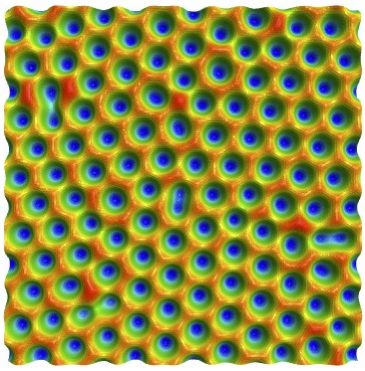
# HANA Numerik SEP Projekt

Simon Stingelin (stiw@zhaw.ch)

28. Oktober 2025



**Gruppe:** Victor Baldini, Lorena Natale, Linda Riesen

**Thema:** Komplexe Muster in einem einfachen System

**Lerninhalt:**

- Berechnen der schwachen Gleichung.
- Anwenden der Methode der finiten Elemente auf ein Problem aus der Naturwissenschaft.
- Lösen eines gekoppelten zeitabhängigen nichtlinearen Randwert Problems.
- Periodische Randwerte

**Abgabe:**

- Die Abgabe erfolgt als Kurzbericht in Form eines **PDF**-Dokuments über die moodle Abgabe bis zum in der moodle Abgabe definierten spätesten Abgabezeitpunkt.
- Die numerischen Resultate sind im Bericht dokumentiert und können mit lauffähigen Skripts nachvollzogen werden.

**Bewertung:**

| Note 1 | Note 3 | Note 4 | Note 4.5 | Note 5 | Note 5.5 | Note 6 |
|---|---|---|---|---|---|---|
| **Keine Abgabe** | Nicht bewertbar: die Aufgaben wurden **nicht ernsthaft** bearbeitet. | Die Aufgaben wurden **knapp** bearbeitet und sind nur im Ansatz dokumentiert. | Die Aufgaben wurden **halbwegs** bearbeitet und sind sehr knapp Ansatz dokumentiert. | Die Aufgaben wurden **bearbeitet** und dokumentiert. | Die Aufgaben wurden **detailliert** bearbeitet und dokumentiert. | Die Aufgaben wurden detailliert bearbeitet und dokumentiert. Es wurden noch **zusätzliche** Aspekte bearbeitet. |

# 1 Aufgabestellung

Muster sind überall in der Natur. Beispiele umfassen Flecken auf Schmetterlingen, Streifen auf Zebras, Dreiecke auf Muscheln, Wellen auf Sanddünen, Wolken am Himmel, Wellen im Ozean, chemische Wellen, und Fingerabdrücke. Muster können regelmässig oder quasi-regelmässig oder zufällig sein; sie können stationär, quasi-stationär oder sich schnell bewegen. Woher kommen diese Strukturen? In vielen Fällen entstehen sie als Lösungen von Reaktions-Diffusions-Gleichungen, das heisst, Systeme von PDEs (partielle Differentialgleichungen), die lineare Diffusion mit nichtlinearen Wechselwirkungen kombinieren.

Abbildung 1: Zebras

Die Gray-Scott-Gleichungen wurden ursprünglich 1983 von Gray und Scott formuliert. Wir werden ihre ursprüngliche chemische Motivation nicht diskutieren:

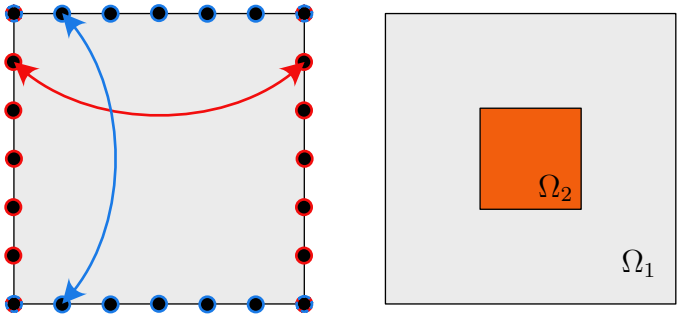
$$\partial_t u = \epsilon_1 \Delta u - u\,v^2 + F\,(1-u) \tag{1a}$$
$$\partial_t v = \epsilon_2 \Delta v + u\,v^2 - (k+F)\,v \tag{1b}$$

