S3 – Reglerteknik
2005-04-06

Introduktion till MATLAB
2E1215

Exempelsamling

Note this exercise compendium is under development, and will be finalized during 2005. Please help us to improve the document by emailing comments and suggestions to `2e1215@s3.kth.se`.

Preface

This exercise compendium is being developed in an effort to give you, as participant of 2e1215, full flexibility in combining your course book of choice with relevant exercises.

Although we recommend the book "Matlab for Engineers Explained" by N. Bergman and F. Gustafsson, Springer-Verlag, ISBN 1852336978, we acknowledge that there are many alternative texts, some of which are free of charge. Links to a selection of free MATLAB tutorials on the Internet can be found on the course homepage http://www.s3.kth.se/courses/2E1215.

We have organized this exercise compendium in different chapters, each covering a particular aspect of MATLAB programming. The first exercise of every chapter is intended to give an intuitive introduction and can be treated without prior knowledge of the field. Detailed solutions are provided for each exercise. Reading instructions are provided in each chapter, and refer to the Matlab help text which is available via the menu item "Full Product Family Help" in the "Help" menu. Please note that the document "Getting started with Matlab" is also available in PDF format via the help system.

It is our intention that when you feel confident in solving these exercises, you should also be well prepared for the final exam and ready to use Matlab effectively in your studies and professional career.

Stockholm, January 2005
The Authors

1 Introduction

Required reading:

Contents/MATLAB/Getting Started/Introduction
Contents/MATLAB/Getting Started/Development Environment

Crash course in Matlab: Introduction

1.1

Execute the following commands and try to interpret the results

```
>> 5
>> 5 + 3
>> a = 10 - 4
>> a
>> b
>> b = a * 4
>> a * b
>> ans
>> [↑] [↑] [↑] [enter]
>> a [↑] [↑] [↑] [enter]
>> a = 2 * 6;
>> a
>> why
```

1.2

Try `help` to get a list of available help topics. Here, the first five might be most important.

- `>> help general` - to get a list of general methods

- `>> help ops` - gives a list of basic operators
- `>> help lang` - help on programming language constructs
- `>> help elmat` - provides a list of elementary matrix functions
- `>> help elfun` - provides a list of elementary math functions

Use `help FUNCTION` to get more information about a specific function or `lookfor KEYWORD` to search the available short descriptions of functions for this keyword.

1.3

Use the command `lookfor` to find a function that tests whether a given integer is prime or not.

1.4

Use the command `help` to find out how the command works and test whether the number 567827 is prime or not.

2 Matrix Manipulation

Required reading:

Contents/MATLAB/Getting Started/Manipulating Matrices

Crash course in Matlab: Arrays and Matrices

2.1

Execute the following commands and try to interpret the results

```
>> A = [1 2 3; 4 5 6; 7 8 9]
>> A(2,3)
>> A(1,:)
>> A(:,3)
>> A(2:3,1:2)
>> A([3 2],2)
>> A(2,:) = 0
>> A(:,2) = 0
>> A([1 3],[1 3]) = -1
>> A(2,:) = [1 2 3]
>> A([1 3],[1 3]) = [1 2;3 4]
>> A(:,1) = [1; 2]
```

2.2

Create the following matrices

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 2 \\ 3 & 3 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 2 \\ 3 & 4 \end{pmatrix}$$

Use `isequal` to verify that

- $A + B = B + A$

- $A * B \neq B * A$
- $(A + B) + C = A + (B + C)$
- $(A * B) * C = A * (B * C)$
- $A * (B + C) = A * B + A * C$
- $(A + B)^t = A^t + B^t$
- $(A * B)^t = B^t * A^t$
- $(A * C)^{-1} = C^{-1} * A^{-1}$

If the result does not match your expectation, try `a-b` instead of `isequal(a,b)` to compare the two matrices and try to explain your observations.

2.3

Let `x=[4 5 9 6]`.

(a) Subtract 3 from each element

(b) Add 11 to the odd index elements

(c) Compute the square root of each element

(d) Raise each element to the power of 3

2.4

Create the following matrix using the colon operator

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 7 & 9 \\ 1 & 1.25 & 1.5 & 1.75 & 2 \\ 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$$

2.5

Create the following matrix using the functions `ones` and `zeros`

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 2 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 2 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.6

Create the following matrix using the colon operator together with element-wise operations and the functions `cumsum`, `cumprod` and `mod`

$$C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 9 & 16 & 25 \\ 1 & 0.5 & 0.33 & 0.25 & 0.2 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 2 & 6 & 24 & 120 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

2.7

Create the 5×5 magic square `E = magic(5)`. Verify that the sum along each row, column and the main diagonal is the same.

