Übungen zur Vorlesung

Mathematische Aspekte der Modellierung und Simulation in den Neurowissenschaften

Dr. S. Lang, D. Popović

Abgabe: 18. Mai 2010 in der Übung

Beachten Sie: Die Übung nächste Woche muss ausfallen. Sie haben daher zwei Wochen Zeit zur Bearbeitung der Aufgaben!

Übung 6 Vektorfelder in Octave (2 Punkte)

Im Listing sehen Sie dem Plot des Richtungs- oder Vektorfeldes eines Systems gewöhnlicher Differential-Gleichungen:

```
% example vector field plot
[x1, x2] = meshgrid(-.5:.05:.5, -.5:.05:.5);
x1dot = -x1 - 2 * x2 .* x1.^2 + x2;
x2dot = -x1 - x2;
quiver(x1, x2, x1dot, x2dot)
grid
xlabel('x1dot');
ylabel('x2dot');
title('Vector field phase portrait');
```

1. Welches System wurde hier geplottet?
2. Implementieren Sie den Code in *Octave* und führen Sie Ihn aus. Welche signifikanten Punkte des Plots erkennen Sie (Es gibt nur einen)? Was könnte der Plot bzw. der Punkt aussagen?

Übung 7 Phasenräume in Octave (3 Punkte)

Auf der Vorlesungs-Homepage finden Sie auf der Octave-Seite einen neuen Unterabschnitt zum Berechnen und Visualisieren eines Phasenraumportraits in *Octave*, in dem für ein zweikomponentiges System das Portrait der beiden Komponenten $v_1 - v_2$ geplottet wird.

Orientieren Sie sich am dortigen einfachen Beispiel, um ein Phasenraumportrait des folgenden dynamischen Systems mit *Octave* zu berechnen:

$$
\begin{aligned}
y_1'(t) &= y_2(t) \\
y_2'(t) &= -sin(y_1(t)) + sin(5t) \\
y_1'(0) &= 1 \\
y_2'(0) &= 0.
\end{aligned}
$$

Das System kann aus einem System 2.ter Orndung, das ein Fadenpendel beschreibt, und von einer zeitabhängigen Kraft $F(t) = sin(5t)$ angeregt wird, abgeleitet werden.

Exportieren Sie einen Plot des Phasenportraits dieses Systems bei den gegebenen Anfangsbedingungen.

Übung 8 Izhikevič-Modell mit C++ (5 Punkte)

In der Vorlesung wurde ein Punkt-Neuronen-Modell von Izhikevič und eine Implementierung mit *Octave* vorgestellt. In dieser Aufgabe soll das Modell mit C++ implementiert werden und die Performance beider Implementierungen (C++, *Octave*) verglichen werden.

1. Implementieren Sie das Modell in C++ mit den gleichen Parametern wie in *Octave* und überprüfen Sie die Ergebnisse auf Richtigkeit (Vergleich mit der Octave-Ausgabe). Um ein Programm in C++ zu implementieren, öffnen Sie im Pool eine Datei in einem Editor (z.B. `gedit`) und speichern den Code in einer Datei mit der Endung `.cc`:

   ```
   $ gedit hallowelt.cc &
   ```

   Das `&` bewirkt, daß der Editor im Hintergrund geöffnet wird, und man trotzdem in der Konsole weiterarbeiten kann. Das aus den Übungen bekannte Hallo, Welt!-Programm kann dann etwa wie folgt aussehen:

   ```
   1  // include i/o library
   2  #include <iostream>
   3
   4  // main is always the first function to be called
   5  int main(int argc, char** argv)
   6  {
   7    std::cout << "Hello, World..." << std::endl;
   8
   9    return 0;
   10 }
   ```

   Um das Programm zu kompilieren, können Sie den C++-Compiler `g++` verwenden:

   ```
   $ g++ -o hallowelt hallowelt.c
   ```

   Die Option `-o <name>` erstellt ein ausführbares Programm mit Namen `<name>`. Dieses kann dann mit folgendem Befehl in der Konsole ausgeführt werden:

   ```
   $ ./hallowelt
   ```

2. Messen Sie die Rechenzeiten für die C++- und *Octave*-Implementierungen für $t = 2000ms$ und $dt = \{0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125\}ms$. Achten Sie darauf, daß Sie ausreichend lange Rechnungen machen, da die Zeitmessung einerseits selber mit einem Fehler im Sekundenbereich behaftet ist und anderseits der C++-Code sonst zu schnell ist. Bei mir reichte es, für jedes dt aus der Liste 50 Durchläufe zu machen um hinreichend große Zeiten für die C++-Simulation zu erhalten (Die *Octave*-Simulation dauerte dann natürlich etwas länger). Gerne können Sie dieses Problem anders lösen. Jede Simulation mit 50 Läufen wurde dann noch jeweils drei mal ausgeführt um zu mitteln.

   Stellen Sie die Zeiten in einem doppelt-logarithmischen Plot Zeitschrittweite-Rechenzeit gegenüber und untersuchen Sie, ob es ein bestimmtes Verhältnis der Rechenzeiten gibt. Auf jeden Fall ist *Octave* um Größenordnungen langsamer – haben Sie eine Idee, woran das liegt?

   Die Rechenzeiten können im Pool und generell unter Unix/Linux am einfachsten mit den Befehlen

   ```
   $ /usr/bin/time octave izhikevich.m > oct.dat
   $ /usr/bin/time ./izhikevich_cpp > cpp.dat
   ```

gemessen werden. Hierbei ist `izhikevich.m` ein *Octave*-Skript und `izhikevich_cpp` das C++-Executable. Die Rückgabe des Befehls ist ungefähr eine Ausgabe wie

```
0.05user 0.04system 0:00.75elapsed 12%CPU (0avgtext+0
  avgdata
3920maxresident)k
0inputs+0outputs (0major+298minor)pagefaults 0swaps
```

in der uns die `elapsed`-Zeit interessiert (hier 0 : 00.75), daß ist die verstrichene Echtzeit in *s*. Die Umleitung nach dem Befehl mittels `> datei.dat` bewirkt, daß eventuelle Ausgaben des Programms in eine Datei umgeleitet werden.

Übung 9 Spiking Patterns des Izhikevič-Modells (5 Punkte)

Eine generelle Formulierung des in der Vorlesung vorgestellten Izhikevič-Modells ist auf der Webseite von Herrn Izhikevič verfügbar und lautet

$$\begin{aligned}
\dot{v} &= 0.04 \cdot v^2 + -u + I + 140 \\
\dot{u} &= a(b \cdot v - u) \\
v &= c \qquad \text{falls } v > 30; \\
u &= u + d \qquad \text{falls } v > 30;
\end{aligned}$$

In Abbildung 0.1 sehen Sie einige der Spiking Patterns, die mit diesem Modell erzeugt werden können.

Implementieren Sie das Modell mit *Octave* oder modifizieren Sie den in der Vorlesung gezeigten Code und versuchen Sie, die Figuren aus Abbildung 0.1 zu reproduzieren. Der Startwert des Potentials liegt meist zwischen -70 und -60 mV, die Zeitschrittweite für das Forward Euler-Verfahren bei ungefähr $dt = 0.25$. Der applizierte Strom liegt bei Izhikevič meist um die 0.5 mV für (TC) und eventuell für (LTS), ansonsten bei etwa 15 mV. Ganz sicher ist dies aber nicht.

Sie sehen die angegebenen Werte für die freien Parameter a, b, c, d. Fertigen Sie Plots für die jeweiligen Parameter an. Schaffen Sie es, die Plots hinzubekommen? Sie können über die Funktion `subplot` in *Octave* übrigens mehrere Plots in einem Bild platzieren. Wenn die Plots nicht voll übereinstimmen ist es kein Problem, geben Sie einfach die beste Näherung, die Sie geschafft haben, ab, auch wenn diese signifikante Abweichungen enthält. Eventuell kann man aus diesen Abweichungen Rückschlüsse auf das Modellverhalten ziehen.