Dies ist ein gekoppeltes Paar von Gleichungen in zwei unabhängigen Variablen $u$ und $v$. Jede Variable diffundiert unabhängig. Wir setzen die Diffusionskonstanten auf $\epsilon_1 = 2{\cdot}10^{-5}$ und $\epsilon_2 = 10^{-5}$ fest. Jede wächst oder zerfällt ebenfalls unabhängig entsprechend dem linearen Term $F\,(1-u)$ oder $(k+F)\,v$. Die Kopplung zwischen den beiden Gleichungen ist durch den dritten Term $\pm uv^2$ gegeben, der nichtlinear Energie von $u$ zu $v$ überträgt. Die Parameter $k$ und $F$ sind beliebige positive Zahlen, die wir anpassen werden. In der Abb. 2 sind ein paar Beispiele für verschiedene Parametersätze zu sehen.

(a) $F = 0.03, k = 0.06$ (b) $F = 0.05, k = 0.065$ (c) $F = 0.06, k = 0.0625$

Abbildung 2: Muster für verschiedene Parametersätze $(F, k)$.

Wir betrachten das PDE System auf einem *periodischen* Gebiet, welches durch ein Quadrat mit dem Ursprung in der Mitte und einer Breite von 1.5 gegeben ist. Um eine periodische Lösung berechnen zu können, muss auch das Mesh entsprechend periodisch generiert werden. In der Abb. 3a ist die Periodizität zu sehen. In `ngsolve` können wir dies für die Geometrie gemäss Code 5 definieren.




(a) Periodisches Gebiet (b) Startwert

Abbildung 3: Periodisches Gebiet und Startwert

Die Muster entstehen, wenn wir initial die Startfunktionen

$$u_0(\mathbf{x}) = \begin{cases} 1 & \text{für } \mathbf{x} \in \Omega_1 \\ 0.5 & \text{für } \mathbf{x} \in \Omega_2 \end{cases}$$
$$v_0(\mathbf{x}) = \begin{cases} 0 & \text{für } \mathbf{x} \in \Omega_1 \\ 0.25 & \text{für } \mathbf{x} \in \Omega_2 \end{cases} \tag{2}$$

benutzen. Um die Symmetrie zu brechen, wird eine kleine normal verteilte Zufallszahl addiert:

```
l_init = l_dom/n_dom*20
gfuold.Set(IfPos((l_init/2)**2-x*x,
                 IfPos((l_init/2)**2-y*y,0.5,1),1))
gfvold.Set(IfPos((l_init/2)**2-x*x,
                 IfPos((l_init/2)**2-y*y,0.25,0),0))
gfxold.vec[:] += 0.01*np.random.normal(size=X.ndof)
gfx.vec.data = gfxold.vec
```

Listing 1: Startwert $u_0, v_0$

Wir betrachten daher die Muster, welche sich aus dieser initialen Konzentrationsverteilung der beiden Stoffe $u, v$ durch die Reaktions-Diffusions-Gleichung entstehen.

## Input zur Numerik

Für die Diskretisierung benutzen wir einen $H^1$ finite Elemente Raum mit Ordnung 3, jeweilen für $u$ und $v$, wobei wir auch hier die Periodizität berücksichtigen müssen.

```
order = 3
V = Periodic(H1(mesh,order=order))
X = V*V
(u,v) = X.TrialFunction()
(w,q) = X.TestFunction()
```

Listing 2: FEM space definition

Die GridFunction setzt sich nun aus diesen zwei Funktionen zusammen:

```
# Loesung zum neuen Zeitschritt
gfx = GridFunction(X)
gfu, gfv = gfx.components

# Loesung zum alten Zeitschritt
gfxold = GridFunction(X)
gfuold, gfvold = gfxold.components
```