2.8

Consider the matrix created in the preceding task and replace all numbers that are larger than 15 by zeros.

Hint. You can either use the `find` command or logical indexing.

2.9

The Emperor of India put the call out to his subjects that he wanted a new game invented and that there would be a reward for the inventor of the best one. An old man came to him with Chess - after showing him the game and looking at the other inventions CHESS was declared the winner. The Emperor was so exultant over the invention of chess that he offered the inventor anything he wanted in the kingdom. The inventor thought for a minute and then said, " One grain of rice, Your Majesty."

"Just one grain of rice?"

"Yes, just one grain of rice on the first square of the chess board, two grains of rice on the second square, four grains of rice on the third square," and so on. Each sqare got double the grains of rice that the last square had.

Create the 8×8 matrix representing the chessboard with the number of rice grains specified for each square, like

$$D = \begin{pmatrix} 1 & 2 & 4 & \dots & 64 & 128 \\ 256 & 512 & & \dots & & \vdots \\ \vdots & \vdots & & & \ddots & \end{pmatrix}$$

What are the numbers on the 4 squares in the middle of the chess board?

Hint: use the function `reshape` and set `format long` to get all important digits.

2.10

Use Matlab to solve the following set of equations

$$\begin{aligned} 7x_1 + 14x_2 - 6x_3 &= 95 \\ 12x_1 - 5x_2 + 9x_3 &= -50 \\ -5x_1 + 7x_2 + 15x_3 &= 145 \end{aligned}$$

Hint: This is a system of linear equations on the form $Ax = b$.

2.11

Plot a spiral originating from the origin that reaches a radius of 5 after 4 revolutions.

2.12

Plot the function

$$f(x) = \sin\left(\frac{1}{x}\right)$$

over the interval $[0.01, 0.1]$. Add the label `'x'` on the x-axis, the label `'1/sin(x)'` on the y-axis and the title `'A simple example'` to the plot.

How do you choose the x-values to make the plot look nice?

2.13

Determine the solutions to the equation

$$\sin(x) = 3\cos(x)$$

in the interval $x \in [-2\pi, 2\pi]$ by plotting the two functions

$$f_1(x) = \sin(x) \qquad\qquad f_2(x) = 3\cos(x)$$

in the same plot. Export the plot to a PDF file.

Hint. You can either use the command `help plot` to find out how `plot` can take multiple (x,y) vector pairs as input, or use the command `hold` to add plots to the current axis.

The `zoom` command can also be useful in finding the solutions with sufficient accuracy.

2.14

The Fourier series is a series representation of a periodic function in terms of sines and cosines. The Fourier series representation of the function

$$f(x) = \begin{cases} 1 & 0 < x < \pi \\ -1 & -\pi < x < 0 \end{cases}$$

is

$$\frac{4}{\pi}\left(\frac{\sin x}{1} + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \frac{\sin 7x}{7} + \cdots\right)$$

Plot on the same graph the function $f(x)$ and its series representation using 4, 8 and 12 terms.

Select individual line styles and colors for the four plots according to your own taste.

2.15

Recall the identity

$$e = \lim_{n\to\infty} r_n, \qquad\qquad r_n = \left(1 + \frac{1}{n}\right)^n$$

Make a standard log-log plot of $e - r_n$ for $n = 5, 10, 15, 20, \ldots, 500$. What does the log-log plot tell about the asymptotic behaviour of $e - r_n$.

2.16

To get an impression of other visualizations, type the following commands and try to interpret the results

```
>> figure
>> x = -2.9:0.2:2.9;
```

```
>> bar(x,exp(-x.*x));
>> figure
>> x = 0:0.25:10;
>> stairs(x,sin(x));
>> figure
>> x=-2:0.1:2;
>> y=erf(x);
>> e=rand(size(x))/10;
>> errorbar(x,y,e);
>> figure
>> r=rand(5,3);
>> subplot(1,2,1); bar(r, 'grouped');
>> subplot(1,2,2); bar(r, 'stacked');
>> figure
>> x=randn(200,1);
>> hist(x,15);
```

2.17

Create a vector `v` with 10 normally distributed elements using the function `randn`. Extract the positive elements in `v` using the following two techniques

(a) Use the `find` command to find the indices of the positive elements, and extract the elements of `v` in these entries

(b) Use logical indexing, *i.e.*, first extract the elements by first creating a vector with ones on the positions where `v` has positive elements and zeros on all others; then use this vector to index `v` and extract the positive elements.

