Karlsruher Institut für Technologie

Institut für Angewandte und Numerische Mathematik
Prof. Dr. Tobias Jahnke, Dipl.-Math. techn. Patrick Krämer, M. Sc. Simon Baumstark

Einführung in das Wissenschaftliche Rechnen

Sommersemester 2016

Übungsblatt 8: Räuber-Beute-Systeme

Abgabe spätestens am 20.07.2016

Aufgabe 8:
Ein räumlich inhomogenes Räuber-Beute-System kann durch die folgenden gekoppelten Reaktions-Diffusionsgleichungen modelliert werden:

$$\partial_t u_1(t,x) = \delta_1 \Delta u_1(t,x) + F_1\big(u_1(t,x), u_2(t,x)\big) \qquad \forall x \in \Omega, t > 0 \tag{1a}$$

$$\partial_t u_2(t,x) = \delta_2 \Delta u_2(t,x) + F_2\big(u_1(t,x), u_2(t,x)\big) \qquad \forall x \in \Omega, t > 0 \tag{1b}$$

$$\frac{\partial u_i(t,x)}{\partial \eta} = \nabla u_i(t,x)^T \eta(x) = 0 \qquad \forall x \in \Gamma, t > 0, i \in \{1,2\} \tag{1c}$$

$$u_i(0,x) = u_i^0(x) \qquad \forall x \in \overline{\Omega}, i \in \{1,2\} \tag{1d}$$

mit Diffusionskonstanten $\delta_1 > 0$, $\delta_2 > 0$, nichtlinearen Reaktionstermen

$$F_1(u_1,u_2) = (c_3 u_2 - c_1)u_1, \qquad F_2(u_1,u_2) = (c_4 - c_5 u_2 - c_2 - c_6 u_1)u_2$$

und Konstanten $c_j > 0$. Dabei ist $u_1(t,x)$ die Dichte der Räuber und $u_2(t,x)$ die Dichte der Beute (siehe Vorlesung).

Ziel der Aufgabe ist es, die Lösung $\big(u_1(t,x), u_2(t,x)\big)$ von (1) mit dem in der Vorlesung vorgestellten Splitting-Verfahren numerisch zu approximieren. Sie müssen die numerische Lösung nicht speichern, was bei feinen Triangulierungen und vielen Zeitschritten wohl sowieso zu Problemen führen würde. Es genügt, wenn Sie die Lösung nach jedem Zeitschritt plotten bzw. einen Film der Lösung erstellen (Details siehe unten).

Weil dies das letzte Übungsblatt ist, dürfen Sie Ihrer Kreativität freien Lauf lassen, d.h. Sie dürfen ein hübsches Gebiet definieren, mit Hilfe von `distmesh` eine geeignete Triangulierung erzeugen und alle Daten – Werte der Konstanten, Anfangsdaten, Länge des Zeitintervalls – selber wählen. Einzige Nebenbedingung: Die Lösung soll einigermaßen interessant sein, also z.B. nicht die konstante Lösung $u_1 = u_2 = 0$. Die Dateien, die Sie zur Erzeugung der Triangulierung verwendet haben, müssen Sie nicht abgeben. Es genügt, wenn sie die Datei abgeben, in welcher die Triangulierung gespeichert ist.

Ihre Lösung sollte aus folgenden Programmen zw. Funktionen bestehen:

- Hauptprogramm `aufgabe08.m`
  Hier werden die Triangulierung geladen, die Konstanten, Parameter und Anfangsdaten definiert und das Splitting-Verfahren ausgeführt wie in der Vorlesung besprochen. Die Cholesky-Zerlegung von Matrizen können Sie mit dem Befehl `L = chol( ... ,'lower')` berechnen.

- `function [M,A] = computeMA(T,P)`
  Assemblierung der Steifigkeitsmatrix A und der Massematrix M.

- `function [F,FJac] = evalF(u,c)`
  Der Vektor `c` enthält die Konstanten $c_1, \ldots, c_6$. Die zweispaltige Matrix `u` enthält die Werte der aktuellen Approximationen von u_1 bzw. u_2. Die zweispaltige Matrix `F` enthält die Auswertung der Reaktionsterme $F_1(u_1, u_2)$ und $F_2(u_1, u_2)$. Jede Zeile entspricht einem Punkt der Triangulierung, d.h. in Zeile 17 von `F` steht

  $$\Big(F_1\big(u_1^n(x_{17}, y_{17}), u_2^n(x_{17}, y_{17})\big), F_2\big(u_1^n(x_{17}, y_{17}), u_2^n(x_{17}, y_{17})\big)\Big)$$

  mit $P_{17} = (x_{17}, y_{17})$, wobei u_1^n und u_2^n die aktuellen Approximationen sind (die in den Spalten von `u` stehen).

