Informatik mit Matlab – Labor 6

Thema des Labors: Dateien, Excel, MAT-Files, Strings

Vorübung: Versuchen Sie die in der Vorlesung besprochenen Beispiele zum Laufen zu bringen. Die zu gehörigen M-Files finden Sie hier in diesem Verzeichnis.

Aufgabe 1: saveData(name, telnr)

Schreiben Sie die Funktion *saveData*, die die übergebenen Daten einer Person (*name* und *telnr*) in die Text-Datei 'p.txt' abspeichert, in der Form: '<name> hat Telefonnummer <telnr>.'. Schreiben Sie die weitere Funktion **readData()**, die die in der Datei 'p.txt' abgespeicherte Textzeile einliest und sie auf dem Bildschirm ausgibt.

Aufgabe 2: v = readVecFile(fn)

Die ASCII-Datei mit dem Namen *fn* enthält eine Reihe von Zahlen-Werten, also jeweils pro Zeile einen Text, der mittels *str2double* in eine reelle Zahl gewandelt werden muss. Erstellen Sie die Funktion *readVecFile*, die die Zahlen aus dieser Datei ausliest und damit den Vektor v aufbaut und zurückgibt. Klappt das Einlesen nicht, dann wird die leere Menge [] zurückgegeben.

Aufgabe 3: v = leseExcel(sK, z, datei)

Erstellen Sie die Funktion *leseExcel*, die aus einer Excel-Datei mit dem Namen *datei* in der Tabelle 1 den Text aus der Zelle zu Spaltenkenner *sK* (z.B. 'C') und Zeile *z* (z.B. Zahl 7) ausliest und in der Variablen *v* zurückgibt. .
Hinweis: Verwenden Sie zum Erstellen des Namens der Excel-Zelle, z.B. `C7´ aus dem Spaltenkenner *sK* (z.B. 'C') und der Zeile *z* (z.B. 7), die String-Funktion *sprintf*.
Die Funktion können Sie an der Excel-Datei *Module.xls* testen. Zum Auslesen von Text müssen Sie den Aufruf mit den erweiterten Rückgabewerten *[u,v] = xlsread(...)* verwenden.

Aufgabe 4: AmplResFnc(f0, delta, datei)

Erstellen Sie die Funktion **AmplResFnc**, die für eine Reihe von Werten für ω_e die Funktion $A(\omega_e)$ berechnet:

$$A = \frac{F_0}{m\sqrt{(\omega_0^2 - \omega_e^2)^2 + (2\delta\omega_e)^2}}$$

Mit $\omega_0 = 2\,\pi\,f_0$, δ = *delta*, F_0 = 1 N und m = 0.1 kg.
Definitionsbereich: ω_e = [1/10 * ω_o, 3 * ω_0], mit einer Schrittweite von 1/10 * ω_o
Schreiben Sie die berechneten Werte in die Excel-Datei zum Namen *datei*, in Spalte A die Frequenzen $f_e = \omega_e/2\pi$ und in Spalte B die Amplitudenwerte *A*.
Plotten Sie außerdem die Funktion $A(f_e)$, z.B. für f_0 = 2 Hz und *delta* = 1 /s.

Zusatzaufgabe: plotMAT(datei)

Schreiben Sie die Funktion **plotMAT(datei)**. Die Funktion liest den MAT-File zum Namen *datei* (z.B. *Absorp.mat*) ein. Der MAT-File enthält Daten in der Array-Variablen *D*, und zwar in der ersten Spalte von *D* Wellenlängenwerte *lambda* und in der zweiten Spalte Absorptionsgrößen *A*, d.h. *lambda* = *D*(:,1), *A* = *D*(:,2). Plotten Sie die Funktion *A*(*lambda*).

Zusatzaufgabe: CaesarCrypt(txtFile, cryptFile, incr)

Erweitern Sie die in der Vorlesung besprochene Verschlüsselungs-Funktion ***encrypt*** dahingegehend, dass Sie eine Textdatei *txtFile* einliest, alle Zeilen der Datei verschlüsselt und den verschlüsselten Text in die Datei *cryptFile* schreibt.
Verwenden Sie für die Verschlüsselung folgenden Algorithmus von Gaius Julius Cäsar:
Die Buchstaben werden um den Wert von *incr* im Alphabet verschoben, ähnlich wie in der Funktion *encrypt*. Wird jedoch das Ende des Alphabets erreicht, geht es von vorne wieder weiter, also z.B. ergibt eine Verschiebung von ‚Z' um 2 Positionen den Buchstaben ‚B'. Dabei muss man zwischen Groß- und Kleinbuchstaben unterscheiden.
Alle anderen Zeichen, z.B. das Leerzeichen ` ´, werden nicht verändert.
Mit der Funktion *CaesarCrypt* kann man die Verschlüsselung auch wieder rückgängig machen, indem man den entsprechenden negativen Wert für *incr* eingibt. Achten Sie deshalb darauf, dass man nach der Verschiebung vor den Buchstaben ‚A' gelangen kann und dann von hinten im Alphabet weitermachen muss.
Zum Test können Sie die kurze Textdatei *Caesar.txt* verwenden, die die ersten Sätze aus Cäsars Gallischem Krieg enthält.

Zusatzaufgabe: savePerson(pers, filename)

Ändern Sie die Funktion **printPerson** aus dem OOP-Labor so ab, dass die Daten der Personen (das übergebene Objekt-Array *pers* mit den Komponenten *name* und *telnr*) nicht auf dem Bildschirm ausgegeben, sondern in der Text-Datei zu dem übergebenen Namen *filename* abgespeichert werden.
Schreiben Sie die weitere Funktion **readPerson(filename)**, die die in der Datei zu *filename* abgespeicherten Daten einliest und zeilenweise auf dem Bildschirm ausgibt.