1 Prüfung - Applikationssoftware und Programmierung

- Erlaubt ist jegliche Benutzung Ihrer Unterlagen, Ihrer Übungsbeispiele und der Unterlagen am Web.
- Verboten ist während der Prüfung jedoch der Austausch von Files, E-mails und ähnlichem mit anderen Studierenden oder mit der Aussenwelt.
- Die Dauer des schriftlichen Teils sollte drei Stunden nicht überschreiten. Insgesamt stehen vier Stunden für die gesamte Abwicklung zur Verfügung.
- Anschließend an den schriftlichen Teil findet ein mündliches Einzelgespräch mit einer Dauer von ca 15 Minuten statt. Die Reihung erfolgt nach dem Zeitpunkt der Abgabe.
- Die Abgabe erfolgt mit Hilfe des Skripts `pruefungsabgabe`. Eventuell notwendige Daten bekommt man mit `pruefungsdaten`.
- Bitte geben Sie fertig gestellte Beispiele ab und programmieren Sie dann die weiteren. Dies erleichtert die Korrektur erheblich.
- Voraussetzung für die Teilnahme an der Prüfung war die rechtzeitige Abgabe der Übungsbeispiele. Falls Sie diese Voraussetzung nicht erfüllt haben, können Sie unter Vorbehalt mit der Prüfung beginnen. Ich werde den Stand der Abgabe in der Anfangsphase überprüfen und Ihnen dann die entgültige Entscheidung mitteilen.

2 Prüfung - Gruppe C

2.1 Bandmatrix

1. Schreiben Sie eine Matlab-Funktion `matrixdexp.m`, die mit folgendem Aufruf `r = matrixdexp(n,kl,ku,a)` eine Matrix `r` mit der Größe $(2n+1) \times (2n+1))$ erzeugt, die folgende Eigenschaften hat: Die Matrix `r` enthält ein Band aus Hauptdiagonale, `kl` Diagonalen unterhalb der Hauptdiagonale und `ku` Diagonalen oberhalb der Hauptdiagonale. In jeder dieser Diagonalen stehen jeweils gleiche

Werte, die durch die Formel `w=exp(-a*d)` berechnet werden, wobei `a` übergeben wird und `d` der Abstand von der Hauptdiagonalen ist. Alle restlichen Plätze sind mit Nullen belegt.

2. Hinweis: Erzeugen Sie zuerst eine Hilfsmatrix der Form (`n=3`)

$$h = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 0 & 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 \\ 4 & 3 & 2 & 1 & 0 & 1 & 2 \\ 5 & 4 & 3 & 2 & 1 & 0 & 1 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix} .$$

   Dazu eignet sich `meshgrid` und eine einfache Kombination der aus `meshgrid` erzeugten Matrizen. Die restliche Arbeit kann man durch Anwendung obiger Formel und mit Hilfe logischer Maskierung (logischer Indizierung) erreichen.

3. Vergeben Sie Defaultwerte `n=5`, `kl=2`, `ku=3` und `a=1`. Stellen Sie die Werte für `n`, `kl`, `ku` richtig, falls sie kleiner als Null oder größer als `2n` sind.

2.2 Fitten mit Inline-Funktionen

1. Zur Kurvenanpassung an Datenpunkte soll folgende Funktion verwendet werden

$$y(x) = \sum_{k=1}^{n} a_k f_k(x) \ ,$$

   wobei die $f_k(x)$ Inline-Funktionen sind, die in einer Zelle der Länge n gespeichert sind.
   Zur Erinnerung: Eine Zelle ist ein Matlab-Datentyp, der wie ein Array aufgebaut ist, aber an jeder Position verschiedenste Datentypen (daher auch Inline-Funktionen) enthalten kann. Ihre Länge oder Größe kann man mit `length` oder `size` abfragen. Erzeugt werden Zellen mit `c={f1,f2,f3}` und auf ihren Inhalt kann man z.B. mit `c{1}` zugreifen.

2. Erzeugen Sie eine Matlab-Skript `cinlinetest` in dem Sie x_d- und y_d-Datenwerte vom File `cinlinefit.dat` einlesen. Definieren Sie drei einfache Inline-Funktionen für $sin(x)$, $cos(x)$ und x^2 und speichern diese in einer Zelle.

3. Erzeugen Sie eine Matlab-Funktion, die mit dem Aufruf
   `[a,err2]=cinlinefit(xd,yd,c)`
   die Koeffizienten `a_k` und die Summe der Fehlerquadrate `err2` berechnet, wobei ihr die Daten x_d und y_d und die Zelle der Funktionen übergeben wird.

