beiden Muster. Zwei Beispiele sind in Abb.1 zu sehen. Hierbei sind die dunklen Bereiche blau und die hellen haben die hellgelbe, ursprüngliche Farbe der Gelscheibe.
Die Muster bleiben bei fortschreitender Versuchsdauer konstant. Trennt man die Scheibe von den Lösungen, so leitet dies die Homogenisierung der Struktur ein. Weiterhin reagiert das System sensibel auf Temperaturveränderungen, was neben geringfü-

gigen Veränderungen der Wellenlänge auch die Zerstörung der Struktur zur Folge haben kann.

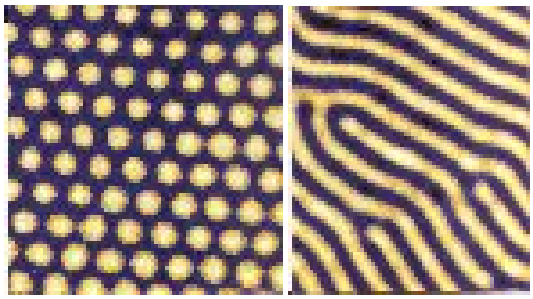

Abb. 2: Turing-Strukturen (Streifen und Hexagone) in der CIMA-Reaktion (aus [8])

Deutung

Die tiefblaue Farbe des Musters wird durch einen I_3^--Stärke-Komplex hervorgerufen. Das Muster selbst verdankt sich Konzentrationsunterschieden des Iodids (I^-), da auch das elementare Iod (I_2) relativ gleichmäßig verteilt ist.

$$I^- + I_2 + \text{Stärke} \rightarrow I_3^-\text{-Stärke- Komplex (tiefblau)}$$

Turing [1] sagte voraus, dass die Kopplung von Diffusion mit einer passenden Reaktionskinetik solche stabilen, inhomogenen räumlichen Strukturen hervorbringen kann. Dabei müssen in dem System ein *Aktivator* und ein *Inhibitor* vorliegen. Der *Aktivator* ist eine Substanz, welche die eigene Produktion verstärkt, beim *Inhibitor* handelt es sich hingegen um einen Stoff, der die eigene Produktion hemmt. Der *Inhibitor* muss in diesem System die Eigenschaft haben, deutlich schneller zu diffundieren als der *Aktivator*:

$$D_i >> D_a \qquad \text{(vgl. [7], S.13142).}$$

Nach der Beobachtung von Turingstrukturen in der CIMA- Reaktion [6] stellten Lengyel et al. eine Theorie über die chemischen Abläufe des Systems vor [2]. Demnach lassen sich die Reaktionen in einem vereinfachten Modell zusammenfassen:

$$MA + I_2 \rightarrow IMA + I^- + H^+ \qquad (1)$$
$$ClO_2 + I^- \rightarrow ClO_2^- + \tfrac{1}{2} I_2 \qquad (2)$$
$$ClO_2^- + 4\,I^- + 4H^+ \rightarrow Cl^- + 2\,I_2 + H_2O \qquad (3)$$

Dabei bezeichnet MA Malonsäure $CH_2(COOH)_2$.

Experimentell wurde herausgefunden, dass bis auf die Konzentrationen von Iodid I^- und Chlorit ClO_2^- alle anderen Stoffkonzentrationen auch lokal relativ konstant und zudem homogen verteilt sind, so dass sich das Modell auf zwei Variablen beschränken lässt:

Bezeichnet man mit X= I^-(*Aktivator*) und Y = ClO_2^- *(Inhibitor*), so gilt:

$$\rightarrow X \qquad (1')$$
$$X \rightarrow Y \qquad (2')$$
$$4X + Y \rightarrow \qquad (3')$$

Aus diesem Modell lassen sich Gleichungen für die zeitliche Änderung der Konzentrationen von X und Y formulieren (vgl. [2]). Für dieses Gleichungssystem kann man eine homogene, stationäre Lösung finden, was einem ortsfesten Muster mit konstanten Konzentrationen von [I^-] und [ClO_2^-] entspricht. Diese Lösung legt nun die Parameter wie Ausgangskonzentrationen der Reaktanten oder Reaktionsgeschwindigkeiten fest. Letztere hängen von der Temperatur ab und bedingen die Temperaturempfindlichkeit der Struktur.

Damit Turingstrukturen überhaupt auftreten, muss neben den eben genannten Bedingungen die weiter oben erwähnte Ungleichung $D_i >> D_a$ erfüllt sein.

Die Diffusionskonstante D kann normaler Weise durch die Stokes-Einstein-Beziehung berechnet werden:

$$D = \frac{kT}{6\pi\eta a}$$

Mit η = Viskosität des Mediums, T = Temperatur, a = Teilchenradius, k = Boltzmannkonstante

Für die vorliegenden Iodid- und Chloritionen wäre D_i/D_a aufgrund des deutlich kleineren I^--Ions sicher kleiner als 1.

An dieser Stelle wird die zweite Funktion der Stärke deutlich. Der Indikator, der mit Iod den tiefblauen I_3^--Stärke-Komplex bildet, ist im Gel unbeweglich. Kommt es zur Komplexbildung, wird auch das komplexierte Iodidion ortsfest und kann sich erst durch den Zerfall des Komplexes aus der Umarmung der Stärke lösen und wieder durch das Gel bewegen. Die Diffusionsgeschwindigkeit des Iodids wird durch dieses Phänomen stark herabgesetzt.

Die Tatsache, dass Stärke für die Erfüllung der Bedingung $D_i >> D_a$ verantwortlich ist, erklärt, warum Turingstrukturen in chemischen Systemen schwer zu finden waren. Die Diffusionskonstanten sind bei vergleichbaren Teilchengrößen ähnlich ($D_i \approx D_a$). Zudem wären die Strukturen bei geringen Viskositäten des Mediums und daraus resultierenden hohen Diffusionskonstanten, wie beispielsweise in wässrigen Lösungen, mikroskopisch klein.

