Goethe-Center for Scientific Computing (G-CSC)
Goethe-Universität Frankfurt am Main

Neurobioinformatik

(Übung NBI, WS 2018/19)

M. Huymayer, J. Wang, Dr. A. Nägel, Dr. M. Hoffer

Aufgabenblatt 9 Abgabe Montag, 11.2.2019, 16h

Auf diesem (letzten) Blatt können Sie das gesamte in der Vorlesung erworbene Wissen zum Einsatz bringen. Gelöst werden sollen die Monodomain-Gleichungen, welche die Signalleitung im Herzmuskel beschreiben. Aufgabe 1 behandelt dazu theoretisch das FitzHugh-Nagumo-Modell für Punktneuronen. In Aufgabe 2 wird dies mit einem Mono-Domain-Modell für den diffusiven Transport der Spannung gekoppelt.[1] In Aufgabe 3 können Sie dies auf den 3D-Fall übertragen.

Aufgabe 1 (3 P)
Gegeben sind die FitzHugh-Nagumo-Gleichungen, die Sie schon in Blatt 5 kennen gelernt haben. Diese beschreiben die Signalverarbeitung eines Punkt-Neurons:

$$\frac{\partial u}{\partial t} = f(u,w) := c_1 u(u-a)(1-u) - c_2 w \tag{1a}$$

$$\frac{\partial w}{\partial t} = g(u,w) := c_3(u - bw) \tag{1b}$$

über das Membranpotential u bzw. eine Aktivierungsvariable w. Beide Größen sowie die Paramter a, b sind dimensionslos. Bei den Parametern c_1, c_2, c_3 handelt es sich um Reaktionsraten.

a) Führen Sie eine Stabilitätsanalyse für den Gleichgewichtspunkt $(u_0, w_0)^T = (0,0)^T$ durch: Bestimmen Sie hierfür zunächst die zur rechten Seite von (1) gehörige Jacobi-Matrix $J = (J_{ij})_{1\leq i,j\leq 2}$ für allgemeine Parameter (siehe Blatt 5). Bestimmen Sie für die konkreten Parameter $a = 0.1$, $b = 0.55$ sowie $c_1 = 0.175 ms^{-1}$, $c_2 = 0.03 ms^{-1}$, $c_3 = 0.011 ms^{-1}$ zunächst die beiden Größen

$$\begin{aligned} \mu &:= Spur(J) := \sum J_{ii}, \\ \triangle &:= detJ = J_{11}J_{22} - J_{12}J_{21}. \end{aligned}$$

Die Eigenwerte von J sind dann durch

$$\lambda_{1/2} = \frac{\mu \pm \sqrt{\mu^2 - 4\triangle}}{2}$$

[1] Dieses Modell ist das mehrdimensionale Analogon zur Kabelgleichung.

gegeben. Sind die Eigenwerte im vorliegenden Fall reeellwertig? Ist das System stabil oder instabil?

Aufgabe 2 (10+4 P)
Erweitert man das System (1) um Diffusion, so ergibt sich das *Monodomain-Modell*, welches z.B. die Signalleitung am Herzmuskel beschreibt:

$$\frac{\partial u}{\partial t} + \nabla \cdot [-\mathbb{D}\nabla u] - f(u,w) = 0 \tag{2a}$$

$$\frac{\partial w}{\partial t} - g(u,w) = 0 \tag{2b}$$

Lösen Sie unter Verwendung der Vorlage `vorlage102.lua` (sowie der im Anhang nachfolgenden Hinweise) die folgenden Aufgaben:

a) Lösen Sie Gleichung (2) auf einem quadratischen Gebiet $\Omega \subset \mathbb{R}^2$ mit den Abmessungen $10cm \times 10cm$. Verwenden Sie *isotrope* Diffusion $\mathbb{D} = D_f$ mit $D_f := 10^{-3} cm^2/ms$, ansonsten sei das Modell wie in Aufgabe 1 a) angegeben parametrisiert. Auf dem Rand $\partial\Omega$ seien (zeitlich konstante) Dirichlet-Randwerte $u = w = 0$ vorgegeben. Der Anfangswert sei $u = 0.6$ in einem Kreis mit Durchmesser $2\,cm$ in der Mitte des Gebietes sowie ansonsten $u = w = 0$.
Tipp: Bauen Sie mittels ProMesh ein Quadrat mit Subsets Inner und Boundary. Für den Kreis brauchen Sie kein eigenes Subset. Legen Sie den Anfangswert für diesen Bereich in der dafür vorgesehenen Lua Funktion in Ihrem Skript mittels x und y Koordinaten fest.

b) Am Herzmuskel ist auch die Orientierung der Muskelfasern zu berücksichtigen. Für ein Gebiet $\Omega \subset \mathbb{R}^d$ ist $\mathbb{D} = \mathbb{D}(x)$ dann als $d \times d$ Matrix anzugeben. Der *anisotrope* Diffusionstensor ($d = 2$)

$$\mathbb{D} := \begin{pmatrix} D_f & 0 \\ 0 & D_f/4 \end{pmatrix},$$

spezifiziert z.B., dass die Diffusion in x-Richtung zweimal schneller verläuft als in y-Richtung. Wiederholen Sie mit dieser Änderung das Experiment aus a). Wie ändert sich das Ergebnis?

Tipp: Achten Sie darauf, dass Sie genug Verfeinerungen nutzen.

Aufgabe 3 (6 P)
Es soll nun eine Simulation für den Verlauf eines Signals auf einer Geometrie in 3D durchgeführt werden. Dazu ist es nun nur ein kleiner Schritt. Als

Grundlage können Sie Ihr Programm aus Aufgabe 2 verwenden, wobei Sie die dritte Raumdimension über den Befehl `InitUG(3, AlgebraType(''CPU'', blockSize))` aktivieren.