3 Strings and Sorting

Required reading:

Contents/MATLAB/Getting Started/Programming with MATLAB/...
.../Other Data Structures/Characters and Text

Crash Course in Matlab: Advanced Data Structures/Strings

3.1

Execute the following commands and try to interpret the results

```
>> a = 5
>> b = '5'
>> double(a)
>> double(b)
>> text = 'Hello World'
>> text([1, 7])
>> ascii = double(text)
>> char(ascii)
>> length(text)
>> tworows = ['Peter'; 'Karin']
>> tworows = ['Anna'; 'Karin']
>> tworows = ['Anna '; 'Karin']
>> tworows = char('Anna','Karin')
>> double(tworows)
```

3.2

What are the ASCII codes representing the capital letters A–Z, the lower case letters a–z and the numerals 0–9?

3.3

Define the string `'2 times 3 equals 6'`

(a) Find all occurences of the letter 'e'

(b) Find all white spaces in the string.

(c) Remove all white spaces from the string

3.4

How many 'a' appear in the help text for the function `char`?

How many times does the word 'form' occur in the help text?

3.5

Create a random vector `vec` with 10 elements

(a) Sort the elements in ascending order (smallest first, largest last). Determine the index of the second smallest element of `vec`.

(b) Sort the elements of `vec` in descending order (largest first, smallest last).

3.6

Suppose that we represent a standard deck of playing cards by a vector `vec` containing one copy of each integer from 1 to 52.

(a) Show how you can use the commands `sort` and `rand` to "shuffle" `vec` by rearranging its contents in a random order.

(b) Use `lookfor` and `help` to search for a Matlab command that permutes the elements of a vector, and use this command to shuffle the cards.

3.7

The following table contains the names of students and their results on the MATLAB course:

```
>> tab = char('Adam 5','Bjorn U','Christian 4','David 4',...
              'Eva 3','Frodo 4','Gerry 5','Hans 3',...
              'Irene U','Johan 4','Karin 5')
```

Sort the table by grades.

3.8

Why does

```
>> char('2' + '5' - '3')
```

give the "correct" result

```
ans =
4
```

but

```
char('4' + '5')
```

gives the "wrong" result

```
ans =
c
```

4 Scripts and Functions

Required reading:

Contents/MATLAB/Getting Started/Programming with MATLAB

Crash course in Matlab: Scripts and Functions
Crash course in Matlab: More on Functions

4.1

Write a function `add.m` that takes two input arguments and returns the sum of those two.

4.2

Write a function `grade.m` that computes the correct grade for a given number of points. The grade is determined by the following rule:

$$\text{grade} = \begin{cases} 5 & \text{if } 90 \leq \text{points} \leq 100 \\ 4 & \text{if } 70 \leq \text{points} \leq 89 \\ 3 & \text{if } 50 \leq \text{points} \leq 69 \\ \text{'U'} & \text{otherwise} \end{cases}$$

You may assume that the given number of points is between 0 and 100.

4.3

Write a function `grade2.m` that does exactly the same as in the preceding task. `grade2.m` must not contain any `if`-clause - use `switch` instead.

4.4

Write a function `hello.m` that displays "Hello World" if no input argument is given. If a name is given as input, the function should display "Hello NAME" instead. If the user types `>> help hello` a help text on how to use this function should be displayed. The function does not have to return any value.

Hint: use `nargin` to determine whether or not an input was given.

4.5

Write a function `middle.m` that takes as input a vector of numbers. If it is called with one output as in `middle(vec)` or `result = middle(vec)` it should return the median of the numbers in the vector. If it is called with two outputs as in `[mean, med] = middle(vec)` it should return the mean value as the first output and the median as the second one.

Hint: use `nargout` to determine the number of outputs requested

4.6

Write a function `sortcolumns.m` that works similar to `sortrows`, but sorts the columns of a matrix.

