Chapter 1

Debugging

60% of coding time is finding errors Famous last words

Es hilft ungemein, wenn zusammengehörige Skripte und Funktionen im gleichen Ordner auf der Festplatte zu finden sind. Es bietet sich also an, für jede Analyse einen eigenen Ordner anzulegen und in diesem die zugehörigen *m-files* abzulegen. Auf eine tiefere Schachtelung in weitere Unterordner kann in der Regel verzichtet werden. MATLAB© erzeugt einen "MATLAB" Ordner im eigenen `Documents` (Linux) oder `Eigene Dokumente` (Windows) Ordner. Es bietet sich an, diesen Ordner als Wurzelverzeichnis für eigene Arbeiten zu verwenden. Natürlich kann auch jeder andere Ort gewählt werden. In dem Beispiel in Abb. 1.1 wird innerhalb dieses Ordners für jedes Projekt ein eigener Unterordner erstellt, in welchem wiederum für jedes Problem, jede Analyse ein weiterer Unterodner erstellt wird. In diesen liegen sowohl die benötigten *m-files* also auch die Resultate der Analyse (Abbildungen, Daten-Dateien). Zu bemerken sind noch zwei weitere Dinge. Im Projektordner existiert ein Skript (analysis.m), das dazu gedacht ist, alle Analysen aufzurufen. Des Weiteren gibt es parallel zu den Projektordnern einen `functions`-Ordner in dem Funktionen liegen, die in mehr als einem Projekt oder einer Analyse gebraucht werden.

Beim Betrachten dieses Layouts fällt auf, dass es sehr wahrscheinlich ist, dass bestimmte Namen für Funktionen und Skripte mehrfach verwendet werden. Es ist nicht verwunderlich, wenn eine `load_data.m` Funktion in jeder Analyse vorkommt. In der Regel wird dies nicht zu Konflikten führen, da MATLAB© zuerst im aktuellen Ordner nach passenden Dateien sucht (mehr Information zum MATLAB©-Suchpfad in Box 1.1).

1.1 Namensgebung von Funktionen und Skripten

MATLAB© sucht Funktionen und Skripte ausschließlich anhand des Namens. Dabei spielt die Groß- und Kleinschreibung eine Rolle. Die beiden Dateien `test_funktion.m` und `Test_Funktion.m` zwei unterschiedliche Funktionen benennen können. Diese Art der Variation des Namens ist natürlich nicht sinnvoll. Sie trägt keine Information über den Unterschied der beiden Funktionen. Auch sagt der Name nahezu nichts über den Zweck der Funktion aus.

Die Namensgebung fällt mitunter nicht leicht — manchmal ist es sogar der schwierigste Aspekt des Programmierens! Ausdrucksstarke Namen zu finden lohnt sich aber. Ausdrucksstark bedeutet, dass sich aus dem Namen Rückschlüsse auf den Zweck ziehen lassen sollte.

Figure 1.1: Mögliche Organisation von Programmcode im Dateisystem. Für jedes Projekt werden Unterordner für die einzelnen Analysen angelegt. Auf Ebene des Projektes könnte es ein Skript (hier "analysis.m") geben, welches alle Analysen in den Unterordnern anstößt.

Info-Box 1.1: Der MATLAB© Suchpfad

Der Suchpfad definiert, wo MATLAB© nach Skripten und Funktionen sucht. Wird eine Funktion aufgerufen wird zunächst im aktuellen Arbeitsverzeichnis einem Treffer gesucht. Schlägt diese Suche fehl, so arbeitet sich MATLAB© durch den *Suchpfad* (siehe Abbildung). Der *Suchpfad* ist eine Liste von Ordnern in denen MATLAB© nach Funktionen und Skripten suchen soll. Die Suche nach der aufgerufenen Funktion wird dabei von oben nach unten durchgeführt. Das heisst, dass es bei Namensgleichheit eine Rolle spielen kann an welcher Stelle im Suchpfad der erste Treffer gefunden wird. Wichtig: MATLAB© sucht nicht rekursiv! Wenn die gewünschte Funktion in einem Unterordner des aktuellen Arbeitsverzeichnisses liegt, dieses aber nicht explizit im Suchpfad enthalten ist, so wird die Funktion nicht gefunden.

Der Suchpfad kann sowohl über die Kommandozeile mit dem Kommandos `addpath()` und `userpath()` als auchüber die in der Abbildung gezeigte GUI angezeigt und eingestellt werden. Die GUI erlaubt Ordner aus dem Suchpfad zu entfernen, neue Ordner (optional inklusive aller Unterordner) hinzuzufügen oder die Reihenfolge der Pfade zu verändern.

Zum Wechseln des aktuellen Arbeitsverzeichnisses wird das Kommando `cd` verwendet. `which` zeigt an, in welchem Pfad eine bestimmte Funktion gefunden wurde. Das aktuelle Areitsverzeichnis wird durch den Aufruf `pwd` auf der Kommandozeile ausgegeben.

Benennung von Funktionen und Skripten

Die Namen von Funktionen und Skripten sollten möglichst viel über die Funktionsweise oder den Zweck aussagen (`firingrates.m` statt `uebung.m`). Gute Namen für Funktionen und Skripte sind die beste Dokumentation.

In Namen verbietet MATLAB© verbietet Leerzeichen, Sonderzeichen und Umlaute. Namen dürfen auch nicht mit Zahlen anfangen. Es mach für die Namensgebung selbst keine weiteren Vorgaben. Allerdings folgt die Benennung der in MATLAB© vordefinierten Funktionen gewissen Mustern:

- Namen werden immer klein geschrieben.
- Es werden gerne Abkürzungen eingesetzt (z.B. `xcorr()` für die Kreuzkorrelation oder `repmat()` für "repeat matrix")
- Funktionen, die zwischen Formaten konvertieren sind immer nach dem Muster "format2format" (z.B. `num2str()` für die Konvertierung "number to string", Umwandlung eines numerischen Wertes in einen Text) benannt.

Benennung von Variablen

Die Namen von Variablen sollten möglichst viel über ihren Inhalt aussagen (`spike_count` statt `x`). Gute Namen für Variablen sind die beste Dokumentation.

Listing 1.1: Unübersichtliche Implementation des Random-walk.

```
1  num_runs = 10;
2  max_steps = 1000;
3
4  positions = zeros(max_steps, num_runs);
5
6  for run = 1:num_runs
7
8
9  for step = 2:max_steps
10
11 x = randn(1);
12 if x<0
13 positions(step, run)= positions(step-1, run)+1;
14
15
16 elseif x>0
17  positions(step,run)=positions(step-1,run)-1;
18  end
19 end
20 end
```

Listing 1.2: Übersichtliche Implementation des Random-walk.

```
num_runs = 10;
max_steps = 1000;
positions = zeros(max_steps, num_runs);

for run = 1:num_runs
    for step = 2:max_steps
        x = randn(1);
        if x < 0
            positions(step, run) = positions(step-1, run) + 1;
        elseif x > 0
            positions(step, run) = positions(step-1, run) - 1;
        end
    end
end
```