Introduction to Computers
20.10.2003–24.10.2003

Vita Rutka
e-mail: rutka@mathematik.uni-kl.de
internet: http://www.mathematik.uni-kl.de/˜rutka

Exercises for the MATLAB part

2 Solutions

2.1 Preparation

Solution 1
Drawing a so called *"flow chart"* is the first step in any kind of programming you are doing. In simpler cases you'll soon be doing it only in your mind, but for more difficult problems every proffesional programmer is doing that. It's not so important, which kind of notations you are using, rectangulars, ellipses or what else. Important is, that you *split the main task in several steps, where you exactly know, in which sequence actions must be done.* It is, you must have an *algorithm*.

Two very powerful tools in **any** programming language are *loops* and *switching between several possibilities.* And if you remember that computer is like a small child to whom everything must be told explicitly, step by step, you can do really a lot.

Programming is nothing but giving step-by-step instructions to computer.

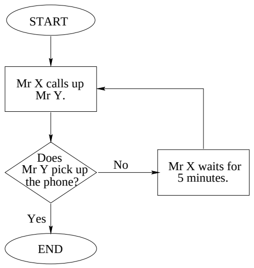

Figure 1: How Mr. X. calls Mr. Y. Note: here is loop.

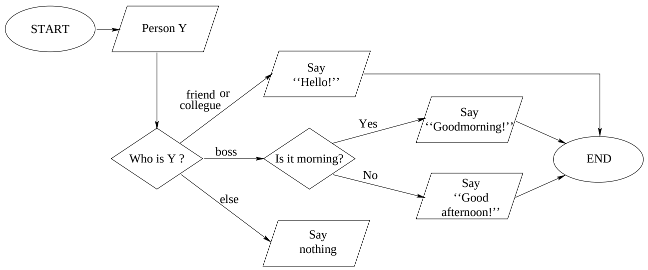

Figure 2: How Mr. X. decides what to say. Here another important concept of programming comes in: switching between several possibilities

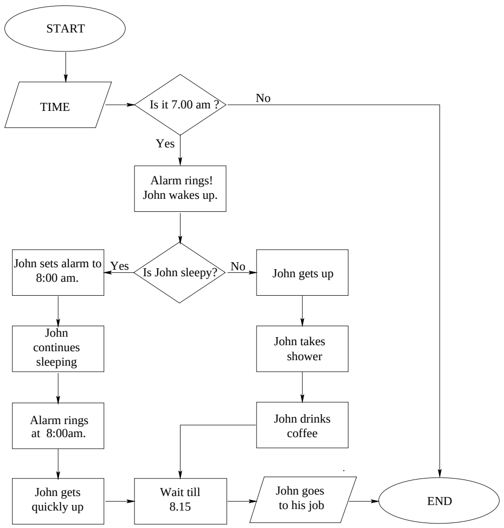

Figure 3: John's morning actions

2.2 Basics

If you use computer system of Mathematics Department, you can start MATLAB by typing **matlab** in 'xterm' or 'login' window. If you get error message, you can run matlab on computers called 'triton' or 'nereid'. (For this you need to make slogin session to one of these computers.)

If you get in some trouble, or you have runned the code you don't want, you cat try to terminate the process by pressing keys [Ctrl] and [C] in the same time.

Don't forget to type exit when you have finished your work!!!

Solution 2

```
>> x = [1 2 3 4]
x =
    1     2     3     4
>> sin(x)
ans =
    0.8415    0.9093    0.1411   -0.7568
>> 2*x
ans =
     2     4     6     8
>> 2+x
ans =
     3     4     5     6
>>
```

Solution 3

```
>> x=[4 5 6];
(Note: here we put ``;'' and result is not displayed)
>> y=[12 13 14];
>> x+y
ans =
    16    18    20
>> x'*y
ans =
    48    52    56
    60    65    70
```

```
    72    78    84
>> x*y'
ans =
   197
>> x.*y
ans =
    48    65    84
>>
```

NOTE: ".*" (as well as ./ and .∧) means *pointwise operation* in MATLAB. Mathematically:

$$x = (x_1\ ,\ x_2\ ,\ x_3)$$
$$y = (y_1\ ,\ y_2\ ,\ y_3)$$

Then, using 'prime' to denote transpose of vector, like it is in MATLAB, we get:

$$\begin{aligned} x * y &= \text{error message} \\ x * y' &= x_1y_1 + x_2y_2 + x_3y_3 \\ x' * y &= \text{matrix} \\ x. * y &= (x_1y_1\ ,\ x_2y_2\ ,\ x_3y_3) \\ x \wedge 2 &= \text{error message} \\ x. \wedge 2 &= (x_1^2\ ,\ x_2^2\ ,\ x_3^2) \end{aligned}$$

Solution 4

```
>> x1=4
 x1 =
      4
>> x2='4'
x2 =
4
>> 2*x1
ans =
     8
>> 2*x2
ans =
   104
>>
```

Logical question here would be: why such difference? Variable $x1$ is so called 'double', it is numerical. Variable $x2$ is 'string'. Thus 104 = 2*(ASCII code of symbol '4').

