7. SOLVING SYSTEMS OF NONLINEAR EQUATIONS. SOLVING OPTIMIZATION PROBLEMS

Objectives of the paper:

- Fixing knowledge regarding the solving of systems of nonlinear equations, both numerically and symbolically, using the Matlab programming environment,
- Fixing knowledge regarding the solving of optimization problems using the Matlab programming environment, by studying some examples and solving some problems.

It is recommended to go through Annex M7 before studying paragraphs 7.1 and 7.2.

7.1. Elementsregarding the solving of systems of nonlinear equations in Matlab. Elements regarding the solving of optimization problems in Matlab

Solving systems of nonlinear equations

Consider the following system of nonlinear equations (see annex M7):

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = c_1 \\ f_2(x_1, x_2, \cdots, x_n) = c_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_n(x_1, x_2, \cdots, x_n) = c_n \end{cases}$$

In order to solve this system with the help of the Matlab environment, it has to be brought to the ***canonical form***:

$$\begin{cases} g_1(x_1, x_2, \cdots, x_n) = 0 \\ g_2(x_1, x_2, \cdots, x_n) = 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ g_n(x_1, x_2, \cdots, x_n) = 0 \end{cases}$$

where $g_i = f_i - c_i, \quad i = 1,2,\cdots,n.$

A system of nonlinear equations can be solved in Matlab numerically or symbolically.

Solving a system of nonlinear equations numerically

In order to solve a system of nonlinear equations numerically, the steps of separation and approximate solving mentioned in annex M7 must be performed.In terms of the localization stage, in the case of systems of nonlinear equations with two unknowns:

$$\begin{cases} g_1(x_1, x_2) = 0 \\ g_2(x_1, x_2) = 0 \end{cases}$$

the graphic method can be used. For this purpose, the set of points that satisfy the equations $g_1(x_1, x_2) = 0$ and $g_2(x_1, x_2) = 0$ are graphically represented. The coordinates of the intersection points of the two graphs represent the solutions of the equation system. Reading them from the graph can be done with the Matlab *ginput* function presented in paragraph 6.1. Since the reading from the graph is made with a certain error, the coordinates read from the graph will be the values in the vicinity of which the solutions are located and will be used in the second stage as starting values in the determination of the solution.

The second stage aims at calculating the solution/solutions in the domain of interest with a predetermined precision.For calculating the solutions, it is necessary to define the functions g_i in a function file beforehand as components of a vector function.

For the calculation of each solution of the non-linear equation system, in the vicinity of a point resulting from separation, the Matlab *fsolve*f unction from the optimization toolbox(*Optimization Toolbox*) will be used. This function has multiple call syntaxes, analogous to the Matlab function *fzero* (seeparagraph6.1):

- If it is desired only to determine the solution, use one of the following call syntaxes:

```
x = fsolve(file_name,x0)
x = fsolve(file_name,x0,options)
```

- If, besides determining the solution, it is of interest, for reasons of accuracy, to also evaluate the functions g_i for the found solution, the function is used with one of the syntaxes:

```
[x,fval]= fsolve(file_name,x0)
[x,fval]= fsolve(file_name,x0,options)
```

- if the reason for stopping the execution of the numerical methodis of interest (solving with respect to imposed conditions, stopping by reaching the maximum number of iterations, etc), one of the following syntaxes should be used:

```
[x,fval,exitflag]= fsolve(file_name,x0)
[x,fval,exitflag]= fsolve(file_name,x0,options)
```

- if certain additional data is of interest, such as the number of performed iterations, one of the following syntaxes should be used:

```
[x,fval,exitflag,output]= fsolve(file_name,x0)
[x,fval,exitflag,output]= fsolve(file_name,x0,options)
```

where:

- *file_name* represents a string which contains the name of the function file in which the functions g_i were defined as components of a vector function;
- *x0* represents the vector of the approximate values of the solution sought;
- *options* represents a structure that contains optimization options for calculating the solution; is an optional argument; optimization options can be changed using the Matlab function *optimset* (see paragraph6.1);
- *x* represents the solution calculated with a predetermined precision, in the form of a vector;
- *fval* represents the values of the functions g_i for the calculated solution *x*;
- *exitflag* represents a control value, which specifies the reason for stopping the algorithm; for example, value 1 has the meaning that the algorithm has reached a solution under the required conditions;

- *output* represents a structure that contains the following information: the number of iterations (*iterations*), number of performed evaluations (*funcCount*), as well as the algorithms used to determine the solution (*algorithm*).

