8 Matlab

Wir widmen uns diese Woche einem kleinen Intermezzo in der Welt von Matlab. Soweit wir gehört haben, verwenden die meisten von Ihnen bereits Matlab. Für alle Fälle hier trotzdem ein paar Anmerkungen zur Installation:

- Sie erhalten Matlab kostenlos bei IDES unter www.ides.ethz.ch. Ebenfalls finden Sie auf der IDES-Website die detailierten Erklärungen, wie Sie IDES als Netzwerklaufwerk in Ihr System einbinden.
- Nach Ihrer Bestellung von Matlab auf IDES, erhalten Sie eine Mail mit der entsprechenden Installationsanleitung und dem Lizenzschlüssel. Die Installation starten Sie auf den meisten Systemen direkt aus dem Netzwerklaufwerk heraus, es lohnt sich nicht, die Installationsdateien zuerst herunterzuladen.
- Falls Sie von zu Hause aus auf das IDES-Laufwerk zugreifen wollen, müssen Sie ETH VPN (AnyConnect) verwenden (www.vpn.ethz.ch). Wir empfehlen Ihnen aber die Installation auf Ihrem Notebook an der ETH vornehmen, so sparen Sie Zeit dank der schnellen Netzwerkverbindung.

8.1 Funktion erstellen

a) Erstellen Sie während der Bearbeitung der Aufgaben eine stichwortartige Liste von beobachteten Unterschieden zwischen Matlab und Java.

b) Schreiben Sie in Matlab eine Funktion, die folgenden Ausdruck berechnet:

$$f(x) = \begin{cases} x \cdot sin(1/x) & \text{für } x \neq 0 \\ 0 & \text{für } x = 0 \end{cases} \tag{1}$$

Unter **New** → **Function** erhalten Sie ein Gerüst für eine neue Funktion. Der Rückgabewert heisst standardmässig `output_args`, die Funktion `untitled` und der Parameter `input_args`. Sie können die Namen aber direkt in der Datei ändern. Die Fallunterscheidung können Sie mit einem `if-then-else`-Konstrukt realisieren. Wenn Sie z. B. nicht wissen, wie die `if`-Verzweigung in Matlab aussieht, hilft der Aufruf `» help if` weiter. Auf der `help`-Ausgabe finden Sie einen Link zur **Befehlsreferenz**, die stets auch Beispiele bereit hält. Wenn Sie bei der Verwendung der Matlab-Funktionen nicht sicher sind, hilft es die Funktionen im **Command Window** kurz separat auszuprobieren.

c) Speichern Sie Ihre Lösung als Datei `f.m`.

d) Testen Sie die Funktion im **Command Window**. Für die Eingabe `» f(1)` sollten Sie die Ausgabe `ans = 0.8415` erhalten.

e) Realisieren Sie die gleiche Funktion in Java als `public double f(double x) { ... }`. Die Sinusfunktion erhalten Sie in Java mittels: `Math.sin(1/x)`.

8.2 Integral

Schreiben Sie in Matlab eine Funktion `int_rechteck`, die das Integral einer beliebigen Funktion mit der Rechteckregel berechnet. Die zu integrierende Funktion f wird als **Function-Handle** übergeben. Signatur und Aufruf von `int_rechteck` sollten folgendermassen aussehen:

Listing 1: int_rechteck.m

```
function res = int_rechteck(f,a,b,steps)
    % TODO: ...
    % Integral von f im Interval [a,b]
    % ... mittels Rechteckregel
end
```

Listing 2: Command Window

```
f0 = @(x)x;
f1 = @(x)x.*x;
f2 = @(x)sin(x);

res0 = int_rechteck(f0,0,1,1000);
res1 = int_rechteck(f1,0,1,1000);
res2 = int_rechteck(f2,0,1,1000);
```

a) Wie Sie im Kopf unschwer nachrechnen können, sollte $res0 = 0.5$ betragen. Überprüfen Sie, ob Ihre Funktion auch auf diesen Wert kommt.

b) Vergleichen Sie Ihre Resultate mit dem Resultat der Funktion `integral(fun,xmin,xmax)`, welche in Matlab eingebaut ist.

c) Wiederholen Sie die Aufgabe unter Verwendung der Trapezregel anstelle der Rechteckregel. Vergleichen Sie die Resultate. Was stellen Sie fest?

8.3 Wettrennen

Freiwillig: Lassen Sie uns Matlab und Java noch ein bisschen vergleichen. Verwenden Sie die Funktion f aus Aufgabe 8.1. Um in Matlab den Funktionspointer zu f zu erhalten, verwenden Sie $@f$.

a) Zur Einstimmung: Können Sie das Integral von f im Interval von -0.4 bis 0.4 mittels der in Matlab eingebauten `integral`-Funktion berechnen? Wenn ja, was erhalten Sie?

b) Nutzen Sie nun in Matlab je einmal Ihre Funktionen mit Rechteckregel und Trapezregel für die gleiche Berechnung. Verwenden Sie je $100'000$ Stützwerte.

c) Schreiben Sie ein Matlab-Script (**New** → **Script**), dass mittels `tic ... toc` die Zeitdauer der Integrationen durch Rechteckregel und Trapezregel ermittelt.

d) Implementieren Sie die Rechteckregel und die Trapezregel auch in Java. Verwenden Sie dazu die gleiche Klasse, in der Sie bereits in der ersten Aufgabe die Funktion f implementiert haben. Auf diese Weise haben Sie in den Methoden direkten Zugriff auf die Funktion und Sie müssen sich nicht um Funktionspointer bemühen. ... OOP hat Vorteile.

e) Führen Sie in Java die gleichen Berechnungen wie in Matlab durch. In der Aufgabenstellung der vergangenen Aufgabe 4.3 g) sehen Sie wie Sie in Java die Ausführungszeit messen können.

f) Vergleichen Sie die erhaltenen Resultate und die Ausführungszeiten.

8.4 BubbleSort

Freiwillig: BubbleSort kam bereits in der freiwilligen Übung 5.3. Sie finden Erklärungen zum Algorithmus in den Slides zu Übung 5.

a) Schreiben Sie in Matlab ein Skript, welches einen Vektor mit 10 Zufallszahlen erstellt, diesen mit Bubble Sort absteigend sortiert und das Resultat ausgibt. Vektoren definieren Sie in Matlab so: `» vec = [6 7 8 9]`. Auf Vektorelemente greifen Sie so zu: `» vec(2)` ergibt `ans = 7`.

b) Lagern Sie nun den Code zum Sortieren in eine universelle Funktion `bubbleSort` aus, die den Vektor als Eingabeparameter entgegennimmt. Testen Sie Ihre Funktion. *Nebenbei: Wenn Sie Matlab-Statements wie z. B. Zuweisungen nicht mit einem Semikolon abschliessen, wird deren Resultat automatisch ausgegeben.*

c) Nachfolgend sind zwei mögliche Signaturen für die Funktion `bubbleSort`, sowie deren Anwendung abgebildet. Probieren Sie beide Varianten aus. Was bemerken Sie? Können Sie es begründen?

Listing 3: Test1.m

```
function bubbleSort2(vec)
    % Sortieren mit BubbleSort
end
```

Listing 4: Command Window

```
bubbleSort(v);
```

Viel Glück & Erfolg!

Listing 5: Test2.m

```
function res = bubbleSort(vec)
    % Sortieren mit BubbleSort
    res = vec;
end
```

Listing 6: Command Window

```
v = bubbleSort(v);
```