**Tag 6: Datenformate, Datenaufnahme und Zufallszahlen**

**A) Datenformate**

Wie wir bereits wissen speichert Matlab seine Daten standardmäßig als .mat Dateien ab, die ausschließlich von Matlab selber vernünftig weiterverarbeitet werden können. Die Alternative ist, die Daten entweder im Text-Format (ascii) oder z.B. als Excel-Datei abzuspeichern.

Für das Abspeichern von Abbildungen verwendet Matlab normalerweise das fig-Format. Auch wenn diese Endung von verschiedenen Programmen vergeben wird,

ist dieses Format ebenfalls nur von Matlab selber vernünftig lesbar.

Die einfachste Möglichkeit zum Abspeichern von Abbildungen ist den Menüpunkt ">File >Save as" zu benutzen. Dort kann man verschiedene Dateiformate auswählen. Das von Matlab als Standard angebotene .fig-Format gewährleistet, dass beim erneuten Einladen alle Eigenschaften der Abbildung beibehalten werden. Unter anderem sind z.B. bei einer abgespeicherten Ausschnittsvergrößerung nach erneutem Einladen noch alle Daten vorhanden – auch die nicht im Bild sichtbaren – so dass man wieder auf den ursprünglichen Bereich zoomen kann.

Wenn man die Abbildungen mit einem anderen Programm benutzen will, z.B. um sie in ein mit Word geschriebenes Protokoll einzubinden, muss man ein anderes Dateiformat benutzen als .fig Dazu kann man wie oben beschrieben abspeichern, muss jedoch das gewünschte Dateiformat auswählen und die Endung der Datei entsprechend anpassen. Wenn man die Abbildung z.B. als jpg abspeichert, lässt sie sich z.B. in Word gut einbinden. Man kann sie auch in Matlab wieder einladen, allerdings ist sie dann auf genau diese Darstellung festgelegt. Zusätzliche Informationen z.B. über abgeschnittene Wertebereiche gehen verloren.

Wir werden uns etwas ausführlicher mit dem Grafik-Format jpg beschäftigen, um zu sehen, wie man in Matlab mit diesem häufig benutzten Format umgehen kann.

T6A1) Eine wichtige Voraussetzung zum Verständnis von Grafik-Formaten ist die Farbkodierung mit der man die in einer Grafik genutzten Farben genau festlegen kann. Farben werden in der sogenannten RGB-Kodierung dargestellt, d.h. sie werden durch einen Vektor farbe=[rot gruen blau] definiert. In Matlab werden zur Kodierung für rot, grün und blau Werte zwischen 0 und 1 angegeben, z.B. ist

[1 0 0] rot, [0 0 0] schwarz, [1 1 0] gelb, [0.6 0 0.6] dunkel-violett. So lassen sich beliebige Farben mischen.

Für ein ganzes Bild braucht man dementsprechend eine dreidimensionale Matrix: jeder Bildpunkt wird durch die beiden Koordinaten (erste beiden Dimensionen) und die drei Farbwerte (dritte Dimension) dargestellt. Die Funktionsweise der RGB-Kodierung demonstriert folgendes Skript: [simpelbild.m]

\*) Wenn Sie Lust auf ausgefeilte Grafik haben: Stellen Sie den Vektor 0:1:10 mit einer durchgezogenen dunkelroten Linie und die einzelnen Punkte mit türkis gefüllten Kreisen dar (hierfür braucht man "line properties", siehe T1H5).

T6A2) Ein wichtiges Grafik-Format ist jpg. jpg-Dateien können in Matlab eingeladen und bearbeitet werden. Darin ist jeder Punkt durch einen Vektor von drei Farbwerten kodiert (allerdings etwas unterschiedlich zum in Matlab üblichen Format). Bei zweidimensionalen Abbildungen ergibt sich also eine 3 dimensionale Matrix der Größe MxNx3 (M und N ist die Anzahl Pixel in vertikaler und horizontaler Richtung). Schauen Sie sich die Arbeitsweise folgenden Skripts an:

[meisenshow.m] hierzu brauchen Sie außerdem die Datei [bmeise.jpg]

Die wichtigen Befehle darin sind: bild=imread(datei.jpg) zum Einlesen und image(bild) zum Darstellen des jpg-Bildes.

Wie unterscheidet sich die Farbkodierung dieser jpg-Datei von der in Matlab üblichen?

Erweitern Sie die Show um eine Meise mit zwei Köpfen.