Solving a system of nonlinear equations symbolically

To solve nonlinear equation systems symbolically, the Matlab *solve* function from the *Symbolic Math Toolbox* should be used,having the syntaxes:

```
s=solve(eq_1,eq_2,...,eq_n)
s=solve(eq_1,eq_2,...,eq_n,var_1,var_2,...,var_n)
```

where:

- *eq_i* is the expression of the left member of equation *i* of the system, written between apostrophes;
- *var_1,var_2,...,var_n* are the unknowns of the system, written between apostrophes;
- *s* is a structure, having as members vectors of values; each vector corresponds to an unknown, having the same name as the unknown, and contains the values obtained for that unknown for each solution found.

The advantage of solving a system of nonlinear equations symbolically is that no set of starting values should be indicated in the solution's calculation and the value of the solution is accurately determined. The disadvantage is that few systems can be solved in this way.

Solving optimization problems

Because any maximization problem can be transformed into a minimization problem by changing the objective function sign, going forward only the problem of minimizing a real function of one or more real variables will be considered. Matlab provides more optimization problem solving features to the user, grouped into the *Optimization Toolbox*. These functions implement various numerical methods. Thus:

- the Matlab *fminbnd* function usesa combination of golden section search and parabolic interpolation methods;
- the Matlab *fminunc* function has implemented a combination of several methods, among which are the trust-region method, Quasi-Newton method, quadratic interpolation methods and cubic interpolation methods;
- the Matlab *fminsearch* function implements the simplex search method.

The Matlab *fminbnd* function searches for the local minimum of a real function with a real variable, located within an open interval. The call syntaxes of the *fminbnd* function are:

```
x = fminbnd(file_name,a,b)
x = fminbnd(file_name,a,b,options)
[x,fval]= fminbnd(file_name,a,b)
[x,fval]= fminbnd(file_name,a,b,options)
[x,fval,exitflag]=fminbnd(file_name,a,b)
[x,fval,exitflag]= fminbnd(file_name,a,b,options)
[x,fval,exitflag,output]= fminbnd(file_name,a,b)
[x,fval,exitflag,output]= fminbnd(file_name,a,b,options)
```

where:

- *file_name* represents a string which contains the name of the function file in which the expression of the objective function was defined;
- *a,b* represent the ends of the open interval in which the local minimum is searched;
- *options* represents a structure that contains optimization options for calculating the solution/ solutions; is an optional argument; optimization options can be changed using the Matlab function *optimset* (see paragraph6.1);
- *x* is the local minimum point;
- *fval* represents the function value at the local minimum point;
- *exitflag* represents a control value, which is 1 if the local minimum is found;
- *output* represents a structure that contains the following information: the number of iterations (*iterations*), number of performed evaluations (*funcCount*), the algorithms used to determine the solution (*algorithm*)and an ending message (*message*).

The Matlab *fminunc* and *fminsearch* functions are used to solve unrestricted minimization issues. The common call syntaxes of the two functions are:

```
x = Matlab_function(file_name,x0)
x = Matlab_function (file_name,x0,options)
[x,fval]= Matlab_function (file_name,x0)
[x,fval]= Matlab_function (file_name,x0,options)
[x,fval,exitflag]=Matlab_function (file_name,x0)
[x,fval,exitflag]= Matlab_function (file_name,x0,options)
[x,fval,exitflag,output]= Matlab_function (file_name,x0)
[x,fval,exitflag,output]= Matlab_function(file_name,x0,options)
```

where:

- *Matlab_function* is one of the functions: *fminunc*, *fminsearch;*
- *file_name* represents a string which contains the name of the function file in which the expression of the objective function was defined (with one or more variables);
- *x0* represents the initial value/ vector of the initial values from which the search of the minimum is started;
- *options* represents a structure that contains optimization options for calculating the solution/ solutions; is an optional argument; optimization options can be changed using the Matlab function *optimset* (see paragraph6.1);
- *x* is the local minimum point (vector);
- *fval* represents the function value at the local minimum point;
- *exitflag* represents a control value, which specifies the reason for stopping the algorithm; for example, in the case of the function *fminsearch* the value exitflag=1 shows that the algorithm has reached the local minimum;
- *output* represents a structure that contains the following information: the number of iterations (*iterations*), number of performed evaluations (*funcCount*), as well as the algorithms used to determine the solution (*algorithm*).

7.2. Examples

Example 7.1: Determine the solution for the following system of nonlinear equations:

$$\begin{cases} xy + z = -3 \\ \dfrac{x}{z} - y = -2 \\ yz + x = 6 \end{cases}$$

located in the vicinity of the point $x_0 = 1$, $y_0 = 0$, $z_0 = -1$.

Solution: Because the value at which the search for the solution is started is specified, it is no longer necessary to go through a solution localization stage; we can go directly to the solution's calculation stage. For this purpose,first the system of nonlinear equations is brought to canonical form:

$$\begin{cases} x \cdot y + z + 3 = 0 \\ \dfrac{x}{z} - y + 2 = 0 \\ y \cdot z + x - 6 = 0 \end{cases}$$

Then, the left system member is defined in canonical form in a function file (for example, `systnonlin1.m`):

```
function g=systnonlin1(v)
% x,y,z are represented by v(1),v(2),v(3)
x=v(1); y=v(2); z=v(3);
% the left member of the equation is represented by g(i)
g(1)=x*y+z+3;
g(2)=x/z-y+2;
g(3)=y*z+x-6;
```

After defining the left-hand member, the following Matlab program sequence is executed for calculating the solution (for example, script file):

```
% the vector of the initial approximation values of the solution
v0=[1; 0; -1];
% solving the system
sol=fsolve('systnonlin1',v0)
```

Following the execution of this sequence the following is obtained:

```
Optimization terminated: first-order optimality is less than
options.TolFun.
sol=      6.0000   -0.0000   -3.0000
```

It results that the solution is x=6, y=0, z=-3.

Example 7.2: Determine, starting from the values (1,2,3,4), the quadruple (a,b,c,d) which simultaneously satisfies the conditions:

i) a, b, c, d are in arithmetic progression,

ii) a-2, b-6, c-7, d-8 are in geometric progression,

with a precision of 10^{-30} both for the solution and the functions in the left system member written in canonical form.

Solution: The conditions i) and ii) are equivalent to the following system of nonlinear equations:

$$\begin{cases} a+c=2\cdot b \\ b+d=2\cdot c \\ (a-2)\cdot(c-7)=(b-6)^2 \\ (b-6)\cdot(d-8)=(c-7)^2 \end{cases}$$

whose canonical form is:

$$\begin{cases} a+c-2\cdot b=0 \\ b+d-2\cdot c=0 \\ (a-2)\cdot(c-7)-(b-6)^2=0 \\ (b-6)\cdot(d-8)-(c-7)^2=0 \end{cases}$$

Having the starting values for the solution determination, we go directly to the second step: the left member of the system of nonlinear equations is represented in a function file:

```
function g=systnonlin2(v)
% a,b,c,d are represented by v(1),v(2),v(3),v(4)
% left member of the system, written under the form of a
% vector function
g=[v(1)+v(3)-2*v(2);
 v(2)+v(4)-2*v(3);
 (v(1)-2)*(v(3)-7)-(v(2)-6)^2;
 (v(2)-6)*(v(4)-8)-(v(3)-7)^2];
```

The following Matlab program sequence is executed in order to solve the system (for example, script file):

```
% the vector of the initial approximation values of the solution
v0=[1; 2; 3; 4];
% setting the requested properties
options=optimset('TolX',10^(-30),'TolFun',10^(-30));
% solving the system
[sol,gval]=fsolve('systnonlin2',v0,options)
```

and the following results are obtained:

```
Optimization terminated: relative function value changing by less
than max(options.TolFun^2,eps) and sum-of-squares of function values
is less than sqrt(options.TolFun).
```

```
sol=       4.9999    6.0000    7.0000    8.0000
gval=    1.0e-008*
-0.0001         0.0000   -0.1595   -0.1599
```

Therefore, the obtained solution is: a=4.9999 (through verification results that the exact value is a=5), b=6, c=7, d=8.