Haben Sie eine Erklärung für den Summanden 140 im Modell?

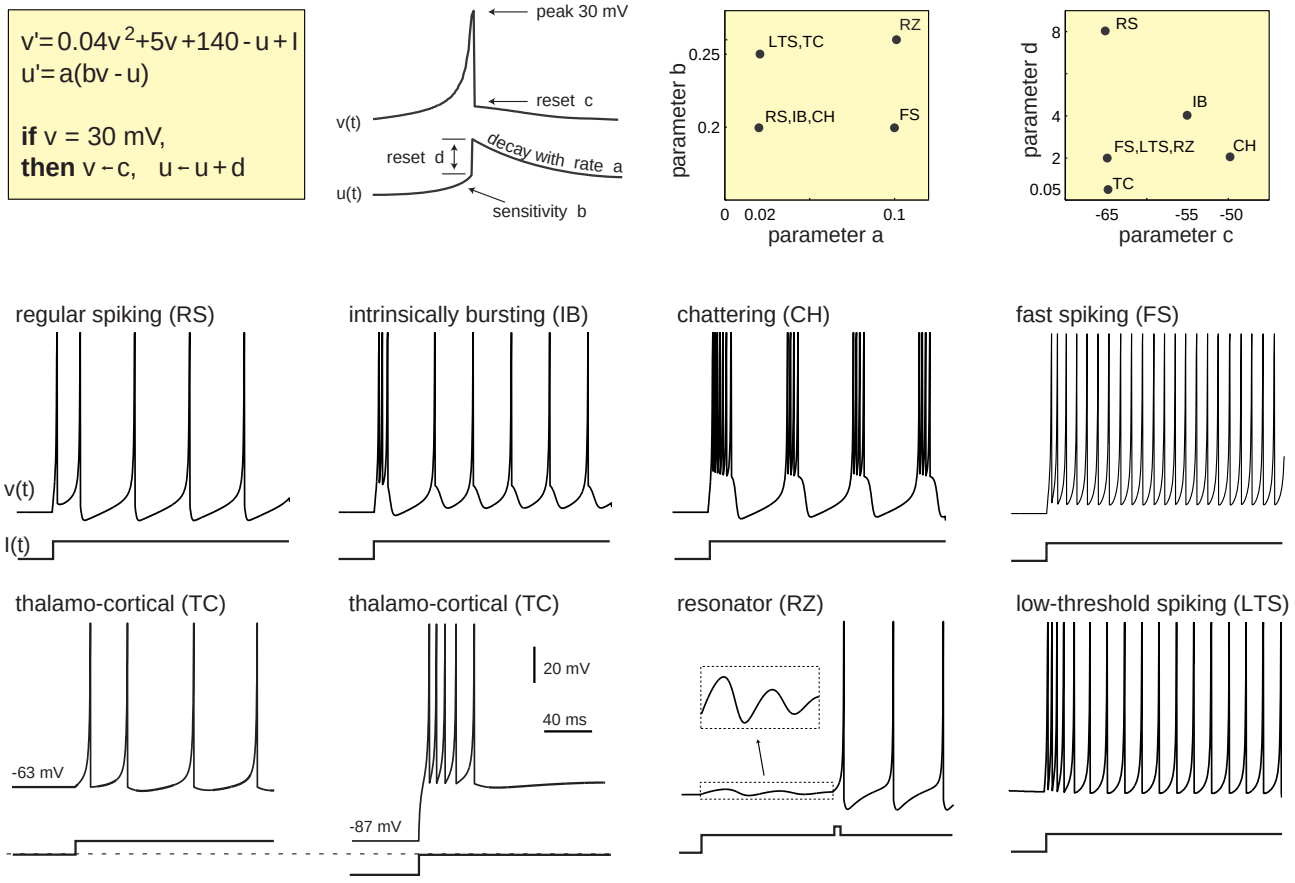

Abbildung 0.1: Spiking Patterns of Izhikevič's model. Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com