4.7

Write a function `fundiff(f,x,y)` that takes the name of a function `f` (as a string), and two scalar values x and y as inputs and returns the difference between the corresponding function values, *i.e.*,

$$f(x) - f(y)$$

The following lines demonstrates how the function should work

```
>> f='sin'; x=1; y=2;
>> fundiff(f, x, y)
ans =
   -0.0678
```

Hint. Use the command `feval`.

4.8

Write a function `i=signchange(x)` that finds where the vector `x` changes sign.

The following lines demonstrates how the function should work

```
>> x=[-1 2 3 -5 -6 4 7];
>> signchange(x)
ans =
     2     4     6
```

4.9

Write a function `I=trap(f,a,b,n)` that implements the trapezoidal quadrature rule

$$\int_a^b f(x)\,dx \approx \frac{h}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)]$$

where $h = (b-a)/n$ and $x_k = a + kh$. Test your function on $\sin(x) + \cos(x)$ for $0 \le x \le \pi/3$.

The inputs to `trap` are `f`: the name of the MATLAB function (as a string) to integrate, the lower and upper integration limits a and b and the number of discretization points n, and returns an approximation of the integral.

Hint: use `feval`

4.10

Write a function `newton(f,df,x0,tol)` that implements Newton's method for root-finding on a scalar equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

The first input is the name (a string) of a function f and the second input is the name of a function that returns the derivative f'. The third input is an initial guess of the root. Continue the iteration until either $|f(x_{n+1})|$ or $|x_{n+1} - x_n|$ is less than `tol` or a maximum number of 100 iterations is reached.

4.11

Write a function `helper.m` that takes as inputs the name of a MATLAB function and one of the letters 'l', 'r' or 'c' and returns a matrix containing the first 20 words of the help text for the given function either left justified, right justified or centered.

4.12

If a data source produces symbol k with probability p_k, the *first-order entropy* of the source is defined as

$$H_1 = -\sum_k p_k \log_2 p_k$$

Essentially, H_1 is the average number of bits needed per symbol to encode a long message; i.e., it measures the amount of information content (and therefore the potential success of compression strategies). The value $H_1 = 0$ corresponds to a single symbol, while for M symbols of equal probability, $H_1 = \log_2 M$.

Write a function `[H,M]=entropy(v)` that computes the entropy of a vector v. The probabilities should be computed empirically, but counting occurrences of each unique symbol and dividing by the length of v.

Hint: the commands `find` and `unique` may be helpful.

4.13

One way to compute the exponential function e^x is to use its Taylor series expansion around $x = 0$. Unfortunately, many terms are required if $|x|$ is large. But a special property of the exponential is that $e^{2x} = (e^x)^2$. This leads to a *scaling and squaring* method: divide x by 2 repeatedly until $|x| < 1/2$, use a Taylor series (16 terms should be more than enough) and square the result repeatedly. Write a function `expss(x)` that does this. Test your function on the x values $-30, -3, 3, 30$.

Hint. The function `polyval` can be useful for evaluating the Taylor series.

4.14

Re-write the function `trap` so that it does not use any loops.

Hint. Set up a vector of all x values and evaluate `f` for it. Then set up a vector of quadrature weights and use an inner product.

4.15

Re-write the function `entropy` so that it does not use any loops.

Hint. The functions `sort`, `diff`, `find` and (perhaps) `sum` can be useful.