Teilen Sie die Matrix in drei Teile, um sich die Rot-, Grün- und Blau-Anteile des Bildes einzeln anzusehen.

\*T6A3) Da man Grafik meistens sowieso ansehen möchte, bevor man sie abspeichert, bietet sich meistens der Weg über die Menüleiste an. In manchen Fällen (z.B. wenn man für seine Bachelorarbeit viele gleich formatierte Abbildungen braucht) möchte man jedoch aus einem Skript oder einer Funktion heraus Abbildungen speichern. Hierfür gibt es den Befehl saveas. Damit dieser Befehl weiß, welche Grafik gespeichert werden soll, muss diese jedoch beim plotten benannt werden. Probieren Sie aus:

x=1:1:100; y=sqrt(x);

h=plot(x,y);

saveas(h,'testbild.fig')

Schließen Sie die Abbildung und laden Sie testbild.fig neu ein.

Das Praktische an dieser Art des Speicherns ist, dass Matlab direkt verschiedene Formate speichern kann und das gewünschte Dateiformat an der Endung des Dateinamens erkennt. Unter anderem können .jpg, .eps, .pdf und .bmp benutzt werden. (Die gesamte Liste möglicher Formate und ihrer Endungen findet man in der Hilfe unter saveas.)

\*) Probieren Sie, die Abbildung z.B. als bmp abzuspeichern und in Word zu importieren.

\*T6A4) Für das Abspeichern von Daten, die nur aus Zahlen bestehen, haben wir letzte Woche schon gesehen, wie man mit Textdateien umgeht: Abspeichern mit save dateiname variablennamen –ascii einladen mit load dateiname.

Falls Sie mit als Text gespeicherten numerischen Daten umgehen müssen, die durch bestimmte Zeichen getrennt sind (z.B. Semikolon), brauchen Sie den Befehl dlmread (delimited characters read).

Komplizierter ist die Sache, wenn eine Datei gemischte numerische und ASCII Daten enthält. Hierfür gibt es die Funktion textread. Sie basiert darauf, dass man durch eine Abfolge von %s (string) und %f (float Zahlen) das Muster angibt, mit dem jede der Zeilen aufgebaut ist.

Schauen Sie sich bei Interesse in der Hilfe an, wie diese Funktionen funktionieren und probieren Sie diese aus.

\* T6A5) Speziell für Nutzerinnen und Nutzer von Excel: Matlab kann excel-Dateien einlesen mit dem Befehl variable=xlsread('dateinme.xls'). Probieren Sie das mit der Datei [entfernungen.xls] aus.

Der Befehl xlswrite('dateiname',variable) soll eine Datei im xls-Format speichern (funktioniert aber auf meinem Apple nicht), probieren Sie ihn auf den Windows-Rechnern aus, indem Sie ein große Matrix erzeugen, abspeichern und in Excel öffnen.

Man kann auch mit copy-paste direkt zwischen dem Array Editor in Matlab und Excel hin und her kopieren, allerdings muss man in Excel zunächst den Bereich markieren, in den eine Matrix kopiert werden soll. Probieren Sie auch das aus.

\* T6A6) Matlab kann im Prinzip sehr viele Formate importieren - dies per Hand zu tun ist aber oft anstrengend. Deshalb gibt es den "Matlab import wizard", um dabei zu helfen. Schauen Sie bei Interesse in der Hilfe nach, wie man ihn verwendet. Für eine Auflistung verschiedener Importfunktionen für verschiedene Fileformate (inklusive Sound und Filmen) suchen Sie bitte in der Hilfe nach dem Schlagwort "File Formats".

**B) Datenaufnahme**

Die meisten biologischen Daten ändern sich nicht schrittweise, sondern kontinuierlich (z.B. Ph-Wert, Körpertemperatur, Membranpotential einer Nervenzelle etc.). Wenn man solche Daten aufzeichnet, steht man vor dem Problem, das eine kontinuierliche Aufzeichnung grundsätzlich nicht möglich ist. Spätestens wenn ein Computer benutzt wird, um die Daten zu speichern, müssen sie diskretisiert werden - es ist nur in bestimmten Zeitschritten eine Aufzeichnung möglich und durch die begrenzte (wenn auch sehr hohe) Genauigkeit der Darstellung von Nachkommazahlen sind auch die Messwerte nicht zu 100% kontinuierlich messbar. Grundsätzlich gilt: je größer die Abtastrate ist (also je mehr Messungen pro Zeiteinheit vorgenommen werden), desto genauer ist die Messung - aber auch desto "kostspieliger", denn es fallen mehr Daten an, die gespeichert und verarbeitet werden müssen (und häufig wird die Messhardware deutlich teurer, wenn man hohe Abtastraten erreichen will.)