Die Strukturbildung kann im Beispiel der CIMA-Reaktion stattfinden, da chemische Energie dissipiert wird. Dem chemischen System werden ständig Malonsäure, Jod und Dichloroxid ClO_2 zugeführt. Als entwertete Produkte entstehen Iodmalonsäure, Chlorid Cl^- und Wasser. Die Turingstrukturen treten in einem relativ kleinen Parameterbereich auf. Um in diesen Bereich zu gelangen, müssen mehrere kritische Punkte überschritten werden.

2 Computersimulation von Turingstrukturen

Der hohe technische Aufwand, der sich durch den Versuchsaufbau (Abb.1) erahnen lässt, die Instabilität der Strukturen bei Zimmertemperatur und ihre

Dimension im 1/10-mm-Bereich lassen das CIMA-Experiment für die Schule ungeeignet erscheinen. Als Alternative stellen wir im folgenden eine Computersimulation vor, die die Entstehung zweidimensionaler Turingstrukturen auf einer rechteckigen Darstellungsfläche visualisiert. Sie entspricht der quasi zweidimensionalen Gelscheibe in der CIMA-Reaktion. Der Aktivator wird in der Simulation blau, der Inhibitor gelb dargestellt.
Die Simulation wurde in der Programmiersprache Pascal unter Einbindung der Unit *japi.pas* geschrieben. *japi.pas* stellt Pascalbefehle zur Verfügung, die über die Datei *japi.dll* auf Java-Funktionen zurückgreifen, so dass auf unkompliziertem Wege die Benutzeroberfläche gestaltet werden kann[1]. Als Compiler für das Programm diente *Free Pascal*.

2.1 Die Möglichkeiten des Programms

Die vorliegende Simulation berechnet auf der Grundlage mehrerer Parameter in Echtzeit die Entstehung einer Struktur, sofern die Parameter zu einer solchen führen. Der Anwender hat die Möglichkeit, vier dieser Parameter zu variieren. Es handelt sich hierbei um die Parameter „*a*“ und „*b*“ (vgl. Abschnitt 2.4) sowie die *Breite* und *Höhe* der Reaktionsfläche. Der Parameter *a* ist proportional zur Malonsäurekonzentration und *b* zur Iodkonzentration im Realexperiment. Zusätzlich kann man auswählen, ob die Ränder der Reaktionsfläche durchlässig, undurchlässig oder aber konstant gehalten werden sollen.

2.2 Bedienung des Programms

Beim Start des Programms öffnet sich das Hauptfenster, welches eine Menüleiste mit den Menüs *Datei* und *Simulation* enthält (vgl. Abb. 3). Rechts im Fenster befindet sich die Anzeige der verwendeten Parameter. Hier erscheinen bei Programmstart neben „*a* =“, „*b* =“ und „*randb.*:“ noch keine Werte, da diese erst vom Anwender eingestellt werden müssen.

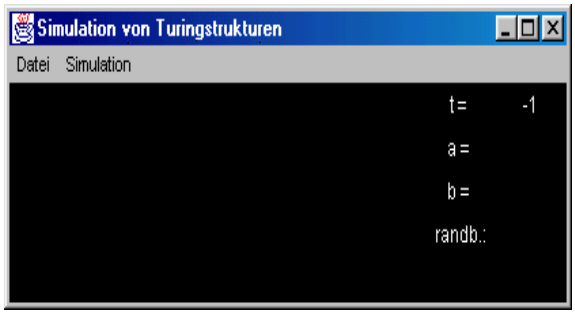

Abb. 3: Startfenster der Simulation

Links im Fenster wird die Reaktionsfläche dargestellt, die ebenfalls noch nicht zu sehen ist, da zunächst eine Eingabe der *Höhe* und *Breite* der Fläche erfolgen muss. Auf die vom Anwender einzustellenden Parameter und Randbedingungen hat man im Menü *Simulation* unter *Parameter* Zugriff. Das Anklicken von *Parameter* öffnet eine Dialogbox, in der man die voreingestellten Werte von *a*, *b*, *tmax*, *breite* und *hoehe* über Scrollbars verändern kann (vgl. Abb. 4). Die Eigenschaften der Ränder der Reaktionsfläche lassen sich in einem Auswahlmenü bestimmen. Durch das Drücken des *OK-Buttons* werden die eingestellten Werte übernommen. Die Simulation startet dann mit einer Zufallsverteilung zum Zeitpunkt *t = 0*. Rechts im Fenster wird hinter „*t* =“ die Anzahl der Rechenschritte, die das Programm durchlaufen hat, angezeigt. Diese ist proportional zur simulierten Zeit.