5 Example Tasks from old Exams

5.1

```
------------------------------------------------------------------------
Barbecue!
------------------------------------------------------------------------

You want to arrange a barbecue for which you have to buy sausages,
bread and beer. Beer is sold in packages of 6, 11 and 20 bottles
and sausages are sold in packages of 5, 8 and 14. The three
different available sizes of bread packages will be given as an
input argument. Is it possible to buy exactly x of each without
any leftovers?


Syntax: OkFlag = barbecue(x, b1, b2, b3)
        -------------------------

where OkFlag     is a Flag indicating if it is possible (1) or not (0)
                 to buy exactly x without leftovers
      x          is the number of beer, bread and sausages to buy
      b1, b2, b3 are the three available bread package sizes
                 with b1 < b2 < b3


Example 1:

x = 37;  b1 = 4;  b2 = 11;  b3 = 14
Is it possible to buy exactly 37 beer, bread and sausages without
leftovers?

Beer:     6 + 11 + 20 = 37   -> OK
Sausages: 5 + 4*8 = 37       -> OK
Bread:    3*4 + 11 + 14 = 37 -> OK

Answer: 1 (It is possible to buy exactly 37 of each without
leftovers)
```

```
Example 2:

x = 27;  b1 = 4;  b2 = 11;  b3 = 14
Is it possible to buy exactly 27 beer, bread and sausages without
leftovers?
Beer:     there is no way to buy exactly 27 bottles of beer
Sausages: 5 + 8 + 14 = 27 -> OK
Bread:    4*4 + 11 = 27   -> OK

Answer: 0 (It is not possible to buy exactly 27 of each)


Basic functionality:

>> OkFlag = barbecue(37,4,11,14)

 OkFlag =
          1


>> OkFlag = barbecue(27,4,11,14)

 OkFlag =
          0

A correct solution gives one point if it passes a set of random
input problems.

Hint: This problem originates from the so-called McNugget numbers

--------------------------------------------------------------------
```

Solutions 2

2.2

```
>> A = [1 2;3 4]
>> B = [2 2;3 3]
>> C = [2 2;3 4]
>> (A+B) - (B+A)
>> (A*B) - (B*A)
>> (A*B)*C - A*(B*C)
>> A*(B+C) - (A*B + A*C)
>> (A+B)' - (A'+B')
>> (A*B)' - B'*A'
>> (A*C)^(-1) - C^(-1)*A^(-1)
```

2.3

```
>> x = [4 5 9 6]
>> x = x - 3
>> x(mod(1:4,2)==1) = x(mod(1:4,2)==1) + 11
>> x = sqrt(x)
>> x = x.^3
```

2.4

```
>> A=[1:5;1:2:9;1:0.25:2;0.1:0.1:0.5;5:-1:1]
```

2.5

```
>> B=zeros(7)
>> B(2:6,2:6)=ones(5)
>> B(3:5,3:5)=ones(3)*2
```

```
>> B(4,4)=3
```

2.6

```
>> A = [1:5;...
        [1:5].^2;...
        1./[1:5];...
        cumsum(1:5);...
        cumprod(1:5);...
        mod(1:5,2)]
```

2.7

```
>> E = magic(5)
>> sum(E,1)
>> sum(E,2)
>> sum(diag(E))
```

2.8

```
>> E>15
>> E(E>15) = 0
>> find(E>15)
>> E(find(E>15)) = 0
```

2.9

```
>> CB = reshape(2.^[0:63],8,8)'
>> format long
>> CB(4:5,4:5)
```

gives

$$\begin{pmatrix} 134217728 & 268435456 \\ 34359738368 & 68719476736 \end{pmatrix}$$

grains of rice on the four squares in the middle.

2.10

```
>> A = [7 14 -6; 12 -5 9; -5 7 15]
>> b = [95; -50; 145]
>> x = A\b
```

2.11

¿¿ t = 0:0.01:2*pi*4; ¿¿ x = 5*t/(8*pi).*cos(t) ¿¿ y = 5*t/(8*pi).*sin(t) ¿¿ plot(x,y)

2.12

```
>> figure(1)
>> x = 0.01:0.001:0.1
>> plot(x,sin(1./x))
>> figure(2)
>> x = 0.1 ./ (1:0.01:10)
>> plot(x,sin(1./x))
```

2.13

```
>> x = -2*pi:0.01:2*pi
>> figure(1)
>> plot(x,sin(x),'r',x,3*cos(x),'b')
>> figure(2)
```

```
>> plot(x,sin(x),'r')
>> hold on
>> plot(x,3*cos(x),'b')
```

2.14

```
>> x = -pi:0.01:pi
>> fx = ones(size(x))
>> fx(find(x<0)) = -1
>> four = 4/pi * sin(x)
>> for cnt = 2:12
>>     c = 2*cnt -1
>>     four(cnt,:) = [four(cnt-1,:) + 4/pi * sin(c*x)/c]
>> end
>> plot(x,fx,'k')
>> hold on
>> plot(x,four(4,:),'r')
>> plot(x,four(8,:),'g')
>> plot(x,four(12,:),'b')
```

2.15

```
>> n = 5:5:500
>> r = (1 + 1./n).^n
>> loglog(n,exp(1)-r)
```

2.17

```
>> vec = randn(1,10)
>> pos = find(vec >= 0)
>> vec(pos)
>> help logical
>> pos = (vec > 0)
>> islogical(pos)
```

```
>> vec(pos)
```

