ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Kommunikationstechnik

Prof. Dr. H. Bölcskei
Sternwartstrasse 7
CH-8092 Zürich

PPS "Bits on Air"
1. Teil, Matlab-Tutorial
Vorbereitungsaufgaben
(Lösungsvorschläge)

Markus Gärtner, Samuel Brändle & Patrick Kuppinger

Revidierte Version vom 16. August 2016

1 Einleitung

Bei den hier aufgeführten Lösungen handelt es sich um Vorschläge, vielfach gibt es auch andere Lösungsmöglichkeiten.

2 Operationen auf Matrizen

2.1 Eingeben von Matrizen

- `[7:7:70]`
- `[17:2:33]'`
- `[140:-5:87]`

2.2 Zugriff auf einzelne Matrixelemente und Submatrizen

- Die Elemente werden Spalte für Spalte durchnumeriert.
- `C(3,2) = 12`
- `B(:,3) = 3`
- `C(3,2:4) = A(2,:)`
- `C(:,6) = A(1,:)'`
- `E` wird "umgedreht" und in `F` gespeichert.

2.3 Arithmetische Operationen

- Der Skalar wird zu jedem Element der Matrix addiert.
- `5*ones(9,6)`
- `A^2` multipliziert `A` mit sich selbst (Matrixmultiplikation). `A.^2` quadriert jedes Matrixelement einzeln. Generell bedeutet ein Punkt vor einem Operator, dass die Operation elementweise ausgeführt wird.
- `2.^[1:10]'`

3 Funktionen

3.1 Skalarfunktionen

- Die Antworten sind in der Hilfe zu finden.
- Der Nachkomma-Teil jedes Elementes von `H` muss ≥ 0.5 sein, damit `round` aufrundet (also dasselbe passiert wie bei `ceil`).
- Alle Elemente von `H` müssen ≤ 0 sein (oder es sind alles Integer und es wird gar nicht gerundet).

3.2 Vektorfunktionen

- Werden `max` zwei Matrizen gleicher Dimension übergeben, so vergleicht `Matlab` sie elementweise und wählt für die Ergebnismatrix jeweils das grössere der beiden Elemente aus.
- Wir wenden `sum` auf die Transponierte von `A` an und transponieren das Ergebnis erneut: `sum(A')'`
- `D = [D; sum(D)]` oder `D(5,:) = sum(D)`

3.3 Matrixfunktionen

- Einfach ausprobieren.
- Die Spalten von `U` sind die Eigenvektoren von `A` und in der Diagonale von `V` stehen die Eigenwerte von `A`. `U*V*inv(U)` ist gleich `A` (Diagonalisierung).

3.4 Zufallszahlen

- `700*rand(10,1)`
- `floor(21*rand(4,7))-10`
- `2*floor(50*rand(1,15))+1`

4 Vergleiche, Bedingungen und Schleifen

4.1 Vergleiche und Bedingungen

- `if A~=B` entspricht `if all(all(A~=B))`, d.h. der Ausdruck ist nur dann true, wenn *alle* Elemente sich unterscheiden. Für uns genügt es aber, wenn nur schon ein Element verschieden ist (denn dann ist ja `A` nicht mehr gleich `B`).
- `all(all(A<min(min(B))))`
- `any(any(A>=1.2*B))`

4.2 While- und for-Schleifen

- In `f` werden alle Fibonacci-Zahlen unter 1000 in aufsteigender Reihenfolge gespeichert.
- Die `for`-Schleife berechnet die Summe aller ganzen Zahlen von 1 bis 100. Dasselbe ohne Schleife: `s = sum([1:100])` (oder mit der Formel $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$).
- Die Lösung ohne Schleife ist deutlich schneller (auf unserem Rechner etwa Faktor 17). Verwendet man die oben angegebene Formel, so benötigt `Matlab` so wenig Zeit, dass die Messung null ergibt.