Solution 5

```
>> A = [2 3 4; 0 2 3; 6 8 13]
A =
     2     3     4
     0     2     3
     6     8    13
>> sin(A)
ans =
    0.9093    0.1411   -0.7568
         0    0.9093    0.1411
   -0.2794    0.9894    0.4202
>> exp(A)
ans =
   1.0e+05 *
    0.0001    0.0002    0.0005
    0.0000    0.0001    0.0002
    0.0040    0.0298    4.4241
>> expm(A)
ans =
   1.0e+07 *
    0.1602    0.2467    0.3886
    0.1041    0.1602    0.2525
    0.5049    0.7773    1.2247
>>
```

Normally, if argument is matrix (or even some more dimensional array!), MATLAB functions act to it "componentwise", i.e., if

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

then

$$\exp(A) = \begin{pmatrix} \exp(a_{11}) & \exp(a_{12}) \\ \exp(a_{21}) & \exp(a_{22}) \end{pmatrix}$$

There are also so called "matrix functions" (check help-system of MATLAB), and, e.g.,

$$\mathrm{expm}(A) = e^A$$

how it is really defined for functions with matrix argument.

Solution 6
Plotting of some functions is done in three steps:

1. Create points in which function will be computed; let's say, vector $\vec{x}$.
2. Calculate function values in these points; denote, e.g., by vector $\vec{y}$.
3. Plot vector $\vec{y}$ against vector $\vec{x}$.

```
>> x=linspace(-10,10,500);
>> y=sin(x)./x;
>> plot(x,y)
```

After it you should get something like figure 4:

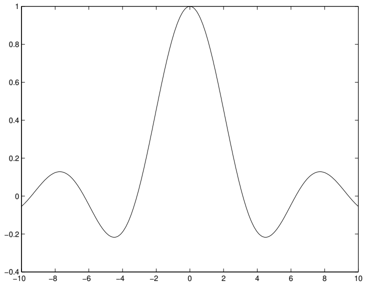

Figure 4: Graphics to solution 6

2.3 Script-files

Solution 7
See exercise.

Solution 8
Practically, here we use the same steps as in ex. 6. The only difference is, that now we write all steps in one file.

```
% Script file xsinx.m
% plot function x * sin(x) in the interval
% given by user. Number of grid points given by user
%
% Last changes: 19.10.2001

clc;
clear all;
close all;

% enter the interval
a = input('Enter the left end of the interval: ');
b = input('Enter the right end of the interval: ');

% number of grid points
n = input('Enter the number of grid points: ');

x = linspace(a,b,n); % create grid points
y = x.* sin(x);      % calculate function value

plot(x,y,'r')        % plot the function in red

% make the graphic easier to understand

title('x sin(x)')    % put the title
xlabel('x')          % label x axis
ylabel('y')          % label y axis
```

Solution 9

```
% script file write_hallo.m
% plot a lot of parametrized functions
% 24.09.03 by Vita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;

n = 200;
t = linspace(0,1,n)'; % work with column vectors

x1 = zeros(n,1);
y1 = 2*t;

x2 = 2*t;
y2 = ones(n,1);

x3 = 2*ones(n,1);
y3 = 2*t;

x4 = 2*t+2.5;
y4 = 2-2*abs(2*t-1);

x5 = 3+t;
y5 = ones(n,1);

x6 = 5*ones(n,1);
y6 = 2*t;

x7 = 5+t;
y7 = zeros(n,1);

x8 = 6.5*ones(n,1);
y8 = 2*t;

x9 = 6.5+t;
y9 = zeros(n,1);

x10 = 9+cos(2*pi*t);
```



```
y10 = 1+sin(2*pi*t);

figure
P=plot(x1,y1,'r',x2,y2,'r',x3,y3,'r',...
       x4,y4,'c',x5,y5,'c',...
       x6,y6,'m',x7,y7,'m',...
       x8,y8,'g',x9,y9,'g',...
       x10,y10,'b');
axis equal
set(P,'linewidth',2)
axis([-1 11 -1 3]);
```

Solution 10

```
% Script file myfactorial.m
% computes n!, where n is given by user
% actually, does the same as the built-in function factorial
% last changes: 19.10.2001

clc;            % clear MATLAB window
clear all;
close all;

n = input('Enter natural number: '); % asks user to enter the number

fact=1;         % initialize

for i=1:n       % run loop to find factorial
       fact = fact*i;
end

disp('Factorial of your number is ') % displays message
fact            % just prints computed value of n!


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% More optimal version of the same code:
```

```
disp('More optimal version:')

clear all;
close all;

n = input('Enter natural number: '); % asks user to enter the number

numbers = 1:1:n; % generate a VECTOR from 1 to n with step size 1

fact = prod(numbers); % multiply all entries of the vecor

disp(sprintf('Factorial of your number is %d',fact))

% (%d stands for the place of number)
```

As you can easily guess, factorial is often used mathematical command, so it would be really strange if there would be no built-in function for that. And there really is function **factorial(n)**, where instead of n is your number.

Solution 11

```
% plots the histogram of normally distributed random numbers
% 07.10.2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

n = 1000; % how many points to use

X = randn(n,1);

figure(1)
hist(X); % the default modus

figure(2)
N = hist(X,50); % use 50 intervals
bar(N,'r')
```

Solution 12

See exercise ...