Example 7.3: Determine symbolicallyall the triplets (a,b,c) that simultaneously satisfy the conditions:

i) a, b, c are in arithmetic progression,

ii) their sum is 30,

iii) a-5, b-4, c are in geometric progression,

Solution: The following system of nonlinear equations written in canonic form corresponds to the conditions: i), ii) and iii):

$$\begin{cases} a+c-2\cdot b=0 \\ a+b+c-30=0 \\ (a-5)\cdot c-(b-4)^2=0 \end{cases}$$

It is easy to notice that the first two equations are of the first degree and the third equation is of second degree only in relation to b, so the system has at most two solutions.

For the symbolic solving of this system, the Matlab `solve` function is used which receives as arguments the expressions of the functions in the left system member and the variables written between apostrophes. The following Matlab program sequence is executed (for example, script file):

```
sol=solve('a+c-2*b','a+b+c-30',...
    '(a-5)*c-(b-4)^2','a','b','c')
% displaying the solutions
for i=1:length(sol.a)
    disp([sol.a(i) sol.b(i) sol.c(i)])
end
```

After executing the sequence above the following results are obtained:

```
 sol =
     a: [2x1 sym]
     b: [2x1 sym]
     c: [2x1 sym]
 [17, 10, 3]
 [8, 10,12]
```

Two solutions were obtained: the triplets (a,b,c)=(17,10,3) and (a,b,c)=(8,10,12).

Example 7.4: Solve numerically the system of nonlinear equations:

$$\begin{cases} x^2 + y^2 = 2 \\ x^2 - y^2 = 1 \end{cases}$$

Solution: Note that the first equation of the system is the implicit equation of the circle with the center in the origin of the coordinate system, (0,0), and a radius $\sqrt{2}$, and the second equation is the implicit equation of a equilateral hyperbola with semi-axesof 1.

1. To locate the solutions of the system of nonlinear equations, the graphical method is applied see paragraph 7.1): the two geometric figures are graphically represented and the coordinates of the intersection points are "read" with the mouse. For this purpose, the following Matlab sequence is executed (for example,script file):

```
% circle with the centre (0,0) and radius sqrt(2)
theta=0:pi/60:2*pi; r=sqrt(2);
x=r*cos(theta); y=r*sin(theta);
plot(x,y,'r--')
hold on
% equilateral hyperbolawith the centre (0,0) and semi-axes 1
% y^2=x^2-1 => x^2>=1, so x<=-1 or x>=1,
% and y1=sqrt(x^2-1), y2=-sqrt(x^2-1)=-y1
x1=-3:0.1:-1; y11=sqrt(x1.^2-1); y12=-y11;
x2=sort(-x1); y21=sqrt(x2.^2-1); y22=-y21;
plot(x1,y11,'b',x1,y12,'b',x2,y21,'b',x2,y22,'b')
axis equal, grid, hold off
[xcoord,ycoord]=ginput;
```

The graph is shown in the figure 7.1.