5 Graphische Darstellung von Matrizen

5.1 Plots

- Als drittes Argument kann ein String übergeben werden, der spezifiziert, wie gezeichnet werden soll (Details in der Hilfe). Übergibt man nur einen einzelnen Vektor, so werden dessen Elemente gegenüber ihren Indizes geplottet.
- ```
  x = [0:0.01:2*pi];
  y1 = sin(x);
  y2 = sin(2*x);
  y3 = sin(3*x);
  plot(x,y1,'b');
  hold on;
  plot(x,y2,'g');
  plot(x,y3,'r');
  ```
- Hier gibt die Hilfe Auskunft oder man kann's ausprobieren.
- `Matlab` rechnet numerisch und mit einer beschränkten Genauigkeit. $\sin^2(\alpha) + \cos^2(\alpha)$ ist also nie genau gleich eins. Die Frage ist, ob das Ergebnis genug nahe bei eins ist, so dass `Matlab` es mit seiner beschränkten Genauigkeit nicht von eins unterscheiden kann.
  ```

5.2 Histogramme

- ```
  W = 100*rand(1,1000);
  hist(sqrt(W));
  ```
- `rand` liefert *gleichverteilte* Zufallszahlen auf dem Intervall $[0, 1]$, während `randn` *standardnormalverteilte* Zufallszahlen liefert.

6 M-Files

6.1 Script-Files

- Hier müssen schlicht die Befehle aus der Lösung der genannten Aufgabe in einem M-File gespeichert werden. Wir zeigen noch eine etwas elegantere Lösungsmethode:
  ```
  x = [0:0.01:2*pi]';
  xv = x*[1 2 3];
  y = sin(xv);
  plot(x,y);
  ```

6.2 Function-Files

- Falls `randint` lediglich mit zwei Parametern aufgerufen wurde (`m` und `n`), so werden hier `a` und `b` auf ihre Standardwerte 0 und 9 gesetzt.
- ```
  function a = ntiefk(n,k)
  if n==0 | k==0
    a = 1;
  else
    a = factorial(n)/(factorial(k)*factorial(n-k));
  end
  ```
- Eine erste Möglichkeit mit einer Schleife:
  ```
  function b = binomdist(n,p)
  for k=[0:n]
    a(k+1) = nchoosek(n,k) * p^k * (1-p)^(n-k);
  end
  if nargout == 0
    stem([0:n],a);
  else
    b = a;
  end
  ```

  Und eine zweite Lösung ohne Schleifen:
  ```
  function b = binomdist(n,p)
  k = [1:n-1];
  bc = [1 factorial(n)./(cumprod(k).*fliplr(cumprod(k))) 1];
  ```

```
a = bc .* p.^[0 k n] .* (1-p).^[n fliplr(k) 0];
if nargout == 0
  stem([0:n],a);
else
  b = a;
end
```

Als dieses Tutorial entstand, war die Version 6 von `Matlab` aktuell. Die Funktion `factorial` sah damals so aus:

```
function p = factorial(n)
%FACTORIAL Factorial function.
%    FACTORIAL(N) is the product of all the integers
%    from 1 to N, i.e. prod(1:N). Since double precision
%    numbers only have about 15 digits, the answer is
%    only accurate for N <= 21. For larger N, the answer
%    will have the right magnitude, and is accurate for
%    the first 15 digits.
%
%    See also PROD.

%    Copyright 1984-2002 The MathWorks, Inc.
%    $Revision: 1.7 $

if (length(n)~=1) | (fix(n) ~= n) | (n < 0)
  error('MATLAB:factorial:NNotPositiveInteger', ...
        'N must be a positive integer.');
end

p = prod(1:n);
```

Wie sieht `factorial` heute aus? Was hat sich verändert? Wie könnte man `binomdist` nun programmieren?

7 mat-files

Folgende Erkenntnisse sollten aus der Ausführung des Codes geschlossen werden:

- der Befehl `load` schreibt immer alle Daten, die die .mat Datei enthält in die Variable mit dem Namen, der beim Speichern spezifiziert wurde.
- der Befehl `load` überschreibt auch vorhandene Variablen.
- Sowohl bei `save` als auch bei `load` können mit nur einem Parameter verwendet werden (`save('data.mat'), load('data.mat')`) um den ganzen Workspace zu speichern bzw. den gesammten Dateiinhalt zu laden. Meisst ist es jedoch sinnvoller mit den

weiteren Parametern anzugeben welche Variablen man gerne speichern/laden würde (`load('data.mat','A')`)