T6B1) Plotten Sie eine Parabel: x=-5:1:5, y=x.^2, plot(x,y)

Was fällt Ihnen an dieser Parabel auf? Machen Sie die einzelnen Punkte sichtbar.

Machen Sie die Schritte kleiner und plotten Sie die Parabel erneut.

Erstellen Sie einen gemeinsamen plot, bei dem die oben erzeugten Werte mit einzelnen Punkten und die Werte für x\_fein=-5:0.01:5 mit einer durchgezogenen Linie in eine Abbildung gezeichnet werden.

T6B2) Wenn eine kontinuierliche Größe gemessen wird, ist die Wahl der Abtastrate wichtig - zu geringe Abtastraten können zu grundlegend falschen Ergebnissen führen. Dieser Effekt soll mit folgendem Skript verdeutlicht werden:

[aliasing\_effekt.m]

**C) Zufallszahlen und Wahrscheinlichkeitsverteilungen**

Bevor wir uns mit der statistischen Auswertung von echten Messdaten beschäftigen, erzeugen wir zunächst einmal selber "Messdaten" mit Matlab, nämlich Zufallszahlen. Diese werden beispielsweise gebraucht, wenn man Experimente plant, in denen bestimmte Reize in zufälliger Reihenfolge präsentieren werden sollen. Eine weitere wichtige Anwendung von Zufallszahlen sind Simulationen biologischer Prozesse.

Künstlich erzeugte Zufallszahlen entstammen einer bestimmten Wahrscheinlichkeitsverteilung. Diese gibt die Wahrscheinlichkeit dafür an dass eine Zufallsvariable einen bestimmten Wert annimmt. Im Rahmen des Kurses werden wir nur zwei verschiedene Verteilungen betrachten: Die Gleichverteilung bei der alle möglichen Werte gleich häufig auf treten und die Normalverteilung (auch Gaußverteilung genannt), bei der die Wahrscheinlichkeit des Auftretens eines Erwartungswerts am höchsten ist und ein Wert umso unwahrscheinlicher auftritt, je weiter er vom Erwartungswert entfernt liegt.

Für Messdaten (auch künstlich vom Computer erzeugte Zufallszahlen) werden Häufigkeitsverteilungen empirisch ermittelt, um dadurch auf die zugrundeliegende Wahrscheinlichkeitsfunktion zu schließen. Dafür benutzt man Histogramme. Diese teilen den gesamten Wertebereich der Variablen in mehrere Stücke auf und geben an, wie häufig die Variable in einer Messung jeden Teilbereich des Wertebereichs angenommen hat. Dazu wird für jeden Teilbereich (sogenannte Klassen) ein Rechteck dargestellt, dessen Fläche die gemessene Häufigkeit repräsentiert.

T6C1) Zufallszahlen werden in Matlab mit zwei Funktionen erzeugt: rand und randn. Beide Funktionen bekommen als Eingaben die Dimensionen der Matrix mit Zufallszahlen, die sie erzeugen und zurückliefern sollen (M=randn(7,10)).

Erzeugen Sie sich einige Beispiele der Zufallszahlen: Was passiert, wenn man die Funktionen mehrfach hintereinander in gleicher Weise aufruft?

In welchen Bereich liegen die Werte für die beiden Funktionen?

Was könnte der Unterschied zwischen den beiden Funktionen sein?

T6C2) Erzeugen Sie sich jeweils einen sehr langen Vektor (z.B. 10000 Punkte) mit jeder der beiden Funktionen rand und randn und schauen Sie sich die jeweilige Verteilung der Zufallszahlen mit dem Befehl hist an.

Was sind die Unterschiede zwischen den beiden Verteilungen?

In seiner Standardversion teilt hist die Daten in 10 gleich grosse Bereiche ein, in denen jeweils die Anzahl der Datenpunkte gezählt werden. Beim Aufruf hist(v,n) teilt hist den Vektor v in n gleich große Bereiche ein. Sehen Sie sich die Verteilungen für verschiedene Werte von n an.