Die dieser Aufgabe zugrundeliegenden Geometrien finden Sie in den Dateien `model1.ugx` und `model1_cardio.ugx`. Diese Volumengitter basieren auf einem Ellipsoid, welches Sie auf einem vorherigen Blatt bereits erzeugt haben. Wie in Abb. 1 dargestellt ist, ist das Ellipsoid im einen Fall vollständig gefüllt, während im anderen Fall ein Hohlraum verbleibt, welcher an die Herzkammer erinnern soll.

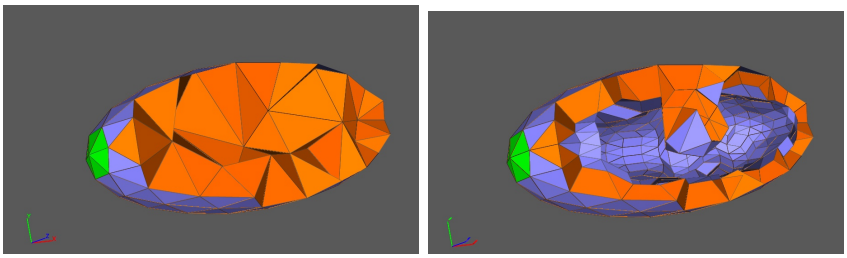

Abbildung 1: Querschnitt durch die Geometrien `model1.ugx` (links) und `model1cardio.ugx` (rechts).

Führen Sie nun mit dem bekannten Modell, jedoch der Diffusionskonstante $D_f = 0.01 cm^2/ms$ eine Simulation für $1s$ mit konstanter Zeitschrittweite $\tau = 5ms$ durch. Verwenden Sie dazu die Anfangswerte

```
function InitialValueU_3D(x, y, z)
  if ((x-7.4)*(x-7.4)+y*y+z*z <= 4) then  return 0.6
  else return 0.0 end
end

function InitialValueW_3D(x, y, z)
    return 0.0
end
```

sowie die Dirichletrandwerte

```
local dirichletBND = DirichletBoundary()
dirichletBND:add(0.0, "u", "GND")
dirichletBND:add(0.0, "w", "GND")
```

Wie unterscheiden sich die Ergebnisse auf den beiden Geometrien?

Hilfestellung:

- Für das System (2) wird zunächst ein größerer Ansatzraum benötigt:

```
-- Definiere Ansatzraum
local approxSpace = ApproximationSpace(dom)
approxSpace:add_fct("u", "Lagrange", 1)
approxSpace:add_fct("w", "Lagrange", 1)
```

- Zudem benötigt jede Gleichung eine eigene Elementdiskretisierung:

```
-- Definiere Elementdiskretisierungen
local elemDisc = {}
elemDisc["u"] = ConvectionDiffusion("u", "Inner", "fv1")
elemDisc["w"] = ConvectionDiffusion("w", "Inner", "fv1")
```

  Die Parameter für eine Konvektions-Diffusions-Reaktionsgleichung $\partial_t u + \nabla \cdot [-\mathbb{D}\nabla u] + \rho = q$ können bekanntermaßen wie folgt gesetzt werden:

```
elemDisc["u"]:set_diffusion(dDiffusion)
elemDisc["u"]:set_reaction(rhoReaction)
elemDisc["u"]:set_source(qSource)
```

  Dabei sind `dDiffusion`, `rhoReaction`, `qSource` entweder Zahlen oder durch benutzerdefinierte Objekte des Typs `UserFunction` gegeben.

- Dies soll für eine Gleichung der Form mit $\rho = \rho(u, w) := \sin(u)w^2$ illustriert werden. Eine solche Funktion definiert man z.B. wie folgt:

```
-- Definiere Funktion und ihre Ableitungen
function LuaReactionFct(u,w) return sin(u)*w*w end
function LuaReactionFct_du(u,w) cos(u)*w*w end
function LuaReactionFct_dw(u,w) return sin(u)*2*w  end

-- Definiere ein Objekt für rho
rhoReaction = LuaUserFunctionNumber("LuaReactionFct", 2)
rhoReaction:set_input(0, elemDisc["u"]:value())
rhoReaction:set_input(1, elemDisc["w"]:value())
rhoReaction:set_deriv(0, "LuaReactionFct_du")
rhoReaction:set_deriv(1, "LuaReactionFct_dw")
```

- Einen Diffusionstensor erzeugen Sie z.B. über

```
anisoDiff = ConstUserMatrix()
anisoDiff:set_diag_tensor(Df)     -- Setze Diagonalmatrix
anisoDiff:set_entry(1,1, Df/4) -- Setze Diagonaleintrag (y-Richtung)
```

Alternativ können Sie die Matrix auch über Funktionen definieren:

```
function Diff2D(x,y) return Df, 0, 0, Df/4 end
function Diff3D(x,y) return Df, 0, 0, 0, Df/4, 0, 0, 0, Df/4 end
```

und dann mit `elemDisc["u"]:set_diffusion("Diff2D")` die Elementdiskretisierung initialisieren.

Anmerkung: Senden Sie den Quelltext als VRL-Studio Projekt (.vrlp Datei) und als Lua Skripte (.lua Datei), die erstellten Geometrien (.ugx), ParaView Ergebnisse als Screenshots und die Antworten zu den Fragen am besten als Word oder PDF Datei. Plots senden Sie bitte als pdf oder png Dateien und Daten Ausgaben als Text Dateien.
Senden Sie Ihre Lösungen an `practical.sim1@gcsc.uni-frankfurt.de`.
Abgabe bis spätestens Montag, 11.2.2019, 16h.