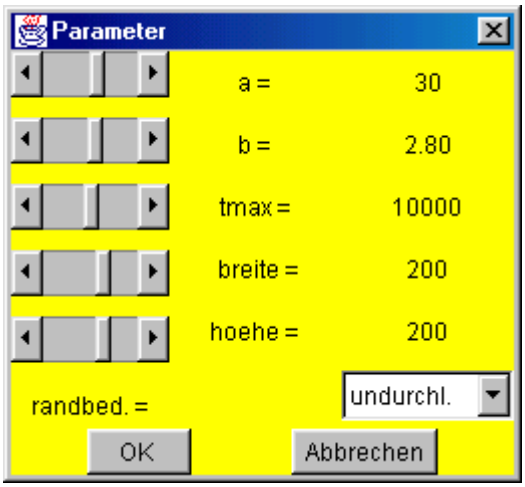

Abb. 4: Dialogbox „Parameter“

Die Simulation kann mit Hilfe der Menüunterpunkte *Start*, *Stop* und *Weiter* im Menü *Simulation* neugestartet, angehalten oder fortgesetzt werden. Bei einem Neustart hat der Anwender die Wahl zwischen einer nahezu homogenen Verteilung mit einer starken Störung in der Mitte (*Zentriert*) und einer Zufallsverteilung (*Zufall*).
Nach 500 Zeitschritten wird der Programmablauf dadurch beschleunigt, dass die Reaktionsfläche nicht mehr nach jedem, sondern nur noch nach jedem 50. Rechenschritt graphisch dargestellt wird.
Erreicht die Anzahl der Zeitschritte (t =) *tmax*, so stoppt das Programm. Es kann durch Erhöhen von *tmax* fortgesetzt, über das Untermenü *Start* neugestartet oder über *Beenden* im Menü *Datei* geschlossen werden.
Je nach Bildschirmauflösung und Monitorformat variiert die Größe des Programmfensters und insbesondere der graphischen Darstellungsfläche. Eine Verringerung der Auflösung auf (640x480) Pixel ist daher häufig sinnvoll. (Dies erfolgt am einfachsten über Anklicken des Desktop mit der *rechten Maustaste* über den Menüpunkt *Eigenschaften* auf der Karteikarte *Einstellungen*.)

2.3 Beispiele von Strukturen mit zugehörigen Parametern

Wie in Abschnitt 1 angedeutet, reagieren die Strukturen der CIMA-Reaktion empfindlich auf Veränderungen der Reaktionsbedingungen. Hier nun einige

[1] siehe www.japi.de

Einstellungen der Parameter, bei denen sich in der Simulation interessante Phänomene beobachten lassen:

- regelmäßige Hexagone (vgl. Abb. 5)
  a = 30, b = 2.80
  Hier empfiehlt sich das Verwenden von (*Start >> Zentriert*)
- Streifenmuster (vgl. Abb. 6)
  a = 15, b = 0.52
- Schwingungen, die zu Strukturen führen (*pinning*)
  a = 30, b = 0.60
- Schwingungen zwischen zwei homogenen Zuständen
  a = 30, b < 0.55

(Eine ausführlichere Anleitung befindet sich auf der CD unter dem Verzeichis: /CD_DPG2003/ AusUndFuerDenUnterricht/HuiskenDD5_5/ Simulationen/); Dateiname: „*Computersimulation von Turingstrukturen.pdf*“)

2.4 Die Arbeitsweise der Simulation

Die Simulation stützt sich auf das *Lengyel-Epstein-Modell* und verwendet die sich daraus ergebenen Gleichungen. Es handelt sich um zwei Differentialgleichungen, die die Konzentrationsänderungen des Aktivators bzw. des Inhibitors mit der *Zeit* beschreiben. Um in einer Simulation *räumliche* Strukturen zu erhalten, ist eine *Diskretisierung des Raumes* oder in diesem Falle der Fläche notwendig. Damit ist eine Unterteilung der Fläche in diskrete Bereiche gemeint. Die entstandenen *Zellen*, denen Konzentrationen des Aktivators und Inhibitors zugeordnet werden, wechselwirken untereinander. Ein solches Modell bezeichnet man als *Zellulärautomat* .

Das Lengyel-Epstein-Modell

Das von István Lengyel und Irving R. Epstein [2] 1991 veröffentlichte Modell zur Beschreibung der Vorgänge in der CIMA-Reaktion lässt sich wie oben beschrieben in guter Näherung auf ein zwei Variablen Modell reduzieren:

$$\rightarrow \quad X \qquad (1')$$

$$X \rightarrow Y \qquad (2')$$

$$4X + Y \rightarrow \qquad (3')$$

mit $X = I^-$ (*Aktivator*) und $Y = ClO_2^-$ *(Inhibitor*)

Nach Jensen et al. [10] ergeben sich aus diesen drei Reaktionsgleichungen nach einigen Umformungen und Substitutionen zwei Differentialgleichungen:

$$\frac{\partial u}{\partial t} = a - u - \frac{4uv}{1+u^2} + \nabla^2 u \qquad (1)$$

und

$$\frac{\partial v}{\partial t} = \delta\left(b\left(u - \frac{uv}{1+u^2}\right) + c\nabla^2 v\right) \qquad (2)$$

In diesen Gleichungen steht *u* für die Konzentration des Aktivators (*u = [X] = [I⁻])* und *v* für die Konzentration des Inhibitors (*v = [Y] = [ClO_2^-]). a* und *b* sind Parameter, die von den Konzentrationsverhältnissen der zugeführten Lösungen abhängen.

$$a \propto [MA]/[ClO_2] \qquad (3)$$

$$b \propto [I_2]/[ClO_2] \qquad (4)$$

c ist das Verhältnis der Diffusionskonstanten D_v/D_u und δ ist ein von der Konzentration abhängiger Skalierungsparameter.

Beide Differentialgleichungen (1 u. 2) haben die selbe Struktur:

$$\frac{\partial \vec{c}}{\partial t} = \vec{f}(\vec{c}) + \nabla^2 \vec{c} \qquad (6)$$

$\vec{c}$ ist in dieser Gleichung ein raum- und zeitabhängiger Vektor und $\vec{f}$ eine Vektorfunktion. Es handelt sich um eine Reaktions-Diffusions-Gleichung, wobei $\vec{f}(\vec{c})$ der Reaktions- und $\nabla^2\vec{c}$ der Diffusionsterm ist.

Da der Computer nur die Grundrechenarten beherrscht, werden in diesem Kapitel numerische Lösungsmöglichkeiten dieser Reaktions-Diffusions-Gleichung vorgestellt.

Zeitliche Integration

Zunächst betrachten wir nur die folgende Gleichung:

$$\frac{\partial \vec{c}}{\partial t} = \vec{f}(\vec{c}) \,. \qquad (7)$$

Zur Annäherung an die zeitliche Ableitung benutzen wir den Differenzenquotienten:

$$\frac{\partial \vec{c}}{\partial t} = \vec{f}(\vec{c}(t)) \approx \frac{\vec{c}(t+\Delta t) - \vec{c}(t)}{\Delta t}$$

$$\Rightarrow \vec{c}(t+\Delta t) \approx \vec{c}(t) + \Delta t \cdot \vec{f}(\vec{c}(t)) \qquad (8)$$

Daraus ergibt sich eine Approximation von $\vec{c}$ durch numerische zeitliche Integration. Anstelle infinitesimal kleiner Zeitschritte verwenden wir Zeitschritte Δt endlicher Größe.