Solution 13

```
% script file finite_precision.m
% demonstrate finite precision arithmetics
% 24.09.03 by Vita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 10000=sum_{k=1}^{100000} 0.1
my_sum = 0;
for k=1:100000
  my_sum = my_sum + 0.1;
end

disp('Example 1a, difference:')
disp(my_sum-10000);

% 10000=sum_{k=1}^{80000} 0.125
my_sum = 0;
for k=1:80000
  my_sum = my_sum + 0.125;
end

disp('Example 1b, difference:')
disp(my_sum-10000);


% Algebraically, f(x)=g(x)
x = 500;
f = x*(sqrt(x+1)-sqrt(x));
g = x/(sqrt(x+1)+sqrt(x));

disp('Example 2, difference:')
disp(f-g);
```

Solution 14

The “trick” in this exercise is to combine built-in MATLAB function **sum** and loop in degrees ...

```
% Script file mysums.m
```

```
% compute sum(k), sum(k^2) and sum(k^3), k=1...n
% where n is given by the user.
% Compare result with known formulas
% Last changes: 19.11.2001

% the "direct" version:
clc;
clear all;
n = input('Enter natural number: '); % asks user to give a number

sums = zeros(3,1); % before loop reserve the memory

% run a loop
for degree = 1:3   % loop over degrees
  for k = 1:n % loop over entries
    sums(degree) = sums(degree) + k^degree;
  end
end

% print result on the screen
sums

% compare with standard formulas and print the result on the screen

sumstandard =[ n*(n+1)/2 ; n*(n+1)*(2*n+1)/6 ; n^2*(n+1)^2/4]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% More optimal - remove the outer loop

clear all;
n = input('Enter natural number: '); % asks user to give a number

k = 1:n;            % that we sum up

sums = zeros(3,1); % i-th entry corresponds to sum(k^i)
```



```
for degree = 1:3
     sums(degree) = sum(k.^degree);
end

% print result on the screen
sums

% compare with standard formulas and print the result on the screen


sumstandard =[ n*(n+1)/2 ; n*(n+1)*(2*n+1)/6 ; n^2*(n+1)^2/4]
```

Here, you can get surprized, if you enter a large number, e.g., 1000. The result is returned as vector

```
sums =
   1.0e+11 *

    0.0000
    0.0033
    2.5050
```

The reason for "trouble" is that the last component is very large if compared with the first one. More usefull result you'll get, if you'll ask MATLAB to display vector component-by-component. In the same example you'll get the following results:

```
>> sums(1)
ans =
      500500
>> sums(2)
ans =
   333833500
>> sums(3)
ans =
   2.5050e+11
>>
```

Message: Results can be *displayed* differently!

Solution 15

```
% Script file PCW.m
% plots function
%        0      , x <= -1
% f(x) = x + 1  , -1 < x <= 0
%        -x + 1 , 0 < x <= 1
%        0      , else
% in the interval [-2 , 2]
% Last changes: 19.11.2001

clc;
clear all;
close all;

n = 100;                % number of grid points
x = linspace(-2,2,n);   % create grid points
y = zeros(n,1);         % reserve memory for function values

for i = 1:n             % run loop through all grid points
        if x(i)<=-1
                y(i) = 0;
        elseif (x(i)>-1)&(x(i)<=0) % if both conditions are satisfied
                y(i) = x(i)+1;
        elseif and((x(i)>0),(x(i)<=1))==1
                        % if both conditions are satisfied
                y(i) = -x(i)+1;
        else                                  % other cases
                y(i) = 0;
        end
end

figure % open a figure window
plot(x,y,'ro') % plot the function with red circles
title('Piecewise defined function') % add title
```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% More optimal version

clear all;
n = 100;                % number of grid points
x = linspace(-2,2,n);   % create grid points
y = zeros(n,1);         % reserve memory for function values

% find the domain where function has values x+1:
dom1    = find((x>-1)&(x<=0));
y(dom1) = x(dom1) + 1; % assign values in THIS DOMAIN

% find the domain where function has values -x + 1:
dom2    = find((x>0)&(x<=1));
y(dom2) = -x(dom2) + 1; % assign values in THIS DOMAIN

% in other parts zero is assigned by default

figure % open a figure window
plot(x,y,'go') % plot the function with green circles
title('Piecewise defined function') % add title
```

Solution 16

```
% Script file grades.m
% Asks for a grade and gives answer
%    very good       for 1.0 and 1.3
%    good            for 1.7, 2.0 and 2.3
%    acceptable      for 2.7, 3.0 and 3.3
%    passed          for 3.7 and 4.0
%    failed          for 4.3, 4.7 and 5.0
%    there exists    in other cases
%    no such grade
% Last changes: 19.11.2001

clc;
clear all;
```

```
continued = 'y'; % initialize

while continued=='y'
  answer = input(' Give a grade: ');

  switch answer
      case {1.0 , 1.3}
          disp(' very good ')
      case {1.7 , 2.0 , 2.3}
          disp(' good ')
      case {2.7 , 3.0 , 3.3}
          disp(' acceptable ')
      case {3.7 , 4.0}
          disp(' passed ')
      case {4.3 , 4.7 , 5.0}
          disp(' failed ')
      otherwise
          disp(' there exists no such grade ')
  end

  continued = input('Do you want to repeat your query? (y=yes)','s');
              % 's' assures string input
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The same could be done with if:

clc;
clear all;

continued = 'y'; % initialize

while continued=='y'
  answer = input(' Give a grade: ');
```



```
  if answer==1.0|answer==1.3
     disp(' very good ')
  elseif answer==1.7|answer==2.0|answer==2.3
     disp(' good ')
  elseif answer==2.7|answer==3.0|answer==3.3
     disp(' acceptable ')
  elseif answer==3.7|answer==4.0
     disp(' passed ')
  elseif answer==4.3|answer==4.7|answer==5.0
     disp(' failed ')
  else
     disp(' there exists no such grade ')
  end

  continued = input('Do you want to repeat your query? (y=yes)','s');
              % 's' assures string input
end
```

Solution 17