Solutions 3

3.2

```
>> [char((32:255)'),' '*ones(224,2),num2str((32:255)')]
```

gives the ascii codes of all printable characters. Thus, we get

- A-Z: `char(65:90)`
- a-z: `char(97:122)`
- 0-9: `char(48:57)`

3.3

```
>> str = '2 times 3 equals 6'
>> find(str == 'e')
>> blank = find(str == ' ')
>> str(blank) = []
```

3.4

```
>> text = help('char')
>> sum(text == 'a')
>> sum(lower(text) == 'a')
>> length(strfind(text,'form'))
>> length(strfind(text,' form '))
```

3.5

```
>> vec = rand(1,10)
```

```
>> [val, ind] = sort(vec)
>> ind(2)
>> val(end:-1:1)
>> -sort(-vec)
```

3.6

```
>> help sort
>> rvec = rand(1,52)
>> [val, ind] = sort(rvec)
>> vec = ind
>> lookfor permute
>> lookfor permutation
>> help randperm
>> vec = randperm(52)
```

3.7

```
>> tab = char('Adam 5','Niklas U','Christian 4','David 4',...
              'Eva 3','Frodo 4','Gerry 5','Hans 3',...
              'Irene U','Johan 4','Karin 5')
>> tabr = strjust(tab,'right')
>> tabr(tabr(:,end)=='U',end) = '2'
>> [v, ind] = sort(-tabr(:,end))
>> tab(ind,:)
```

3.8

The numerals `0-9` have the ascii codes `48-57`, respectively. Thus, `'2' + '5' - '3'` is evaluated to `50 + 53 - 51 = 52` and `char(52)` returns the string `'4'`.

`'4' + '5'` is evaluated to `52 + 53 = 105`. As `105` is the ascii code for `'c'`, `char('4' + '5')` returns `'c'`.

Solutions 4

4.1

```
1: function [ output ] = add( arg1, arg2 )
2:
3: output = arg1 + arg2;
4: return
```

4.2

```
 1: function [ output ] = grade( points )
 2:
 3: if (points >= 90)
 4:   output = 5;
 5: elseif (points >= 70)
 6:   output = 4;
 7: elseif (points >= 50)
 8:   output = 3;
 9: else
10:   output = 'U';
11: end
12: return
```

```
1: function [ output ] = grade( points )
2:
3: output = floor((points+10)/20);
4: if (output < 3)
5:   output = 'U';
6: end
7: return
```

4.3

```
 1: function [ output ] = grade2( points )
 2:
 3: switch floor(points/10)
 4:   case {9,10}
 5:     output = 5;
 6:   case {7,8}
 7:     output = 4;
 8:   case {5,6}
 9:     output = 3;
10:   otherwise
11:     output = 'U';
12: end
13: return
```

4.4

```
 1: function [ ] = hello( name )
 2: %HELLO function
 3: %  usage: hello          to display 'Hello World'
 4: %         hello('NAME')  to display 'Hello NAME'
 5:
 6: if (nargin == 0)
 7:   disp('Hello World')
 8: else
 9:   disp(['Hello ' name])
10: end
11: return
```

4.5

```
1: function [ outp1, outp2 ] = middle( vec )
2:
3: if (nargout < 2)
```

```
4:   outp1 = median(vec);
5: else
6:   outp1 = mean(vec);
7:   outp2 = median(vec);
8: end
9: return
```

4.6

```
1: function [v, i] = sortcolumns(x,row)
2:   if (nargin==1)
3:     row=1;
4:   end
5:   [v,i] = sortrows(x',row);
6:   v=v';
7: return
```

4.7

```
1: function [ output ] = fundiff( f, x, y )
2:
3: output = feval(f,x) - feval(f,y);
4: return
```

4.8

```
1: function [ output ] = signchange( x )
2:
3: output = find(diff(x>=0))+1;
4: return
```

4.9

```
1: function [ output ] = trap( f, a, b, n )
2:
3: h = (b - a) / n;
4: output = 0;
5: for x = a:h:b;
6:     output = output + 2 * feval(f,x);
7: end
8: output = h/2 * (output - feval(f,a) - feval(f,b));
9: return
```