Graphisch lässt sich diese Vorgehensweise damit beschreiben, dass man den Graph der Funktion $\vec{c}(t)$ an der Stelle *t* durch eine Tangente annähert und annimmt, dass ihre Steigung der Steigung des Funktionsgraphen im Intervall von *t* bis $t+\Delta t$ entspricht. Dieses Verfahren wird dann jeweils im Abstand Δt wiederholt.

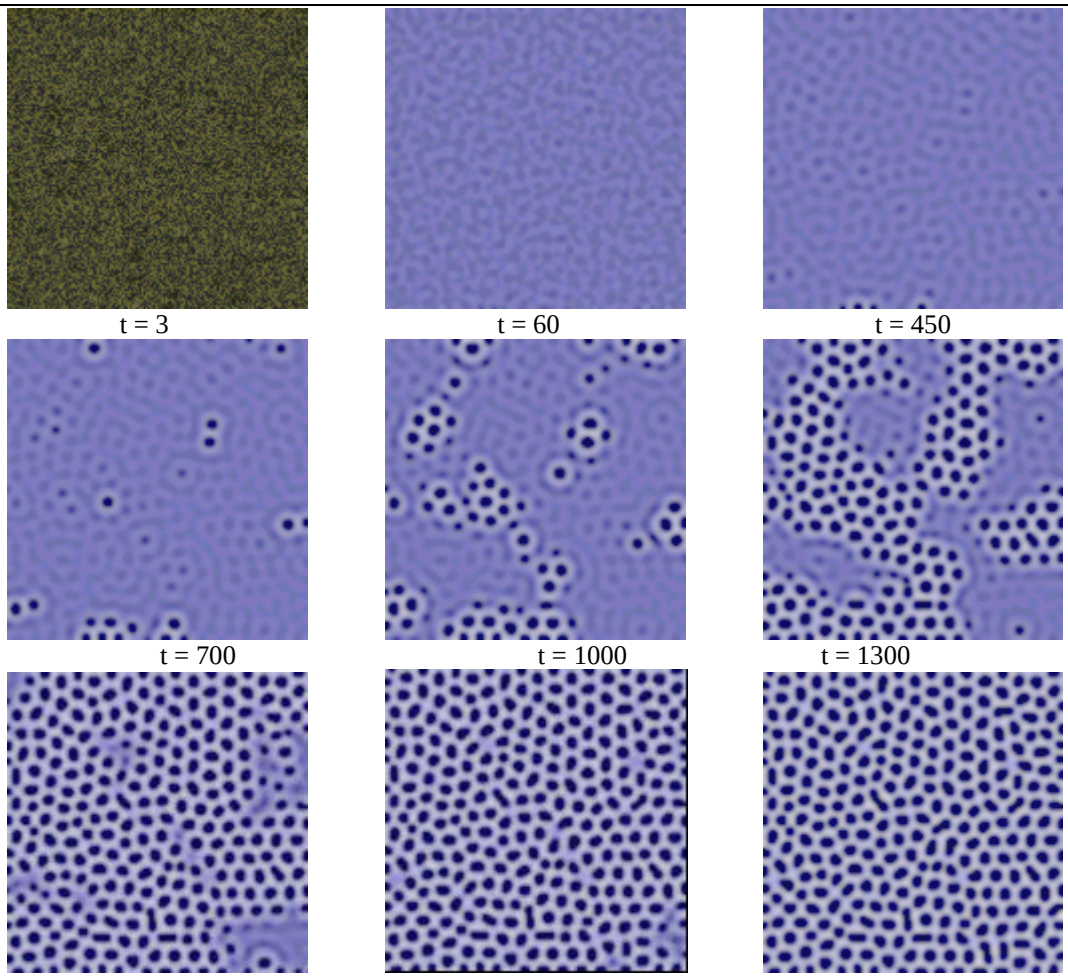

Abb. 5: Entwicklung von Hexagonen, ausgehend von einer Zufallsverteilung. Parameter: a = 30, b = 2,8.