```
% script file my_cheb
% plot the first n Chebyshev polynomials
% of the first type, where n is given by the
% user
% 07.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

disp('Script file my_cheb')
disp('Plot the first n Chebyshev polynomials ')
disp('of the first type')
disp(' ')

n = input('  Enter n: ');


x = linspace(-1,1);

figure(1)
```

```
%hold on % uncomment if want all plots in the same window

for i = 1:n
  P = cos(i*acos(x));
  plot(x,P)
  pause
end
```

Solution 18

```
% script file bezier.m
% bezier curve for n points given
% by the mouse click
% 07.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

n = input('Enter the number of points: ');

figure(1)
axis([-1 1 -1 1])
grid on
hold on

% input of the points
% n+1 as indexation in the def. starts by zero
x = zeros(n+1,1);
y = zeros(n+1,1);

for i = 1:n+1
  [x(i),y(i)] = ginput(1);
  plot(x(i),y(i),'*')
end

% parameter
m = 200; % number of discretization points
t = linspace(0,1,m)';

C = zeros(m,2); % initialize
```



```
for i = 0:n
  % binomial coefficient
  Bcoeff = factorial(n)/factorial(n-i)/factorial(i);
  % Bernstein polynomial
  B = Bcoeff*t.^i.*(1-t).^(n-i);
  C(:,1) = C(:,1) + x(i+1)*B;
  C(:,2) = C(:,2) + y(i+1)*B;
end


% plot the result
plot(C(:,1),C(:,2),'r-')
```

Solution 19

```
% script file delta.m
% plots sequence of functions
%           1/n , -n/2 <= x <= n/2
% f_n(x) = 0    , else
% leading to delta-function in the limiting case
% last changes: 19.10.2001

% diplay first N approximations:

clear all; % clear all variables
close all; % close all figures

figure
hold on % put all approximations in the same figure

N = 10;     % how many approximations should be plotted

for n = 1:N
     % create the non-zero part
     x = [-1/2/n-1 -1/2/n -1/2/n  1/2/n 1/2/n 1/2/n+1];
         % as piece-wise constant, 2 points are enough
         % in each interval
     y = [0 0 n n 0 0];
```

```
     plot(x , y)
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% More clever - ask user if to continue ...

clear all; % clear all variables
close all; % close all figures

figure
hold on % put all approximations in the same figure

n = 1;     % initialize
answer = 'y';

while answer == 'y'
     % create the non-zero part
     x = [-1/2/n-1 -1/2/n -1/2/n  1/2/n 1/2/n 1/2/n+1];
         % as piece-wise constant, 2 points are enough
         % in each interval
     y = [0 0 n n 0 0];
     plot(x , y)

     % ask for continuing
     answer = input(' Continue? (yes = y, no = other) ','s');
     n = n + 1; % increase n by 1
end
```

After some steps you'll get something like figure 5

Solution 20

```
% script file my_series.m
% exponential series of different functions
% 24.09.03 by Vita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
```


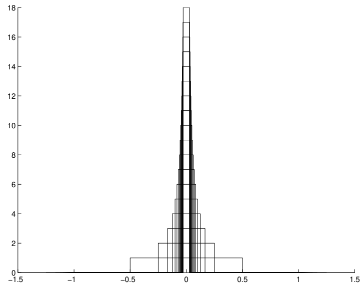

Figure 5: Graphics to solution 19

```
close all;

n = 20;
x = linspace(0,0.5);

% sin(x)
ser = zeros(size(x));
figure(1);
plot(x,sin(x),'r')
title('sin(x)')
hold on
for i = 0:n
  to_add = (-1)^i*x.^(2*i+1)/(factorial(2*i+1));
  ser = ser+to_add;
end
plot(x,ser,'b');

% cos(x)
ser = zeros(size(x));
figure(2);
plot(x,cos(x),'r')
title('cos(x)')
hold on
```

```
for i = 0:n
  to_add = (-1)^i*x.^(2*i)/(factorial(2*i));
  ser = ser+to_add;
end
plot(x,ser,'b');


% 1/(1-x)^3
ser = ones(size(x));
figure(3);
plot(x,(1-x).^(-3),'r')
title('cos(x)')
hold on
for i = 1:n
  to_add = (i+1)*(i+2)*x.^i/2;
  ser = ser+to_add;
end
plot(x,ser,'b');

% ln(1+x)
ser = zeros(size(x));
figure(4);
plot(x,log(1+x),'r')
title('ln(1+x)')
hold on
for i = 1:n
  to_add = (-1)^(i+1)*(x.^i)/i;
  ser = ser+to_add;
end
plot(x,ser,'b');
```

Solution 21