4.10

```
1: function [ output ] = newton( f, df, x0, tol )
2:
3: x(1)=x0;
4: x(2)=x(1)-feval(f,x(1))/feval(df,x(1));
5: cnt=2;
6: while ( ( (x(cnt) - x(cnt-1) > tol) ||
7:           (feval(f,x(cnt)) > tol) ) &&
8:         (cnt < 100) )
9:     x(cnt+1)=x(cnt)-feval(f,x(cnt))/feval(df,x(cnt));
10: end
11: return
```

4.11

```
 1: function [ output ] = helper( func, just )
 2:
 3: text = help(func);
 4: [word,remainer] = strtok(text);
 5: output = word;
 6: for i = 1:19
 7:   [word,remainer] = strtok(remainer);
```

```
 8:   output = strvcat(output,word);
 9: end;
10: switch just
11:   case 'l'
12:     justify = 'left';
13:   case 'c'
14:     justify = 'center';
15:   otherwise
16:     justify = 'right';
17: end
18: output = strjust(output,justify);
19: return
```

4.12

```
1: function [ output ] = entropy( str )
2:
3: unstr = unique(str);
4: output = 0;
5: for k=1:length(unstr)
6:     p_k = sum(str==unstr(k))/length(str);
7:     output = output - p_k*log2(p_k);
8: end
9: return
```

4.13

```
 1: function [ output ] = expss( x )
 2:
 3: cnt = 0;
 4: while (x > 0.5)
 5:     x = x/2;
 6:     cnt = cnt + 1;
 7: end
 8: output = 0;
 9: for i = 0:15
```

```
10:     output = output + x^i/factorial(i);
11: end
12: while (cnt > 0)
13:     output = output^2;
14:     cnt = cnt - 1;
15: end
16: return
```

4.14

```
1: function [ output ] = trap( f, a, b, n )
2:
3: h = (b - a) / n;
4: x = a:h:b;
5: output = h / 2 * (feval(f,x) * [1 ones(1,length(x)-2)*2 1]');
6: return
```

4.15

```
1: function [ output ] = entropy( str )
2:
3: vec = diff([find(diff(sort(str))>0) length(str)]);
4: p = [vec length(str)-sum(vec)]/length(str);
5: output = - log2(p)*p';
6: return
```

Solutions 5

5.1

Of course, there are many possible ways to solve this problem. The algorithm implemented below tries first to make up the whole number of e.g. beer just with the largest package size. If it did not succeed, it reduces the number of the smallest package size bought in the previous trial and distributes the freed quantity on smaller package sizes until it found a valid combination. The last combination tested is buying only small packages. If even this combination is not valid, the required number can not be bought without leftovers.

```
 1: function OkFlag = barbecue(x, b1, b2, b3)
 2:
 3: A = [6 11 20; 5 8 14; b1 b2 b3];
 4:   % the rows of A correspond to available
 5:   % packages sizes of beer, sausages and bread
 6:
 7: for line = 1:3
 8:     % for each foodstuff do
 9:   a = A(line,:);
10:
11:     % try to buy only large packages
12:   c=[0 0 ceil(x/a(3))];
13:     % c are the numbers of small, medium and large
14:     % packages bought
15:
16:   delta = x - c*a';
17:     % delta are the leftovers
18:
19:   while (delta ~= 0)
20:     % while there are leftovers
21:
22:     ind = find(c(2:end));
23:     % what was the minimum package size in the last
24:     % trial? (except small packages)
26:
27:     if isempty(ind)
```

```
28:       % if there where only small packages in the
29:       % last trial and even this combination was
30:       % not valid, the required number can not be
31:       % bought without leftovers
32:       OkFlag = 0;
33:       return
34:     end
35:
36:     c(ind(1)+1) = c(ind(1)+1) - 1;
37:       % otherwise, reduce the number of this
38:       % smallest package size ...
39:     c(1:ind(1)) = 0;
40:     for j = ind:-1:1
41:       delta = x - c*a';
42:       c(j) = floor(delta/a(j));
43:     end
44:       % and distribute the remaining quantity over
45:       % smaller packages
46:
47:     delta = x - c*a';
48:       % are there any leftovers?
49:   end
50:     % no? great! continue with the next foodstuff
51: end
52:   % valid combinations found for all three
53:   % great! return success
54: OkFlag = 1;
55: return
```