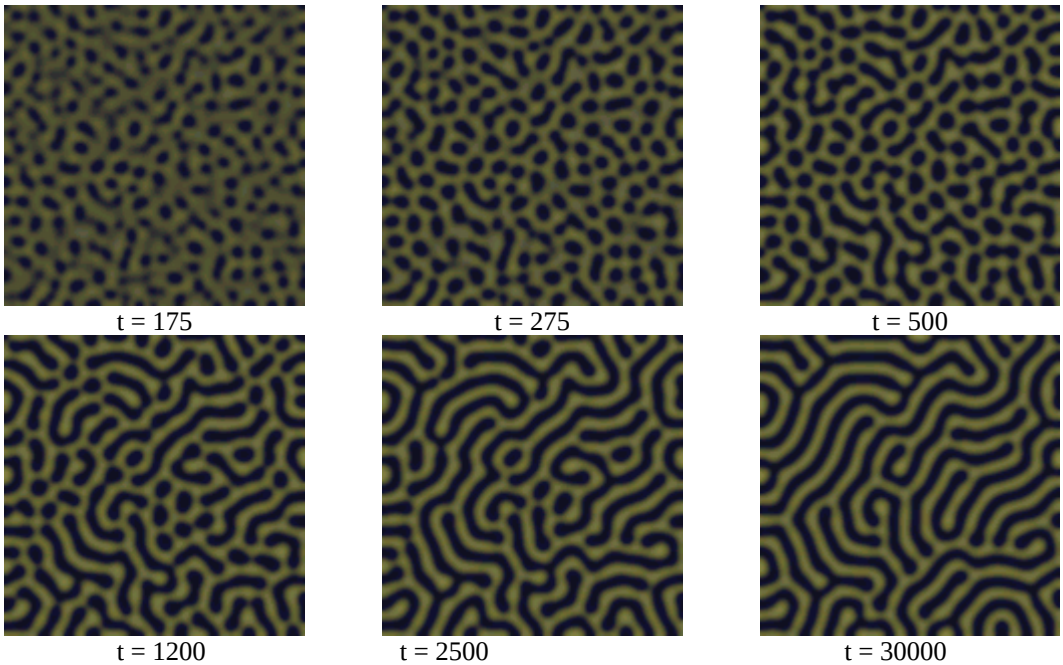

Abb. 6: Entwicklung von Streifenmustern. Parameter: a = 15, b = 0.52

Diskretisierung der Fläche

Die zeitliche Integration betrifft den Reaktionsterm. Bei der näherungsweisen Erfassung des Diffusionsterms muss die durch den *Laplace-Operator* ∇^2 beschriebene zweifache Ableitung nach dem Ort diskretisiert werden.

Die Grundidee zur Lösung solcher Probleme liegt in der Unterteilung eines kontinuierlichen Raumes (hier: Fläche) in eine endliche Anzahl von Zellen. Im eindimensionalen Fall teilt man die Strecke der Länge *l* wie in Abb. 7 in *n* gleichlange Elemente ein.

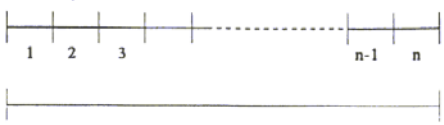

Abb. 7: Diskretisierung eines eindimensionalen Raumes (aus [11])

Die Größe einer Zelle ist dann $\Delta x = \frac{l}{n}$. Wir können jetzt eine Näherung für die durch $\nabla^2 \vec{c}$ beschriebene Diffusion für eine i-te Zelle angeben, die nicht am Rand liegt (1 < i < n):

$$\nabla^2 \vec{c} \approx \frac{1}{\Delta x^2}((\vec{c}_{i-1} - \vec{c}_i) + (\vec{c}_{i+1} - \vec{c}_i))$$
$$\approx \frac{1}{\Delta x^2}((\vec{c}_{i-1} + \vec{c}_{i+1} - 2\vec{c}_i)) \quad (9)$$

Eine rechteckige Fläche wird am einfachsten in Form von quadratischen Zellen (Abb. 8) dargestellt, d.h.: $\Delta x_k = \Delta x_l = \Delta x$.

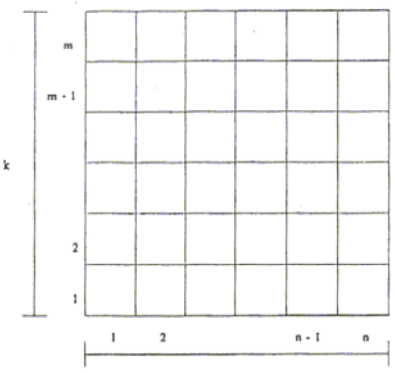

Abb. 8 Diskretisierung eines zweidimensionalen Raumes (aus [11])

Für den direkten Diffusionsaustausch mit einer Zelle (i, j), die nicht am Rand der Fläche liegt (1 < i < n ; 1 < j < m) kommen zunächst die Zellen mit einer gemeinsamen Kante in Betracht. Daraus folgt für unsere Näherung:

$$\nabla^2 \vec{c} \approx \frac{1}{\Delta x^2}((\vec{c}_{i-1,j} + \vec{c}_{i+1,j} + \vec{c}_{i,j-1} + \vec{c}_{i,j+1} - 4\vec{c}_{i,j})) \quad (10)$$

Diese vier Zellen bezeichnet man als *Von-Neumann-Nachbarschaft*. Sie unterscheidet sich von der *Moore-Nachbarschaft*, die aus allen acht Zellen besteht, die eine Zelle umgeben:

$$\nabla^2 \vec{c} \approx \frac{1}{\Delta x^2}(\vec{c}_{i-1,j} + \vec{c}_{i+1,j} + \vec{c}_{i,j-1} + \vec{c}_{i,j+1} + \vec{c}_{i-1,j+1} + \vec{c}_{i+1,j-1} + \vec{c}_{i+1,j+1} - 8\vec{c}_{i,j}) \quad (11)$$

Aufgrund des größeren Abstandes der Zellen, die mit der betrachteten Zelle i,j nur eine Ecke teilen, versehen wir deren Beitrag zur Diffusion mit dem Faktor $1/\sqrt{2}$.

$$\nabla^2 \vec{c} \approx \frac{1}{\Delta x^2}((\vec{c}_{i-1,j} + \vec{c}_{i+1,j} + \vec{c}_{i,j-1} + \vec{c}_{i,j+1} - 4\vec{c}_{i,j}) + 1/\sqrt{2}(\vec{c}_{i-1,j-1} + \vec{c}_{i-1,j+1} + \vec{c}_{i+1,j-1} + \vec{c}_{i+1,j+1} - 4\vec{c}_{i,j})) \quad (12)$$

Bedingungen der Ränder

In der Simulation hat man die Wahl zwischen drei Varianten:

- *undurchlässig*
  Diese Version wird auch als *no flux boundary* oder *Neumann Rand* bezeichnet. Sie zeichnet sich dadurch aus, dass durch die Ränder keine Diffusion möglich ist und die Randzellen somit nur fünf, die Eckzellen drei Nachbarn haben.
- *konstant*
  Auch *Fixed boundary* oder *Dirichlet Rand* genannt. Hier haben die Randzellen feste Werte, die sich im Programmverlauf nicht verändern. Die drei oder fünf Nachbarzellen werden zwar von den Randzellen beeinflusst, beeinflussen aber nicht die Randzellen.
- *periodisch*
  In diesem Fall stehen die Randzellen mit ihrem Pendant am gegenüberliegenden Rand in Kontakt. D.h. zur Moore-Nachbarschaft einer Zelle des rechten Randes zählen auch die drei Zellen am linken Rand gegenüber. Selbiges gilt für den oberen und unteren Rand. Anschaulich bedeutet dieses Verfahren, dass die rechteckige Fläche zunächst zylinderförmig aufgerollt gegenüberliegende Seiten miteinander verbunden werden. Anschließend fügt man die so entstehenden Öffnungen zusammen, so dass ein Torus oder anschaulicher ein Fahrradschlauch entsteht.