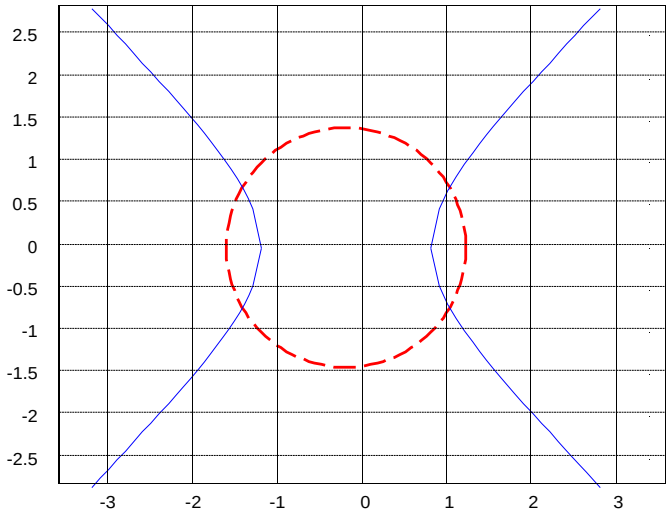

Fig.7.1. *The graph of the geometric figures in example 7.4.*

The graph shows that there are four intersection points, so the system has 4 solutions.

2. The second step - solution computing - begins with bringing the system into canonical form and defining the left system member in a function file:

```
function y=systNonlin(v)
x=v(1); y=v(2);
y=[x.^2+y.^2-2; x.^2-y.^2-1];
```

To determine the solutions, the Matlab *fsolve* function is used as many times as the number of found intersection points, each time having as starting values a set of "read" coordinates. For this purpose, the source code in paragraph 1 is to be completed with the following instructions:

```
disp('system solutions')
for i=1:length(xcoord)
    sol=fsolve('systNonlin',[xcoord(i),ycoord(i)])
    if i<length(xcoord) pause,end
end
```

The solutions are:

```
system solutions
Optimization   terminated:   first-order   optimality   is   less
than options.TolFun.
sol=     -1.2247    0.7071
Optimization   terminated:   first-order   optimality   is   less
than options.TolFun.
sol=      1.2247    0.7071
Optimization   terminated:   first-order   optimality   is   less
than options.TolFun.
sol=     -1.2247   -0.7071
Optimization   terminated:   first-order   optimality   is   less
than options.TolFun.
sol=      1.2247   -0.7071
```

Example 7.5: Determine the minimum point in the vicinity of point x_0=2 and the local minimum of the function:

$$f(x) = x - \sin(x \cdot \pi) - 3, \quad x \in \boldsymbol{R}.$$

Solution: Following steps should be taken:

Step 1. Define the expression of the objective function in a function file (for example, `f.m`):

```
function y=f(x)
y=x-sin(pi*x)-3;
```

Step 2. The minimum is searched using one of the Matlab functions *fminunc* or *fminsearch*. For this example, the second function was used. The following Matlab instruction sequence is executed(for example, script file):

```
[xmin,fmin,ct_termination,details]=fminsearch('f',2)
disp(details.message)
```

Step 3. Following the execution of the sequence in step 2 the following is obtained:

```
 xmin =  2.3969
 fmin = -1.5511
 ct_termination=    1
 details =
  iterations: 13
  funcCount: 26
  algorithm: 'Nelder-Mead simplex direct search'
 message: [1x196 char]
 Optimization terminated:
 the current x satisfies the termination criteria using
 OPTIONS.TolX of 1.000000e-004 and F(X) satisfies the convergence
 criteria using OPTIONS.TolFun of 1.000000e-004
```

The minimum point is 2.3969, the minimum of the function being-1.5511. Thesearch for the minimum was completed successfully. There were 13 iterations performed, the function was used 26 times andthe simplex direct search method was used in Nelder-Mead variant. The precision is 10^{-4} for both the minimum point and the minimum (function value at the minimum point).

Example 7.6: Having the function: $f: \boldsymbol{R} \to \boldsymbol{R}$, $f(x)=\left(\frac{x}{4}\right)^2-\sin(x)-0.5$.

Determine:

i) a minimum local point in the interval (-8,10), and the corresponding minimum of the function;

ii) all the extreme local points in the interval (-8,10), as well as the values of the function in these points, while specifying the extreme type for each point.