```
% script file kummer_sums
% check the convergence of relations
% which hold only asymptotically
% 06.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
```



```
N = 50; % up to which N to compute
x = pi/3; % at which point to compute


S1 = 0;
S2 = (pi-x)^2/4-pi^2/12; % initialize

figure(1)
hold on
disp('S1        S2        difference')
for n = 1:N
  S1 = S1 + (n^2+1)*cos(n*x)/(n^4+1);
  S2 = S2 + (n^2-1)*cos(n*x)/n^2/(n^4+1);
  plot(n,S1,'r*',n,S2,'bo')
  legend('S1','S2')
  disp(sprintf('% 6.6f % 6.6f % 6.6f',S1,S2,S1-S2));
end
```

Solution 22

```
% script file fourier.m
% plot truncated Fourier series of periodic functions
% last changes: 19.10.2001

clc;
clear all;
close all;

L = input(' Give a half of the period: ');
T = 2 * L;
phi = 2 * pi / T;
m = 100;  % numerical parameter, number of points in half period

% the whole time interval split in 4 parts:
t1 = linspace(-2 * L, -L, m); % interval [-2L;-L]
t2 = linspace(-L , 0 , m);    % interval [-L ; 0]
t3 = linspace(0 , L , m);     % interval [0  ; L]
t4 = linspace(L , 2 * L , m); % interval [L  ;2L]
```

```
t = [t1 t2 t3 t4];            % interval [-2L;2L]

% ==================================================================

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% That would be just to plot using first 5 terms, the essential
% part:

% Example 1

% plot the function
figure
hold on

f1 = (L - [t3 t4])/L; % compute only in ONE periodicity interval
plot([t1 t2],f1,[t3 t4],f1,'b') % plot the original function

a_0 = 0;

N = 5; % we will use first 5 terms

% just to be consistent with theory
f1_approx = a_0 * ones(1,length(t))/2; % initialize

for n = 1:N
    % find n-th Fourier coefficients
    a = 0;
    b = 2/pi/n;
    f1_approx = f1_approx + a * cos(n * phi * t) + b * sin(n * phi * t);
end

plot(t,f1_approx,'r')
grid on


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here we ask user about continuation

% Example 1
% plot the function
figure
hold on

f1 = (L - [t3 t4])/L; % compute only in ONE periodicity interval
plot([t1 t2],f1,[t3 t4],f1,'b')

answer = 'y';

a_0 = 0;

N = 0; % initialize
while answer == 'y'
    if N > 1
        plot(t,f1_approx,'w')
    end

    N = N + 5;
    % just to be consistent with theory
    f1_approx = a_0 * ones(1,length(t))/2; % initialize

    for n = 1:N
        % find n-th Fourier coefficients
        a = 0;
        b = 2/pi/n;
        f1_approx = f1_approx + a * cos(n * phi * t) + b * sin(n * phi * t);
    end

    plot(t,f1_approx,'r')
    grid on
    answer = input(' Continue? (yes=y) ','s');
end
% =====================================================================
```

```
% Example 2
% plot the function
figure
hold on

f2 = t3 .* (L - t3); % compute in one periodicity interval
plot(t1,f2,'b',t2,-f2,'b',t3,f2,'b',t4,-f2,'b')

N = 0;
answer = 'y';

a_0 = 0;

while answer == 'y'
    if N > 1
        plot(t,f2_approx,'w')
    end
    N = N + 5;
    % just to be consistent with theory
    f2_approx = a_0 * ones(1,length(t))/2;; % initialize

    for n = 1:N
        % find n-th Fourier coefficients
        a = 0;
        if mod(n,2) == 0 % if n is even
            b = 0;
        else
            b = 8 * L * L / pi / pi / pi / n / n / n;
        end
        f2_approx = f2_approx + a * cos(n * phi * t) + b * sin(n * phi * t);
    end

    plot(t,f2_approx,'r')
    grid on
    answer = input(' Continue? (yes = y) ','s');
end
```



```
% =====================================================================

% Example 3
% plot the function
figure
hold on

f3 = cos((pi * t3)/L); % compute in one periodicity interval
plot(t1,f3,'b',t2,f3,'b',t3,f3,'b',t4,f3,'b')

N = 0;
answer = 'y';
a_0 = 0;

while answer == 'y'
    if N > 1
        plot(t,f3_approx,'w')
    end

    N = N + 5;
    % just to be consistent with theory
    f3_approx = a_0 * ones(1,length(t))/2;; % initialize

    for n = 1:N
        % find n-th Fourier coefficients
        a = 0;
        if mod(n,2) == 1 % if n is odd
            b = 0;
        else
            b = 4 * n / pi /(n^2 - 1);
        end
        f3_approx = f3_approx + a * cos(n * phi * t) + b * sin(n * phi * t);
    end

    plot(t,f3_approx,'r')
    grid on
    answer = input(' Continue? (yes = y) ','s');
end
```

Solution 23

```
% Script file paramcurv.m
% Plot following families of parametrized curves:
%     - evolute of ellipse
%     - epicycloid
%     - folium of Descartes
%     - deltoid
%     - nephroid
% Last changes: 19.10.2001

clc;
clear all;
close all;

phi = linspace(0,2*pi,100); % parameter

% Evolute of ellipse:
figure     % prepare figure
hold on    % the whole family in one picture
A = 2;

for B = 0.5:0.1:1.5 ;
      x = A*cos(phi).^3;
      y = B*sin(phi).^3;
      plot(x,y)
end
title('Evolute of ellipse')
```