Raum-zeitliche Entwicklung des Gesamtsystems

Nachdem wir nun die Reaktionsfläche in $n \times m$ Zellen eingeteilt haben, besteht unser System aus $n \times m$ gekoppelten, gewöhnlichen Differentialgleichungen. Diese lassen sich durch die *Explizite-Euler-Methode* auf numerischen Wege näherungsweise lösen. Durch das Verbinden des Zellulärautomaten mit der Methode der zeitlichen Integration, können wir die raumzeitliche Entwicklung des Sys-

tems bei Kenntnis der Anfangs- und Randbedingungen berechnen.
Betrachten wir zunächst nur den eindimensionalen Fall unter Verwendung folgender Notation:

$$\vec{c}(x_i, t_N) = \vec{c}_i^{\,N} \tag{13}$$

mit $t_N = N \cdot \Delta t + t_0,\ N \in \mathrm{N}$

Wie oben zeigt der untere Index die Zelle an, im oberen Index steht die Anzahl der Zeitschritte, die seit dem Zeitpunkt t_0 vergangen sind.
Analog zu (6)-(8) erhalten wir für die Reaktion und Diffusion beinhaltende Differentialgleichung einer Zelle (6):

$$\frac{\vec{c}_i^{\,N+1} - \vec{c}_i^{\,N}}{\Delta t} = \vec{f}(\vec{c}_i^{\,N}) + \frac{D}{\Delta x^2}(\vec{c}_{i-1}^{\,N} + \vec{c}_{i+1}^{\,N} - 2\vec{c}_i^{\,N}),$$

$$(1 < i < n) \tag{14}$$

Die Differentialgleichungen für *Aktivator* und *Inhibitor* des Reaktions-Diffusions-Systems sind:

$$\frac{\partial u}{\partial t} = f(u,v) + D_u \nabla^2 u$$

$$\frac{\partial v}{\partial t} = g(u,v) + D_v \nabla^2 v \tag{15}$$

Aus (15) und (1), (2) folgt für das Lengyel-Epstein-Modell

$$\vec{c} = (u, v)$$

$$\vec{f} = \left(a - u - \frac{4uv}{1+u^2}, \delta b\left(u - \frac{uv}{1+u^2}\right)\right)$$

$$D = (1, \delta c) \tag{16}$$

Setzt man (16) in (14) ein, so erhält man für 1D und $1 < i < n$:

$$\frac{u_i^{N+1} - u_i^N}{\Delta t} = f(u_i^N, v_i^N) + \frac{D_u}{\Delta x^2}(u_{i-1}^N + u_{i+1}^N - 2u_i^N)$$

$$\frac{v_i^{N+1} - v_i^N}{\Delta t} = g(u_i^N, v_i^N) + \frac{D_v}{\Delta x^2}(v_{i-1}^N + v_{i+1}^N - 2v_i^N) \tag{17}$$

Wenn wir uns erinnern, dass *u* die Konzentration des Aktivators und *v* die des Inhibitors ist, können wir bei Kenntnis der Konzentrationen zum Zeitpunkt t_N die Konzentrationen zum Zeitpunkt t_{N+1} berechnen:

$$u_i^{N+1} = \Delta t \cdot f(u_i^N, v_i^N) + \frac{D_u \Delta t}{\Delta x^2}(u_{i-1}^N + u_{i+1}^N - 2u_i^N) + u_i^N$$

$$v_i^{N+1} = \Delta t \cdot g(u_i^N, v_i^N) + \frac{D_v \Delta t}{\Delta x^2}(v_{i-1}^N + v_{i+1}^N - 2v_i^N) + v_i^N \tag{18}$$

Für 2D erhalten wir ohne Berücksichtigung der Randzellen:

$$u_{i,j}^{N+1} = \Delta t \cdot f(u_{i,j}^N, v_{i,j}^N) +$$
$$\frac{D_u \Delta t}{\Delta x^2}((u_{i-1,j}^N + u_{i,j-1}^N + u_{i,j+1}^N + u_{i+1,j}^N - 4u_i^N)$$
$$+ \frac{1}{\sqrt{2}}(u_{i-1,j-1}^N + u_{i-1,j+1}^N + u_{i+1,j-1}^N + u_{i+1,j+1}^N - 4u_{i,j}^N)) + u_{i,j}^N$$

$$v_{i,j}^{N+1} = \Delta t \cdot g(u_{i,j}^N, v_{i,j}^N) +$$
$$\frac{D_v \Delta t}{\Delta x^2}((v_{i-1,j}^N + v_{i,j-1}^N + v_{i,j+1}^N + v_{i+1,j}^N - 4v_i^N)$$
$$+ \frac{1}{\sqrt{2}}(v_{i-1,j-1}^N + v_{i-1,j+1}^N + v_{i+1,j-1}^N$$
$$+ v_{i+1,j+1}^N - 4v_{i,j}^N)) + v_{i,j}^N \tag{19}$$