T6C3) Modifizieren Sie Ihre Zufallsvektoren, indem Sie diese

- mit verschiedenen Faktoren multiplizieren

- verschiedene Zahlen hinzuaddieren

Wie wirken sich diese Änderungen auf die Verteilungen aus?

T6C4) Eine weitere Funktion, um Zufallszahlen zu erzeugen, ist die Funktion randperm. v=randperm(n) liefert einen Vektor der ganzen Zahlen von 1 bis n in zufälliger Reihenfolge.

Schreiben Sie eine Funktion wuerfel, die Ihnen eine ganze Zahl zwischen 1 und 6 zurückgibt.

Benutzen Sie diese Funktion in einer weiteren Funktion wuerfel\_verteilung die als Eingabewert bekommt, wie oft gewürfelt wird und als Ausgabe die Verteilung der erzielten Würfelergebnisse zurückliefert.

Eine bequeme Methode, um Verteilungen zu ermitteln, ist ebenfalls die Funktion hist. Wird diese mit einem Ausgabeargument aufgerufen, z.B. h=hist(v,n) stellt sie Ihr Ergebnis nicht grafisch dar, sondern gibt es als Vektor zurück.

**D) Hausaufgaben:**

T6H1) Im Rahmen eines Elektrophysiologie-Praktikums wird ein Strompuls (in nA) in eine elektronische Modellzelle gegeben und deren Spannungsantwort (in mV) jeweils mit 1000 Punkten/Sekunde aufgezeichnet. Die entsprechenden Dateien sind [stimulus.mat] und [antwort.mat] (bzw stimulus\_v6.mat und antwort\_v6.mat).

Laden Sie die Dateien ein und plotten Sie sie übereinander in zwei subplots einer Abbildung.

Passen Sie die Zeitachsen in den Plots so an, dass Sekunden dargestellt werden. Beschriften Sie die Achsen

T6H2) Erstellen Sie ein Histogramm der mit der Modellzelle aufgenommenen Spannungswerte (es ist dabei sinnvoll, mehr als 10 Klassen zu verwenden). Was fällt dabei auf?

T6H3) Zu welchem exakten Zeitpunkt beginnt bei den gerade dargestellten Daten die Reaktion relativ zum Reiz? Suchen Sie den Zeitpunkt "per Auge" in der Graphik, indem Sie den entsprechenden Bereich vergrößern. Es kann helfen, den Darstellungsmodus so umzustellen, dass die einzelnen Datenpunkte dargestellt werden.

Was ist ein sinnvoller Schwellwert um diesen Zeitpunkt festzulegen? (Wie kann Ihnen die letzte Aufgabe dabei helfen?)

\*Suchen Sie auch den Zeitpunkt, wenn die Reaktion wieder endet.

\* T6H4) Etwas zum Ansehen für Grafik-LiebhaberInnen als Ergänzung zu den letzten Aufgaben: Man kann Reiz und Antwort auch in einem plot mit zwei verschieden skalierten y-Achsen darstellen. Wie das geht, sieht man in der Demo [doppelachsenplot.m].

Wer es wirklich kompliziert mag kann diesen Plot noch schöner machen, z.B. so [doppelachsenplot\_extra.m]

T6H5) Im Programm [vogelfang.m] werden drei verschiedene Arten von Zufallszahlen benutzt. Vollziehen Sie dieses Programm nach.

Nehmen Sie schrittweise folgende Änderungen vor:

a) Bei Amseln gibt es 60% Weibchen.

b) Bei Spatzen streut das Gewicht von Weibchen 3 mal mehr als das Gewicht von Männchen.

c) Es kommen 25% Meisen und 25% Spatzen in der Gegend vor.

\*T6H6) Die Funktion hist kann man auch auf Matrizen anwenden, dann wird jede Spalte als Histogramm dargestellt. Erzeugen Sie sich eine Matrix mit zwei gegeneinander verschobenen Verteilungen und stellen diese grafisch dar.

\*T6H7) Noch eine kleine Grafik-Übung: Nehmen wir an, sie haben ein paar besonders empfindliche Fische in ihrem Aquarium, in dem die Messreihe [phWerte.mat] vorgenommen wurde. Diese Fische vertragen den Bereich von pH 6.5 bis pH 7.5 gut, sind aber darüber und darunter gefährdet. Plotten Sie die phWerte so, dass sie die Messwerte innerhalb und außerhalb Toleranzbereichs sowie die Messfehler mit Symbolen in drei unterschiedlichen Farben darstellen.