  Gleichzeitig wird auch die ("von Hand" berechnete) Jacobi-Matrix `FJac` ausgegeben, weil wir die für das Newton-Verfahren (siehe unten) brauchen. Genauer: Die Matrix hat `FJac` vier Spalten, die den Einträgen

  $$\partial_{u_1} F_1(u_1, u_2), \partial_{u_1} F_2(u_1, u_2), \partial_{u_2} F_1(u_1, u_2) \text{ und } \partial_{u_2} F_2(u_1, u_2)$$

  entsprechen. Zeilen von `FJac` entsprechen den Punkten der Triangulierung, an denen die Jacobi-Matrix ausgewertet wird.

- `function [] = visualize(T,P,uFEM,t)`
  Visualisierung der Lösung. Diese Funktion soll nach jedem Zeitschritt aufgerufen werden. Dabei soll allerdings *nicht* jedes Mal ein neues `figure` geöffnet werden – sonst haben Sie nach 100 Zeitschritten den Bildschirm voller Bilder.

  Sie können die Lösung mit dreidimensionale Plots (mit `trimesh` oder `trisurf`) oder zweidimensionale Projektionen (d.h. Lösung "von oben" betrachtet mit "Farbe=Höhe") visualisieren. Bei mir sieht das z.B. so aus (die beiden Spalten von `uFEM` enthalten die Werte der aktuellen Approximation):

```
fs=20; % Font Size
x=P(:,1); y=P(:,2);

subplot(1,2,1)
trisurf(T,x,y,uFEM(:,1))
xlabel('x'); ylabel('y')
title(['Raeuber  (t = ',num2str(t,'%2.2f'),')'])
axis square; axis tight
set(gca,'FontSize',fs)
view(2)
colorbar

subplot(1,2,2)

... entsprechend fuer die Beute ...

drawnow
```

- `u = solve_ode_mpr(uold,tau,c,tolres)`
  Diese Funktion soll einen Schritt der impliziten Mittelpunktsregel angewandt auf den Reaktionsteil ausführen. In jedem Punkt $P_k = (x_k, y_k)$ der Triangulierung muss die gewöhnliche Differentialgleichung

  $$\dot{w}_1 = F_1(w_1, w_2), \qquad \dot{w}_2 = F_2(w_1, w_2) \tag{2}$$

gelöst werden, wobei $w_i(t)$ die Approximation von $u_i(t,(x_k,y_k))$ ist. Gehen Sie deshalb am Besten so vor, dass Sie zuerst eine Funktion zur Lösung von (2) mit $w(t) \in \mathbb{R}^2$ schreiben. Die Mittelpunktsregel angewandt auf (2) ist bekanntlich

$$w^{n+1} = w^n + \tau F\left(\frac{w^{n+1}+w^n}{2}\right).$$

Sie müssen also die Nullstelle der Funktion

$$G(w) = w^n + \tau F\left(\frac{w+w^n}{2}\right) - w$$

mit dem Newton-Verfahren bestimmen. Als Startwert für das Newton-Verfahren wählen Sie die alte Approximation w^n. Das Newton-Verfahren soll abgebrochen werden, wenn für das Inkrement Δw des Newton-Verfahrens $\|\Delta w\| \leq \texttt{tol}$ gilt[1] Dabei meint $\|\cdot\|$ die euklidische Vektornorm. Für `tol` wählen Sie einen hinreichend kleinen Wert. Für das Newton-Verfahren brauchen Sie die Jacobi-Matrix von G, was aber kein Problem ist, da die Jacobi-Matrix von F ja mit der Funktion `evalF(u,c)` berechnet wird. In jedem Schritt des Newton-Verfahrens muss bekanntlich ein lineares Gleichungssystem gelöst werden. Das können Sie in diesem Fall jedoch vermeiden, da die Jacobi-Matrix ja nur eine 2×2-Matrix ist und deshalb die Inverse bekannt ist.

Wenn Ihre Funktion für die ODE (2) in *einem* Punkt funktioniert, dann modifizieren sie die Funktion nun so, dass sie die ODE (2) in allen Punkten der Triangulierung *gleichzeitig* lösen, indem Sie alle Operationen vektorisieren.

Als Belohnung für die viele Arbeit können Sie einen Film der Lösung erstellen. Wie das geht können Sie anhand des Demos sehen, das auf der Seite der Vorlesung steht.

[1] In der Vorlesung wurde ein anderes (schlechteres) Abbruchkriterium vorgeschlagen.