4. Hinweis: Erzeugen Sie in einer `for`-Schleife die Koeffizientenmatrix für das Gleichungssystem und lösen dieses anschließend mit dem geeigneten Inhomogenitätsvektor.

5. Schreiben Sie eine Matlab-Funktion, die mit dem Aufruf
   `y=cinlineeval(x,c,a)`
   die y-Werte für gegebene x-Werte berechnet, wobei in `c` die Inline-Funktionen übergeben werden und a die besten Fitparameter sind.

6. Stellen Sie im Skript die Daten und Ergebnisse sinnvoll graphisch dar und zeigen Sie, dass Sie mit Graphik-Handles umgehen können.

2.3 Finden von Minima/Maxima

1. Eine Funktion $y(t, a, b)$ ist gegeben durch

$$y = \sin(at^2)e^{-bt}$$

   mit den Nullstellen bei $t = \sqrt{k\pi/a}$. Durch die Wahl von k_{min} und k_{max} legt man damit den interessierenden Zeitbereich fest.

2. Schreiben Sie ein Matlab-Skript `scminmax` in dem Sie obige Funktion als Inline-Funktion definieren und in dem Sie als Testwerte $a = 1$, $b = 0.8$, $k_{min} = 0$ und $k_{max} = 10$ festlegen. Plotten Sie die Kurve und die Nullstellen.

3. Zwischen jeweils zwei Nullstellen findet sich nun ein Minimum bzw. ein Maximum. Zum Finden eignet sich die Funktion `fminbnd`, die erlaubt, dass man das Suchintervall (hier jeweils zwei benachbarte Nullstellen) angibt. (Im Unterschied dazu erlaubt `fminsearch` nur die Angabe eines Startwertes für die Suche.) Finden Sie damit Minima und Maxima der Funktion und stellen Sie diese im Plot dar.

4. Hinweis: Enthält ein Suchintervall bei der Suche nach einem Minimum kein Minimum sondern ein Maximum, dann liefert die Routine `fminbnd` eine der Intervallgrenzen.

3 Prüfung - Gruppe A

3.1 Bandmatrix mit Zufallszahlen

1. Schreiben Sie eine Matlab-Funktion `matrixrband.m`, die mit folgendem Aufruf `[r,c] = matrixrband(n,kl,ku,s)` eine Matrix `r` mit der Größe $(2n+1) \times (2n+1))$ erzeugt, die folgende Eigenschaften hat:
   Die Matrix `r` enthält ein Band aus Hauptdiagonale, `kl` Diagonalen unterhalb der Hauptdiagonale und `ku` Diagonalen oberhalb der Hauptdiagonale. All diese Diagonalen sollen normalverteilte Zufallszahlen $r = \sigma r_n$ enthalten, wobei σ im Parameter `s` übergeben wird und r_n die von Matlab berechneten Zufallszahlen sind. Alle restlichen Plätze sind mit Nullen belegt.

2. Hinweis: Verwenden Sie logische Maskierung (logische Indizierung) für die unterschiedlichen Regionen in der Matrix.

3. Vergeben Sie Defaultwerte `n=5`, `kl=2`, `ku=3` und `s=0.5`. Stellen Sie die Werte für `n`, `kl`, `ku` richtig, falls sie kleiner Null oder größer als `2n` sind.

4. Das Resultat `c` soll ein Vektor der Länge drei sein, der die Anzahl der Werte in der Matrix enthält, die kleiner, gleich, bzw. größer Null sind.

3.2 Pade-Fit

1. Pade-Fits sind Fits bei denen Datenpunkte durch folgende Funktion angenähert werden.

$$y(x) = \frac{p(x)}{q(x)} = \frac{\sum_{k=1}^{m+1} p_k x^{m+1-k}}{\sum_{k=1}^{n+1} q_k x^{n+1-k}} \ ,$$

   wobei m der Grad des Zählerpolynoms $p(x)$ und n der Grad des Nennerpolynoms $q(x)$ ist.

   Durch willkürliche Festlegung des Koffizienten $p_1 = 1$ kann man obige Gleichung umformen und erhält

$$\sum_{k=2}^{m+1} p_k x_d^{m+1-k} - \sum_{k=1}^{n+1} q_k y_d x_d^{n+1-k} = -x_d^m \ ,$$

wobei dies ein lineares Gleichungssystem für die $m + n + 1$ Koeffizienten $p_2 \ldots p_{m+1}$ und q_1 bis q_{n+1} ist. Die bekannten Werte sind die Daten x_d und y_d.