Solution: i) Step 1. The expression of the objective function is defined in a function file (for example, `fct.m`):

```
function y=fct(x)
y= (x/4).^2-sin(x)-0.5;
```

Step 2. Since the minimum point search is done within an interval, the Matlab function *fminbnd* will be used. The following Matlab instruction sequence is executed (for example, script file):

```
[xmin,fmin,ct_termination,details]=fminbnd('fct',-8,10)
disp(details.message)
```

Step 3. Following the execution of the sequence in step 2 the following is obtained:

```
 xmin =  1.3955
 fmin = -1.3630
 ct_termination =1
```

```
 details =
    iterations: 9
 funcCount: 10
 algorithm: 'golden section search, parabolic interpolation'
 message: [1x112 char]
Optimization terminated:
the current x satisfies the termination criteria using OPTIONS.TolX
of 1.000000e-004
```

The minimum local point obtained is 1.3955, the minimum of the function being -1.3630. The search for the minimum was completed successfully (the ending constant being 1). There were 9 iterations performed, and the function was used 10 times. The golden section search and parabolic interpolation methods were used. The minimum point was determined with a precision of 10^{-4}.

ii) Step 1 is the same as the first step in i).

Step 2. To find out how many local extreme points are in the interval (-8,10), the objective function is graphically represented over this domain. Based on the graph that is obtained, for each extreme point a vicinity interval is chosen,in which there is no other extreme point. Finally, the Matlab function *fminbnd* is used for each extreme point, the search for each point being done in the vicinity interval where it is located.

The following Matlab instruction sequence is executed (for example, script file):

```
clear, clc
% graph
x=-8:0.1:10; plot(x,fct(x))

% function -f for determining the local maximums
g=@(x) -fct(x);

% determining the extreme points and the local minimums and maximums
[xmin1,fmin1]=fminbnd('fct',-6,-3);
fprintf('minimum point: %g, minimum: %g\n',xmin1,fmin1)
[xmax1,fmax1]=fminbnd(g,-3,-1);
fprintf('maximum point: %g, maximum: %g\n',xmax1,-fmax1)
[xmin2,fmin2]=fminbnd('fct',0,3);
fprintf('minimum point: %g, minimum: %g\n',xmin2,fmin2)
[xmax2,fmax2]=fminbnd(g,4,6);
fprintf('maximum point: %g, maximum: %g\n',xmax2,-fmax2)
[xmin3,fmin3]=fminbnd('fct',6,8);
fprintf('minimum point: %g, minimum: %g\n',xmin3,fmin3)
```

Step 3. Following the execution of the sequence in step 2. the 5 extreme points are obtained (the graph of the function is shown in the figure 7.2):

```
 minimum point: -4.16483, minimum: -0.269685
 maximum point: -1.79742, maximum: 0.67635
 minimum point: 1.39547, minimum: -1.36296
 maximum point: 5.46431, maximum: 2.09655
 minimum point: 6.83067, minimum: 1.89559
```

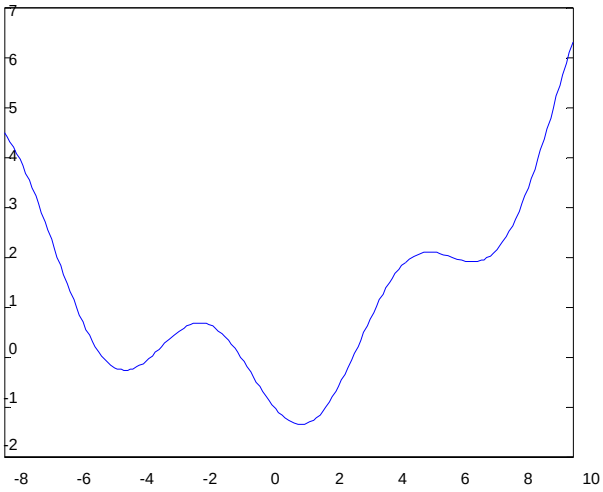

Fig.7.2. *The objective function graph for example 7.6.*

Example 7.7: Using the Matlab function based on the Quasi-Newton methodssolve the following optimization problem without restrictions:

$$(\hat{x},\hat{y},\hat{z}) = \arg \min_{(x,y,z)} f(x,y,z), \qquad f(x,y,z) = \frac{xyz}{(x-y)^2+(y-z)^2+1}, \ (x,y,z) \in \boldsymbol{R^3}$$

starting from the approximation (1,0,-1).