Listing 3: Gridfunction

Die parabolische Gleichung (1) führt im endlich dimensionalen FE-Raum auf das Gleichungssystem

$$M \cdot (\dot{u}, \dot{v}) + A \cdot (u, v) = f(u, v).$$

Benutzt man den Differenzenquotient anstelle der Ableitung nach der Zeit $t$, folgt

$$M \cdot \frac{(u_{n+1}, v_{n+1}) - (u_n, v_n)}{dt} + A \cdot (u_{n+1}, v_{n+1}) = f(u_n, v_n). \tag{3}$$

mit

$$(\delta u, \delta v) = (u_{n+1} - u_n, v_{n+1} - v_n)$$

folgt für 3 das inkrementelle implizite Euler Verfahren

$$\begin{aligned} \underbrace{(M + dtA)}_{=:M^*} \cdot (\delta u, \delta v) &= -dtA \cdot (u_n, v_n) + dtf(u_n, v_n), \\ (u_{n+1}, v_{n+1}) &= (u_n, v_n) + (\delta u, \delta v). \end{aligned} \tag{4}$$

Eine detaillierte Diskussion finden Sie im Skript, Kapitel 12.3.2. Das Verfahren in der Anwendung ist ein sogenanntes IMEX-Verfahren, der Differentialoperator wird implizit und die Nichtlinearität explizit berechnet.

# 2 Aufgaben

## Aufgabe 1

1. Wie lautet die schwache Formulierung des nichtlinearen Gleichungssystems (1)?
2. Aus welchen Termen besteht die Funktion $f(u_n, u_v)$ in (4)?

**Vorgehen:** Testen Sie jede Gleichung mit eigenen Testfunktionen $w, q$ (vlg. Listing 2).

Für die Implementierung können Sie den Code der Listings 6, 7 und 8 benutzen. Verifizieren Sie Ihre schwachen Gleichungen anhand der Listings. Weisen Sie die einzelnen Terme den Definitionen zu.

## Aufgabe 2

Leiten Sie das numerische Schema (4) her und definieren Sie die Bilinearformen $M, A$ sowie die Linearform $f$.

## Aufgabe 3

1. Implementieren Sie das zeitabhängige Verfahren (4) (vgl. auch HANA Skript).

```
dt = 10.
mstar = a.mat.CreateMatrix()
mstar.AsVector().data = m.mat.AsVector() + dt*a.mat.AsVector()
mstarinv = mstar.Inverse(inverse='sparsecholesky')
```

Listing 4: Definition, Berechnung von $M^*$ aus (4).

2. Verschiedene Muster erhalten Sie für unterschiedliche Parametersätze $(F, k)$ in der Abb. 2 sind verschiedene Beispiele dargestellt.

   Berechnen Sie Lösungen für unterschiedliche Parametersätze, in dem Sie 2-4'000 Zeitschritte rechnen. Sie können die Lösung beobachten, in dem Sie eine Scene erstellen und diese während der Rechnung aktualisieren (vgl. Listing 9).

3. Generieren Sie mit Hilfe der Lösung $u, v$ Bilder für die Dokumentation (vgl. Listing 10).

## Aufgabe 4

1. Zeigen Sie, dass $u \equiv 1, v \equiv 0$ eine Lösung ist.

2. Eine zweite (instabile) Lösung gibt es für bestimmte Parametersätze. Zeigen Sie, dass

$$u \equiv \frac{F}{k + 2F}, \quad \text{und} \quad v \equiv \sqrt{k + F}$$

   eine Lösung der Gleichung (1a) ist. Für welche Parametersätze $(F, k)$ erhalten wir auch für die Gleichung (1b) eine Lösung?

   Interessant sind daher Parametersätze, welche aus dem Bereich der instabilen Lösungen kommen (vgl. Abb. 4).

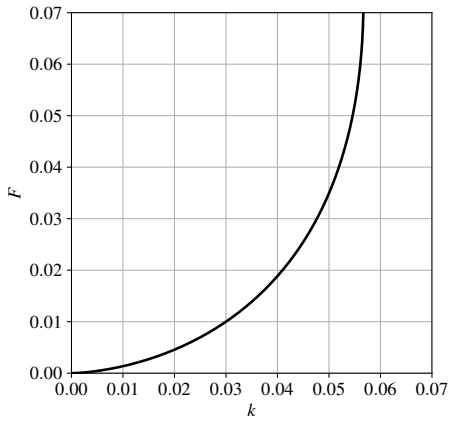



Abbildung 4: Parameter Kombinationen für weitere konstante Lösungen.