2. Erzeugen Sie eine Matlab-Skript `padetest` in dem Sie x_d- und y_d-Werte vom File `padefit.dat` einlesen. Legen Sie den Grad der Polynome p und q mit $m = 3$ und $n = 2$ fest.

3. Erzeugen Sie eine Matlab-Funktion, die mit dem Aufruf
   `[p,q]=padefit(xd,yd,m,n)`
   die Polynome p und q in Matlab-Darstellung berechnet.

4. Hinweis: Stellen Sie die $(\ell \times (m + n + 1))$-Koeffizienten-Matrix auf, wobei ℓ die Länge der Datenvektoren ist. Dafür eignen sich `for`-Schleifen. Lösen Sie das Gleichungssystem mit dem richtigen Inhomogenitätsvektor. Teilen Sie dann den Lösungsvektor richtig auf die Polynome p und q auf und vergessen Sie nicht auf den festgelegten Wert $p_1 = 1$.

5. Stellen Sie im Skript die Daten und Ergebnisse sinnvoll graphisch dar und zeigen Sie, dass Sie mit Graphik-Handles umgehen können. Zur Auswertung der Polynome kann man einfach `polyval` verwenden. Berechnen Sie auch die Nullstellen des Nennerpolynoms. Was passiert an diesen Stellen in der Graphik?

3.3 Bogenlänge

1. Die Bogenlänge einer Kurve in Parameterform

$$x = x(t), y = y(t), z = z(t) \ ,$$

   ist definitionsgemäß gleich

$$s = \int_{t_0}^{t_1} \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2} \ ,$$

   wobei gilt $x'(t) = dx/dt$.

2. Um die Aufgabe einfach genug zu machen, nehmen wir an, dass obige Funktionen immer Inline-Funktionen von `t` und einem zweiten Parameter `k` sind.

3. Schreiben Sie nun eine Matlab-Funktion, die mit dem Aufruf
   `s=inlinebogen(t0,t1,fdx,fdy,fdz,k)`
   die Bogenlänge berechnet, wobei `fdx`, `fdy` und `fdz` bereits die abgeleiteten Funktionen $x'(t,k)$, $y'(t,k)$, $z'(t,k)$ in Form von Inline-Funktionen. Extrahieren Sie jeweils den Formelteil und setzen Sie die Funktion zusammen, die Sie zum Integrieren brauchen. Damit führen Sie dann das Integral aus und geben das Resultat zurück.

4. Schreiben Sie ein Matlab-Skript `scinlinebogen`, das die Bogenlänge für einen Kreis mit Radius $k = 2$, $x(t,k) = k\cos(t)$, $y(t,k) = k\sin(t)$ und $z(t,k) = 0$ berechnet. Die Ableitungen sollten Sie selbst "händisch" zusammenbringen. Überlegen Sie, welche Inline-Funktion Sie für $z(t,k)$ schreiben müssen. (Sie muss gleich viele Nullen liefern, wie ihr t-Werte übergeben werden; sie hängt nicht von k ab, trotzdem muss man k aber in der Parameterliste belassen!).

5. Überprüfen Sie das Ergebniss durch Vergleich mit dem Umfang eines Kreises.

6. Berechnen Sie die Bogenlänge von $x(t,k) = cos(t)$, $y(t,k) = sin(kt)$ und $z(t,k) = cos(t/k)$ zwischen 0 und 6π. Die Ableitungen müssen Sie wieder am "Papier" durchführen. (Resultat für $k = 3$: ≈ 39.51).

4 Prüfung - Gruppe B

4.1 Blockmatrix mit Zufallszahlen

1. Schreiben Sie eine Matlab-Funktion `matrixrblock.m`, die mit folgendem Aufruf `[r,c] = matrixrblock(n,m,s)` eine Matrix `r` mit der Größe $(nm \times nm)$ erzeugt, die folgende Eigenschaften hat:
   Die Matrix `r` enthält schachbrettartige Blöcke der Größe $(m \times m)$ abwechselnd mit normalverteilten Zufallszahlen $r = \sigma r_n$ und Nullen. Der Wert für σ wird im Parameter `s` übergeben und r_n sind die von Matlab berechneten Zufallszahlen.