Solution: Step 1. The expression of the objective function is defined in a function file (for example, `fct2.m`):

```
function y=fct2(v)
y=v(1)*v(2)*v(3)/((v(1)-v(2))^2+(v(2)-v(3))^2+1);
```

Step 2. The minimum is searched with the Matlab function*fminunc*. The following Matlab instruction sequence is executed (for example, script file):

```
clear, clc
v0=[1;0;-1];
options=optimset('LargeScale','off');
[vmin,fmin,exitflag,output]=fminunc('fct2',v0,options);
fprintf('Minimum point [%g %g %g]\n',vmin)
fprintf('Minimum: %g\n',fmin)
fprintf('Optimization method: %s\n',output.algorithm)
```

Step 3. Following the execution of the sequence in step 2. the following is obtained:

```
Optimization terminated: relative infinity-norm of gradient
less than options.TolFun.
Minimum point [4.53616e+015 4.56755e+015 -2.25904e+015]
Minimum: -1.00434e+015
Optimization method: medium-scale: Quasi-Newton line search
```

Example 7.8: Solvethe following optimization problem without restrictions using the Matlab function based on the simplex search method:

$$(\hat{x},\hat{y}) = \arg \min_{(x,y)} f(x,y), \qquad f(x,y) = x \cdot e^{-x^2-y^2}, \ (x,y) \in \boldsymbol{R^2}$$

starting from the approximation (0,0).

Solution: Step 1. The expression of the objective function is defined in a function file (for example, `fvect.m`):

```
function y=fvect(v)
y=v(1)*exp(-v(1)^2-v(2)^2);
```

Step 2. The minimum is searched with the Matlab function *fminsearch*. The following Matlab instruction sequence is executed (for example, script file):

```
clear, clc
[vmin,fmin,exitflag,output]=fminsearch('fvect',[0 0]);
fprintf('Minimum point [%g %f]\n',vmin)
fprintf('Minimum: %g\n',fmin)
fprintf('Optimization method: %s\n',output.algorithm)
```

Step 3. Following the execution of the sequence in step 2. the following is obtained:

```
Minimum point [-0.707127 0.000042]
Minimum: -0.428882
Optimization method: Nelder-Mead simplex direct search
```

7.3. Problems to solve

P7.1. Determine a solution of the system of nonlinear equations:

$$\begin{cases} \sin(x+y) - 1.1 \cdot x = 0.2 \\ 1.1 \cdot x^2 + 2 \cdot y^2 = 1 \end{cases}$$

starting from the initial approximation x_0=1, y_0=1.

P7.2. Determine numerically all the solutions of the system of nonlinear equations:

$$\begin{cases} x^2 + y^2 = 4 \\ x - y = 1 \end{cases}$$

P7.3. Solve symbolically the system of nonlinear equations:

$$\begin{cases} m + p - 2 \cdot n = 0 \\ n \cdot q - p^2 = 0 \\ m + q - 37 = 0 \\ n + p - 36 = 0 \end{cases}$$

P7.4. Solve the underdetermined system:

$$\begin{cases} x^2 + y \cdot a - z = 0 \\ \dfrac{x}{z} = a \end{cases}$$

with the unknowns x, y, z.

P7.5. Determine all the numbers a,b,c which simultaneously fulfill the following conditions:

a) a, b, c are in geometric progression;

b) a, b+4, c are in arithmetic progression;

c) a, b+4, c+32 are in geometric progression.

P7.6. Determine the minimum point located in the vicinity of the point *x0*=0.5 and the local minimum of the function:

$$f(x) = \ln\left(1 - x + \frac{x^3}{3}\right),\ x \in \boldsymbol{R}.$$

P7.7. Having the function $f: \boldsymbol{R} \to \boldsymbol{R}$, $f(x) = \sin(x) + \sqrt{|x|}$ .Determine all the local extreme points in the interval (-6,6), as well as the values of the function in these points, while specifying the extreme type for each point.