Dies kann jetzt in ein Computerprogramm implementiert werden (vgl. [11]).
Da es aus programmiertechnischen Gründen nicht möglich war, die Werte für *u* und *v* in einer Matrix mit *m* Zeilen und *n* Spalten abzulegen[2], verwendet das Programm einen Vektor mit $k = m \times n$ Elementen. Die Zeilen der Matrix sind in diesem Vektor hintereinandergereiht, so dass sich das Element (*i,j+1*) im Vektor an der Stelle $k + n$ befindet.
Es gelten folgende Ersetzungen:

$dr \equiv v_{i,j}^{N+1} - v_{i,j}^N$	$dx \equiv \Delta x$
$b[i] \equiv u_k$	$breite \equiv n$
$db \equiv u_{i,j}^{N+1} - u_{i,j}^N$	$hoehe \equiv m$
$r[i] \equiv v_k$	$0{,}7071 \equiv \sqrt{2}$

Die Lösung der diskretisierten Reaktions-Diffusions-Gleichungen sieht im Pascal-Quelltext folgendermaßen aus:

```
function db;
(...)
du := a-b[i]-(4*b[i]*r[i])/(1+b[i]*b[i])
    +D1/(dx*dx)*((b[i+breite]-b[i])
    +(b[i-breite]-b[i])
    +(b[i+1]-b[i])+(b[i-1]-b[i])
    +0.7071*(b[i+breite+1]-b[i])
    +0.7071*(b[i-breite-1]-b[i])
    +0.7071*(b[i+breite-1]-b[i])
    +0.7071*(b[i-breite+1]-b[i]));
  end;
 end;
 db := du*dt;
end;
function dr;
(...)
dv := delta*(bi*(b[i]-
    (r[i]*b[i])/(1+b[i]*b[i]))
    +D2/(dx*dx)*((r[i+breite]-r[i])
    +(r[i-breite]-r[i])+(r[i+1]-r[i])+(r[i-
    1]-r[i])
    +0.7071*(r[i+breite+1]-r[i])
    +0.7071*(r[i-breite-1]-r[i])
    +0.7071*(r[i+breite-1]-r[i])
    +0.7071*(r[i-breite+1]-r[i])));
  end;
 end;
 dr := dv*dt;
end;
(...)
procedure bestehorn;
(...)
```

[2] japi-Funktionen zur graphischen Darstellung verwenden keine Matrizen sondern Vektoren (s. *Graphische Darstellung*)

```
begin
     for i:=1 to breite*hoehe do begin
         r[i] := r[i]+dr(i);
         b[i] := b[i]+db(i);
(...)
```

Die Berechnung von *db* bzw. von *dr* erfolgt in `function db` bzw. `function dr`. Die Aktivator- bzw. Inhibitorkonzentrationen `b[i]` und `r[i]` zum Zeitpunkt t_{N+1} werden erst weiter unten in der `procedure bestehorn` errechnet. Das hat den Grund, dass die Konzentrationsänderungen für jede der vier Seiten und der vier Eckzellen durch eine andere Formel beschrieben wird. Zusätzliche Fallunterscheidungen kommen durch die drei zur Auswahl stehenden Bedingungen für die Ränder zustande. Die Addition von `r[i]` und `dr[i]` erfolgt in einer FOR...TO...DO-Schleife[3]. Diese erhöht, beginnend bei 1, den Zellenindex `i` bei jedem Durchlauf um 1, bis die letzte Zelle erreicht ist und `i = breite*hoehe` beträgt. Auf diese Weise wird nacheinander jeder Zelle durch das Zeichen `:=` ein neuer Wert `r[i]` zugewiesen (`r[i] := r[i] + dr[i]`). Das gleiche gilt für `b[i]`.

Parameter

In den Gleichungen sind vier Parameter enthalten, die zum Teil durch einen *‚trial and error'* - Prozess bestimmt werden mussten. Dies ist ein schwieriges Unterfangen, da Turingstrukturen nur in einem relativ kleinen Gebiet auftreten (vgl. Abschnitt 1). In der Literatur finden sich bei Jensen und Pannbacker [11] folgende Werte, bei denen Turingstrukturen entstehen:

a = 30.0	c = 1.5
b = 2.7	δ = 8.0

Auch über die Größe von Δx und Δt muss man sich Gedanken machen. Während die Kantenlänge Δx der Zellen, die später auf dem Bildschirm durch ein Pixel dargestellt werden, Einfluss auf die Ausmaße der dargestellten Struktur hat, hängen Simulationsgeschwindigkeit und –genauigkeit von der Größe des Zeitschrittes Δt ab. Je kleiner der Zeitschritt, desto höher die Genauigkeit der zeitlichen Integration. Je größer der Zeitschritt, desto höher die Simulationsgeschwindigkeit. Ein guter Kompromiss zwischen Genauigkeit und Geschwindigkeit besteht bei Δt = 1/60. Für Δx habe ich den Wert 1 ausgewählt.

Start der Simulation

Das Programm besteht im Kern also aus einer *Routine* zur Berechnung der raum-zeitlichen Entwicklung des Gesamtsystems. Diese benötigt einen Satz von *Parametern* der im vorigen Abschnitt gegeben wurde, Kenntnis über die Behandlung der *Ränder* und über die *Anfangsbedingungen*.

[3] Informationen über Turbo Pascal und seine Befehle finden sich in **[12]**

Letztere werden standardmäßig über die `procedure zufallsverteilung` erzeugt.

```
procedure zufallsverteilung;
  begin
   while (do_work = true) do begin
     for i:=0 to breite*hoehe do begin
        r[i]  := (j_random mod 256)/50;
        b[i]  := (j_random mod 256)/50;
```

Die *procedure* startet mit einer WHILE...DO-Schleife. Diese wird so oft durchlaufen, wie die Bedingung `(do_work = true)` zutrifft. In einer FOR...TO...DO-Schleife wird wie oben in der `procedure bestehorn` jeder Zelle ein zufälliger[4] Wert `r[i]` und `b[i]` zugeordnet. Dies geschieht unter Verwendung der japi-Funktion `j_random` die mit dem Zusatz `mod 256` eine ganze Zahl zwischen 0 und 256 ausgibt. Um im Wertebereich der Funktionen `r` und `b` zu bleiben wird noch durch 50 dividiert.