2. Hinweis: Verwenden Sie logische Maskierung (logische Indizierung) für die unterschiedlichen Regionen in der Matrix. Überlegen Sie dabei, was die kleinste Einheitszelle ist, die vervielfältigt werden kann. Passen Sie auch auf, dass nicht alle Blöcke die gleichen Zufallszahlen enthalten.

3. Vergeben Sie Defaultwerte `n=5`, `m=2` und `s=0.5`.

4. Das Resultat `c` soll ein Vektor der Länge drei sein, der die Anzahl der Werte in der Matrix enthält, die kleiner, gleich, bzw. größer Null sind.

4.2 Fit einer Exponentialfunktion

1. Gegeben ist eine Funktion

$$y = c_1 x e^{-c_2 x^2} ,$$

die auch eine nichtlineare Funktion eines der Koeffizienten c_i ist.

Durch Logarithmieren kann man folgende Gleichung erzeugen,

$$\hat{c}_1 - \hat{c}_2 x_d^2 = \log y_d - \log x_d ,$$

wobei gilt $\hat{c}_1 = \log c_1$ und $\hat{c}_2 = c_2$. Damit erhält man bei bekannten x_d- und y_d-Werten ein lineares Gleichungssystem für $\hat{c}_1$ und $\hat{c}_2$ aus dem man c_1 und c_2 berechnen kann.

2. Erzeugen Sie eine Matlab-Skript `exponentialfit` in dem Sie x_d- und y_d-Werte vom File `exponentialfit.dat` einlesen.

3. Führen Sie nun den linearen Fit, den nichtlinearen Fit und als Vergleich auch einen Fit mit `polyfit` durch (Grad 4).

4. Berechnen Sie für alle drei Fälle den Gesamtfehler als

$$e = \sqrt{\sum_{k=1}^{n} (f(x_d) - y_d)^2 / n}$$

und vergleichen Sie die Ergebnisse. Überlegen Sie, was $f(x_d)$ bei Verwendung von `polyfit` ist.

5. Stellen Sie im Skript die Daten und Ergebnisse sinnvoll graphisch dar und zeigen Sie, dass Sie mit Graphik-Handles umgehen können. Was fällt Ihnen an den Ergebnissen auf?

6. Berechnen Sie

$$A = \int_0^\infty f(x) dx$$

für alle drei Fälle. Worauf muss man bei ∞ aufpassen? Was fällt Ihnen hier auf?

4.3 Faltung

1. Als Faltung der Funktionen f und g bezeichnet man folgendes Integral,

$$F(\hat{x}, s) = \int_{-\infty}^{\infty} f(x)g(x, \hat{x}, s)dx \ ,$$

wobei hier als Funktion g die Gaussfunktion,

$$g(x, \hat{x}, s) = \frac{1}{s\sqrt{2\pi}} \exp(-\frac{(x-\hat{x})^2}{2s^2}) \ ,$$

verwendet wird.

2. Schreiben Sie ein Matlab-Skript `scfoldgauss`, in dem Sie die Inline-Funktion $f(x) = sign(x)$, den Wert $s = 0.5$ und einen Vektor $\hat{x}$ mit 50 Werten zwischen $-\pi$ und π definieren.

3. Schreiben Sie eine Matlab-Funktion, die mit dem Aufruf
   `F=foldgauss(xhut,f,s)`
   die Faltung für alle Werte von $\hat{x}$ berechnet, `f` wird als Inline-Funktion übergeben. Nach der Berechnung soll `F` dann die gleiche Größe wie `xhut` haben.

4. Hinweis: Erzeugen Sie den String für die Gaussfunktion, extrahieren Sie aus der Inline-Funktion `f` die Formel und setzen beide zusammen. Damit erzeugen Sie eine Inline-Funktion, die Sie zur Integration an `quadl` übergeben können. Führen Sie dann in einer `for`-Schleife für jeden Wert von $\hat{x}$ obiges Integral aus und speichern das Ergebnis in `F`. Da man nicht von $-\infty$ bis ∞ integrieren kann, wählt man folgende Integrationsgrenzen $x_{1,2} = \hat{x} \mp 5s$. Damit kann die Gaussfunktion vernünftig abgedeckt werden und das Integral von $-\infty$ bis ∞ wird mit hoher Genauigkeit berechnet.

5. Verwenden Sie nun im Skript die Funktion `foldgauss` und plotten Sie die ursprüngliche $sign$-Funktion und die "gefaltete" Funktion. Sie sollten eine Abflachung der steilen Kurve um Null herum sehen.