```
% Epicycloid:
figure
hold on
for a = 0.8:0.1:1.2
      for b = 0.8:0.1:1.2
            x = (a+b)*cos(phi)-b*cos((a+b)/b*phi);
            y = (a+b)*sin(phi)-b*sin((a+b)/b*phi);
            plot(x,y)
      end
end
title('Epicycloid')

% Folium of Descartes:
figure
hold on
for a = 0:2:10
      x = 3*a*phi./(1+phi.^3);
      y = 3*a*phi.^2./(1+phi.^3);
      plot(x,y)
end
title('Folium of Descartes')

% Deltoid:
figure
hold on
for a = 10:0.5:15
      x = 2*a*cos(phi)+a*cos(2*phi);
      y = 2*a*sin(phi)-a*sin(2*phi);
      plot(x,y)
end
title('Deltoid')

% Nephroid
figure
hold on
for a = 0.1:0.1:0.8
      x = a/2*(3*cos(phi)-cos(3*phi));
      y = a/2*(3*sin(phi)-sin(3*phi));
```

```
    plot(x,y)
end
title('Nephroid')
```

Solution 24

```
function y = my_fun(x)
% function y = my_fun(x)
% input: x: scalar or vector
% output: y = exp(sin(x))*(1-x) [scalar or vector]
%%%%%%%%%%%%%%%%%%%%%%%%%
y = exp(sin(x)).*(1-x);
```

Solution 25

```
% script file bisection.m
% bisection search for solving equation
% f(x)=0, where f(x) is calculated by function my_fun
% 24.09.03 by Vita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

a = -10;
b =  10;

tol = 1e-3; % tolerance for the solution
x = linspace(a,b); % just for visualization

while b-a>tol
  midpt = (a+b)/2; % midpoint
  midval = my_fun(midpt); % function value at the midpoint

  if midval==0;break;end; % the root is found

  if my_fun(a)*midval>0
    a = midpt;
  else
    b = midpt;
  end
```



```
end

x_sol = midpt;

disp('Solution:')
disp(x_sol)
disp('Function value at the numerical solution:')
disp(my_fun(x_sol))

% visualize the function
figure(1)
plot(x,my_fun(x),'b',x_sol,0,'r*');
grid on;
title('The function')
```

Solution 26

```
% script file fix_point.m
% fix point iteration for solving equation
% psi(x) = x, where psi(x) is given in function
% my_fixpoint_fun
% 24.09.03 by Vita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

a = 0; b = 1;
x = linspace(a,b); % only for visualization

figure(1)
subplot(2,1,1)
plot(x,my_fixpoint_fun(x)-x)
title('The original function')
grid on
subplot(2,1,2)
plot(x,my_fixpoint_fun(x),'b',x,x,'r')
title('Splitting for fixpoint iteration')
grid on
```

```
tol = 1e-3; % stopping tolerance

x0 = 0.2;
fval = my_fixpoint_fun(x0)-x0;

while abs(fval)>tol
  x_new = my_fixpoint_fun(x0);
  x0 = x_new;
  fval = my_fixpoint_fun(x0)-x0;
end

figure(2)
plot(x,my_fixpoint_fun(x)-x,'b',x0,0,'r*')
title('Solution')
grid on

disp('Solution:')
disp(x0)
disp('Function value at the numerical solution:')
disp(fval)


function y = my_fixpoint_fun(x)
% some function
%%%%%%%%%%%%%%%%%%%%%%%%%
y = 1-sqrt(x);
```

Solution 27

```
% some mandelbrot sets
% 06.10.03
%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all

Rmax = 5; % "infinity"

% domain to visualize
ax = -1; bx = 1;
ay = -1; by = 1;
```



```
% the paramter
%c = i; % dendrite fractal
%c = -0.123 + 0.745*i; % Douady's rabbit fractal
%c = -0.75; % San Marco Fractal
c = -0.391 - 0.587*i; % Siegel disc fractal

maxit = 100;

n = 40; % resolution
x = linspace(ax,bx,n); % real axis
y = linspace(ax,by,n); % imaginary axis

% iterate by loop
% can also vectorize!

iter = 0;

R = zeros(n,n); % initialize
% fill with value 2 if diverged

for ctr_x = 1:n
  for ctr_y = 1:n
    z = x(ctr_x) + i*y(ctr_y);
    iter = 0;
    while iter<maxit
      iter = iter+1;
      z = z^2+c;
      if abs(z)>Rmax
        R(ctr_x,ctr_y) = iter;
        break;
      end
    end
  end
end

contourf(x,y,R,maxit)
view(2)
```

```
shading flat
colorbar
```

2.4 “Real problems”

Solution 28

```
function S_new = bank(S,I,N)
% function S_new = bank(S,I)
%   S: amount you put in bank each month
%   I: interest rate
%   N: after how many monthes you would like to know
% 25.09.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

S_old = S; % initialize

for ctr = 1:N
  S_new = S_old*(1+I/12) + S;
  % reinitialize
  S_old = S_new;
end
```

Solution 29