Graphische Darstellung

Wie schon erwähnt wird jede Zelle des Zellulärautomaten auf dem Bildschirm durch einen Pixel repräsentiert. Alle auf dem Monitor darstellbaren Farben setzen sich aus den drei Grundfarben rot, grün und blau (r,g,b) zusammen. Jedem Pixel der Darstellungsfläche können drei Werte für die Intensitäten der Grundfarben zwischen 0 und 256 zugeordnet werden, wobei 256 der vollen Helligkeit entspricht. Der Farbe schwarz entsprich folglich (r = 0, g = 0, b = 0) und weiß (r = 256, g = 256, b = 256). Der Aktivator Iodid (I^-) wird in der Simulation blau dargestellt, da er im Realexperiment einen tiefblauen Iod-Stärke-Komplex bildet. Das farblose Chlorit (ClO_2^-) wird auf dem Bildschirm gelb sichtbar gemacht, indem rot und grün zu gleichen Teilen gemischt werden.

Um eine ausreichende Helligkeit zu erhalten werden die `r`- und `b`-Werte mit 20 multipliziert. Die Funktion `trunc` liefert den ganzzahligen Anteil dieses Produktes, der in dem Vektor `rr` bzw. `bb` abgespeichert wird.

```
rr[i] := trunc(20*r[i]);
bb[i] := trunc(20*b[i]);
g[i] := rr[i];
```

Der japi-Befehl zur Darstellung eines Bildes ist `j_drawimagesource`. Ihm werden die Bilddaten in Form der Vektoren `rr,g` und `bb` übergeben. `j_sync` und `j_sleep` verursachen eine Pause von etwa einer Sekunde. Nachdem die *Boolean-*

[4] Da ein deterministisches System wie der Computer keinen echten Zufall erzeugen kann, handelt es sich hier um einen pseudozufälligen Wert.

Variable[5] auf den Wert `false` gesetzt wurde, kehrt das Programm in den Hauptteil zurück.

```
j_drawimagesource(canvas,0,0,breite,hoe
he,rr[1],g[1],bb[1]);
j_sync;
j_sleep(1000);
do_work :=false;
```

Die Simulation kann auch über den Menüpunkt *Zentriert* im Untermenü *Start* gestartet werden. Der Aufbau der zugehörigen `procedure mitte-verteilung` ist dem der `procedure zu-fallsverteilung` ähnlich. Der Unterschied besteht darin, dass der Zelle in der Mitte der Reaktionsfläche ein hoher r-Wert und b = 0 zugeordnet wird. Den übrigen Zellen werden zufällige r-Werte in einem schmalen Intervall oberhalb von null, und b-Werte in einem Intervall gleicher Breite auf hohem Niveau zugeteilt.
Die wichtigsten Prinzipien der Arbeitsweise des Programms wurden somit kurz beschrieben. Nach jedem Zeitschritt erfolgt beim Programmablauf in der `procedure bestehorn` durch den Befehl `j_drawimagesource` die Darstellung der gesamten Reaktionsfläche. Die Routinen zur Eingabe der Parameter durch den Anwender oder zum Aufbau des Fensters erscheinen mir in diesem Kontext weniger interessant, der gesamte Quelltext ist jedoch im Anhang nachzulesen.

3 Literatur

[1] A.M. Turing: The chemical basis of morphogenesis, Philos.Trans. R. Soc. London Ser. B. 237, 37 (1952)

[2] I. Lengyel, R. Epstein: Modeling of Turing Structures, Science 251, 650-652 (1991)

[3] Bestehorn, Michael: Strukturbildung durch Selbstorganisation in Flüssigkeiten und in chemischen Systemen (1995)

[4] J. Falbe, M. Regitz: CD Römpp 9. erweiterte und überarbeitete Auflage des Römpp Chemielexikons auf CD-ROM, Thieme Veralag, Stuttgart 1995

[5] P. Ball: The Self-Made Tapestry, Pattern Formation in Nature, Oxford UP, Oxford 1999

[6] V.Castets, e. Dulos, J.Boissonade, and P.De Kepper: Experimental Evidence of a Sustained Turing-Type Nonequilibrium Chemical Pattern, Phys. Rev. Let., 64, 24, 2953-2956 (1990)

[7] I.R. Epstein, K. Showalter: Nonlinerar Chemical Dynamics: Oscillations, Patterns, and Chaos, J. Phys. Chem. 1996, 100, 13132-13147

[8] P. Gómez-Romero: Hexagons in a packed world (1999), URL, http://www.granavenida.com/superciencia/stshexag.html (21.04.2002)

[9] A. Hodges: Alan Turing: The Enigma, Walker and Company, New York 1983

[10] O. Jensen, V.O. Pannbacker, E. Mosekilde, G. Dewel, P. Borckmans: Localized structures and front propagation in the Lengyel-Epstein model, Physical Reviw E Vol 50, Nr. 2, 736-749 (1994)

[11] O. Jensen, V.O. Pannbacker: Pattern formation in reaction-diffusion systems, Kopenhagen, The Technical University of Denmark, Physics Department, Diss., 1993

[12] W. Kassera, V. Kassera: Turbo Pascal 7.0, Das Kompendium, Markt & Technik Verlag, München 1993

[13] P.J. Heaney, A.M. Davis: Observation and Origin of Self-Organized Textures in Agates, Science 269, 1562 (1995)

Die beschriebene Software und die entsprechenden Dokumentationen befinden auf der Tagungs-CD im Verzeichnis:
/CD_DPG2003/AusUndFuerDenUnterricht/
HuiskenDD5_5/Simulationen/

[5] Boolean ist ein Datentyp in Turbo Pascal, der die Werte *true* und *false* annehmen kann.