### Aufgabe 5 (optional)

Implementieren Sie die Trapez-Methode

$$M \cdot \frac{(u_{n+1}, v_{n+1}) - (u_n, v_n)}{dt} + \frac{1}{2}\Big(A \cdot (u_{n+1}, v_{n+1}) + A \cdot (u_n, v_n)\Big) = \frac{1}{2}\Big(f(u_{n+1}, v_{n+1}) + f(u_n, v_n)\Big).$$

anstelle des impliziten Eulerverfahrens. In dem Fall muss in jedem Zeitschritt das nichtlineare System bezüglich $(u_{n+1}, v_{n+1})$ gelöst werden.

# A Periodische Geometrie, Mesh

```
l_dom = 1.5 # Laenge des Quadrats
n_dom = 75 # Anzahl Elemente pro Kante
rec = MoveTo(-l_dom/2,-l_dom/2).Rectangle(l_dom,l_dom).Face()

# Namen der Kanten
rec.edges.Max(Xocc).name = 'right'
rec.edges.Min(Xocc).name = 'left'
rec.edges.Max(Yocc).name = 'top'
rec.edges.Min(Yocc).name = 'bottom'

# Identifizierung links <-> rechts
right=rec.edges.Max(Xocc)
rec.edges.Min(Xocc).Identify(right,name="left")

# Identifizierung oben <-> unten
top=rec.edges.Max(Yocc)
rec.edges.Min(Yocc).Identify(top,name="bottom")

# Geometrie erstellen
geo= OCCGeometry(rec,dim=2)

# Mesh generieren
mesh = Mesh(geo.GenerateMesh(maxh=l_dom/n_dom))
```

Listing 5: Periodische Geometrie, Mesh

# B Bilinearformen

```
eps1 = Parameter(2e-5)
eps2 = Parameter(1e-5)
F = Parameter(0.048)
k = Parameter(0.065)
```

Listing 6: Parameter

```
a = BilinearForm(X,symmetric=True)
a += eps1*grad(u)*grad(w)*dx
a += eps2*grad(v)*grad(q)*dx
a.Assemble()

m = BilinearForm(X,symmetric=True)
m += u*w*dx
m += v*q*dx
m.Assemble()
```

Listing 7: Bilinearformen

```
f = LinearForm(X)
f += dt*(-gfuold*gfvold**2+F*(1-gfuold))*w*dx(bonus_intorder=4)
f += dt*(gfuold*gfvold**2-(k+F)*gfvold)*q*dx(bonus_intorder=4)
```

Listing 8: Linearform

# C Zeitabhängige Lösung

```
scene = Draw(gfu)
scene2 = Draw(gfv)

res = gfx.vec.CreateVector()
deltauv = gfx.vec.CreateVector()
with TaskManager():
    for j in range(2000):
        f.Assemble() # nicht vergessen!
        <snipp, selber machen>

        # Update Visualisierung in jedem 10. Schritt
        if j % 10 == 0:
            scene.Redraw()
            scene2.Redraw()
        deltauvNorm = deltauv.Norm()
        print(j,deltauvNorm,end='\r')
        # Abbruch, wenn sich nichts mehr aendert
        if deltauvNorm < 1e-8:
            print('stopping, no change in solution')
            break
```

Listing 9: Zeit-Loop

# D Visualisierung

```
# Anzahl Pixel
Npixel = 800
# Sampling kartesisch
xi = np.linspace(-l_dom/2,l_dom/2,Npixel)
Xi,Yi = np.meshgrid(xi,xi)
mips = mesh(Xi.flatten(),Yi.flatten())

# Loesung berechnen
ui = gfu(mips)
ui = ui.reshape((Npixel,Npixel))

# Visualisieren
plt.figure(figsize=(5,5))
plt.imshow(ui>.5, cmap='gray_r')
plt.axis('off')
plt.tight_layout()
plt.savefig('example.pdf')
plt.show()
```

Listing 10: Export Figure