```
% script file cardrivers.m
% approximate speed of car drivers with
% least squares approximation
% Last changes: 19.10.2001

clc;
clear all;
close all;

% Enter policeman's observations
x = [1  2  2.5 3 ]; % experience
y = [51 60 64  70]; % average speed

% Compute averaged values
x_bar = mean(x); % it is built-in function
```



```
y_bar = mean(y);

% find neccesary sums
S1 = sum((y-y_bar).*(x-x_bar));
S2 = sum((x-x_bar).^2);

% find coefficients of l.s. approximation
b = S1/S2;
a = y_bar - b*x_bar;


% create l.s. line
experience = linspace(1,24,10);
speed = a + b*experience;

% plot it
hold on
plot(x,y,'ro',experience,speed,'m')
legend('measurements','ls-approximation')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Add one more observation: practically repeat everything

x1 = [x 20 ]; % experience
y1 = [y 120]; % average speed

% Compute averaged values
x1_bar = mean(x1);
y1_bar = mean(y1);

% find neccesary sums
S11 = sum((y1-y1_bar).*(x1-x1_bar));
S21 = sum((x1-x1_bar).^2);

% find coefficients of l.s. approximation
b1 = S11/S21;
a1 = y1_bar - b1*x1_bar;
```

```
% create l.s. line
speed1 = a1 + b1*experience;
plot(x1(end),y1(end),'bo')  % plot last measurement
plot(experience,speed1,'b') % plot l.s. approximation
grid on % switch on the grid

xlabel(' Experience ')
ylabel(' Speed ')

hold off
```

Solution 30

```
% Script file signals.m
% filtering of incoming signal
% Last changes: 09.10.2000

clc;        % clear MATLAB window
clear all;  % clear all variables
close all;  % close opened figures

x = rand(100,1);  % simulate energy of incoming signal
y = zeros(100,1); % reserve memory for filtered signal

% start filtering

for i = 1:100
    if x(i)>=0.5  % if signal is strong enough
       y(i) = 1;
    else
       y(i) = 0;
    end
end

plot(1:100,x,'r*',1:100,y,'b*')
legend('incoming','filtrated')
```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% More optimal version

clc;        % clear MATLAB window
clear all;  % clear all variables
close all;  % close opened figures

x = rand(100,1);  % simulate energy of incoming signal
y = zeros(100,1); % reserve memory for filtered signal

% start filtering

strong = find(x>=0.5); % find where the signal is strong enough
y(strong) = 1;         % assign there value 1
                       % otherwise do not need to assign anything

plot(1:100,x,'r*',1:100,y,'b*')
legend('incoming','filtrated')
```

Solution 31

```
% Script file rdecay.m
% Radioactive decay of U; Process: U -> Ra
% described by dm/dt = -lambda*m, m - mass of material
%                                 lambda - decay rate
%                                 lambda = ln2*T_1/2
%                                 T_1/2 - half-life period
% Approximated solution using explicit Euler's method
% Last changes: 19.10.2001

clc;       % clear MATLAB window
clear all; % clear all variables
close all; % close all figures

time_interval = 5;     % min

% physical parameters
```

```
T = 23.5;               % half-life period; min
lambda = log(2)*T;      % decay rate

m_0 = 50;               % initial mass, kg


% numerical parameters
h = 0.05 ;              % step size, years
n = floor(time_interval/h+1);
                        % number of grid points


m = zeros(n,1);         % reserve memory for vector of masses
m(1) = m_0;             % initial value

% start Euler's iteration - the essential part

for i = 2:n
    m(i) = m(i-1)*(1-lambda*h);
end

% plotting

time = 0:h:time_interval;      % time steps;
                               % neccesary only for plotting
exact = exp(-lambda*time)*m_0; % exact solution

plot(time,m,'b',time,exact,'r')% plot approximated line in blue,
                               % exact in red
xlabel('time')
ylabel('mass')
title(' Radioactive decay of U ')
legend('approximated','exact')
```

Solution 32

```
% script file Town.m
% describes population of NN-town using model
% dp/dt = p/25 - p^2/(26*10^6)
```



```
% where p is the population
% known that in 1970 there were 8000000 inhabitants
% last changes: 19.10.2001 by Vita

clc;        % clear MATLAB window
clear all;  % clear all variables
close all;  % close all figures

n = 500;                           % number of time steps
t = linspace(1970 , 2001 , n + 1); % grid points are more!
p = zeros(1 , n + 1);              % reserve memory
h = t(2) - t(1);                   % time step

p(1) = 8000000;                    % initialize

for i = 1:n
    p(i + 1) = h * (p(i)/25 - p(i)^2/25*1e-6) + p(i);
end

plot(t,p)
title('Population')
xlabel('Year')

% display the answer
disp(sprintf(' In year 2001 there are approximately %10d inhabitants', ...
               floor(p(n+1)))) % of course, we round the result...
```

Solution 33

```
% scrip file rabbits.m
% example leading to Fibbonaci numbers:
% initially one rabbit pair; rabbits propogate twice and
% then die.
% Last changes: 09.10.2000

clc;
clear all;
close all;
```

```
A = [1 1 ; 1 0]; % system matrix

disp('Fibbonaci numbers: ')

x = [1 0]'; % initialize

% start iteration
for i = 1:20
    x = A * x;
    disp(x(1))
end
```

Solution 34

```
% script file flow.m
% compute movement of a particle in a given flow field
% in 2D. Velocity field given in file velocity.m
% 25.09.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;

ax = -1; bx = 1;
ay = -1; by = 1;

nx = 20; ny = 20;

x = linspace(ax,bx,nx);
y = linspace(ay,by,ny);

[X,Y] = ndgrid(x,y);

[V1,V2] = velocity(X,Y);

% visualize the velocity field
figure(1)
quiver(X,Y,V1,V2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```
% the real computation
% initial coordinate
x0 = 0.1; y0 = -0.1;

Tmax = 0.5; % until what time to compute
tau = 0.01; % time step

t = 0;
figure(2)
plot(x0,y0,'ro')
hold on
while t<Tmax
  t = t+tau;
  [v1,v2] = velocity(x0,y0);
  xnew = x0 + tau*v1;
  ynew = y0 + tau*v2;
  % plot the new value in red,
  % the old recolor in blue
  plot(xnew,ynew,'r.',x0,y0,'b.')
  axis equal
  grid on

  % update
  x0 = xnew;
  y0 = ynew;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a set of initial coordinates
x0 = rand(10,1)-0.5; y0 = rand(10,1)-0.5;

Tmax = 0.5; % until what time to compute
tau = 0.01; % time step

t = 0;
figure(3)
plot(x0,y0,'ro')
hold on
```

```
while t<Tmax
  t = t+tau;
  [v1,v2] = velocity(x0,y0);
  xnew = x0 + tau*v1;
  ynew = y0 + tau*v2;
  % plot the new value in red,
  % the old recolor in blue
  plot(xnew,ynew,'r.',x0,y0,'b.')
  axis equal
  grid on

  % update
  x0 = xnew;
  y0 = ynew;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a set of initial coordinates: movement of scaled unit circle
psi = linspace(0,2*pi,40);
x0 = 0.5*cos(psi);
y0 = 0.5*sin(psi);

Tmax = 0.5; % until what time to compute
tau = 0.01; % time step

t = 0;
figure(4)
plot(x0,y0,'ro')
hold on
while t<Tmax
  t = t+tau;
  [v1,v2] = velocity(x0,y0);
  xnew = x0 + tau*v1;
  ynew = y0 + tau*v2;
  % plot the new value in red,
  % the old recolor in blue
  plot(xnew,ynew,'r-',x0,y0,'b-')
  axis equal
```



```
  grid on

  % update
  x0 = xnew;
  y0 = ynew;
end

function [v1,v2] = velocity(x,y);
% function [v1,v2] = velocity(x,y);
% uncomment the necessary
% 25.09.03

% Example 1
%v1 = x; v2 = 5*y;

% Example 2
%v1 = x; v2 = -5*y;

% Example 3
%v1 = -x; v2 = -5*y;

% Example 4
%v1 = x; v2 = x + 0.5*y;

% Example 5
v1 = x; v2 = x - 0.5*y;
```

Solution 35

```
% script file decoder.m
% the code key is given in a file code_key.dat
% 07.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

message = ['With this string I want to test my decoder!'];

cod_mes = code_message(message,'code_key.dat');
```

```
orig_mes = decode_message(cod_mes,'code_key.dat');

disp('The original message:')
disp(message)
disp(' ')

disp('The coded message: ')
disp(cod_mes)
disp(' ')

disp('The back-coded message:')
disp(orig_mes)
disp(' ')


function cod_mes = code_message(message,code_file);
% function code_mes = code_message(message,code_file);
% reads the code from file code_file and
% converts a "readable" message into "unreadable" coded message
% 07.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% read the code key
code = []; % initialize
fid=fopen(code_file);
while 1
  line = fgetl(fid);
  if ~isstr(line), break, end
  code = [code;line];
end
fclose(fid);

% convert
L = length(code); % number of signs
key1 = code(:,1);
key2 = code(:,3);

% convert the message
```



```
MesL = length(message);

cod_mes = [];
for ctr = 1:MesL
  % go through all symbols
  orig=find(key1==upper(message(ctr)));
  if ~isempty(orig)
    % add the decoded symbol
    cod_mes = [cod_mes, key2(orig)];
  else
    % add the original symbol
    cod_mes = [cod_mes, message(ctr)];
  end
end

function orig_mes = decode_message(cod_mes,code_file)
% function orig_mes = decode_message(code_mes,code_file)
% converts "unreadable" message code_mes into a "readable"
% message orig_mes using code in file code_file
% 07.10.03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% read the code key
code = []; % initialize
fid=fopen(code_file);
while 1
  line = fgetl(fid);
  if ~isstr(line), break, end
  code = [code;line];
end
fclose(fid);

% convert
L = length(code); % number of signs
key1 = code(:,1);
key2 = code(:,3);
```

```
% and now code the message back!
cMesL = length(cod_mes);
orig_mes = [];

for ctr = 1:cMesL
  % go through all symbols
  key=find(key2==lower(cod_mes(ctr)));
  if ~isempty(key)
    % add the decoded symbol
    orig_mes = [orig_mes, key1(key)];
  else
    % add the original symbol
    orig_mes = [orig_mes, cod_mes(ctr)];
  end
end
```