P7.8. Using the Matlab function based on the Quasi-Newton methods solve thefollowing optimization problem without restrictions:

$$(\hat{x}, \hat{y}) = \arg \min_{(x,y)} f(x,y), \qquad f(x,y) = x^2 - y^2,\ (x,y) \in \boldsymbol{R^2}$$

starting from the approximation (1,0).

P7.9. Solvethe following optimization problem without restrictions using the Matlab function based on the simplex search method:

$$(\hat{x}, \hat{y}, \hat{z}) = \arg \min_{(x,y,z)} f(x,y,z), \qquad f(x,y,z) = \frac{x}{x^2 + y^2 + z^2 + 1},\ (x,y,z) \in \boldsymbol{R^3}$$

starting from the approximation (0.6,-0.2,-0.1).

ANNEX M7. ELEMENTS REGARDING THE SOLVING OF SYSTEMS OF NONLINEAR EQUATIONS. ELEMENTS REGARDING THE SOLVING OF OPTIMIZATION PROBLEMS

M7.1. Systems of nonlinear equations

It is called ***a system of n nonlinear equations with n unknowns*** any system that can be brought to the form:

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = c_1 \\ f_2(x_1, x_2, \cdots, x_n) = c_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_n(x_1, x_2, \cdots, x_n) = c_n \end{cases}$$

where:

- $c_1, c_2, \ldots, c_n$, are real constants;
- $f_1, f_2, \ldots, f_n$, are on the domain of interest continuous functions of the real variables $x_1, x_2, \ldots, x_n$, and do not contain constant terms;
- at least one of the functions, f_i, is nonlinear with respect to at least one unknown.

Comment: The function f_i is linear if $\forall x, y$ n-dimensional vectors in the domain of interest and $\forall a,b \in \boldsymbol{R}$, the following relationship is fulfilled:

$f_i(a \cdot x + b \cdot y) = a \cdot f_i(x) + b \cdot f_i(y)$.

A system of nonlinear equations can have a finite number of solutions (compatible anddetermined), an infinity of solutions (compatible and undetermined) or no solution (incompatible).

The solving of a system of nonlinear equations using numerical methods implies - as well as the solving transcendent equations - the following two stages:

I. ***separating (localizing)*** the solutions, that is, decomposing the domain of interest into a partition of subdomains, so that each subdomain contains at most a solution;

II. ***calculating the solutions with a predetermined precision***, usually starting from sets of approximate values of them.

Arithmetic Progressions. Geometric Progressions

Arithmetic Progressions

It is called ***an arithmetic progression*** a series of numbers in which, each term, starting with the second term, is obtained from the previous one by adding to it a constant real number, called ***the common difference of the arithmetic progression***.

The sum of the first n terms of an arithmetic progression $a_1, a_2, \ldots, a_k, \ldots$ is

$$S_n = \frac{(a_1 + a_n)n}{2}$$

The necessary and sufficient condition for three real numbers a, b and c to form an arithmetic progression (with the middle term b) is that they satisfy the relation $2 \cdot b = a + c$ (i.e., the middle term is the arithmetic mean value of the other two terms).

Geometric Progressions

It is called ***a geometric progression*** a series of numbers in which, each term, starting with the second term, is obtained from the previous one bymultiplying it with a constant real nonzero number, called ***the common ratioof the geometric progression***.

The sum of the first n terms of an geometric progression $b_1, b_2, \ldots, b_k, \ldots$ with the ratio different than 1 is $S_n = b_1 \frac{1-q^n}{1-q}$.

The necessary and sufficient condition for three real numbers a, b and c to form an geometric progression (with the middle term b) is that they satisfy the relation $b^2 = a \cdot c$ (i.e., the middle term in absolute value, be the geometric mean of the other two terms).

M7.2. Optimization problems

a. Optimization problems

Given a function f: $D \rightarrow \boldsymbol{R}$, where D is a set, usually, $D \subseteq \boldsymbol{R^n}$, $\boldsymbol{R^n}$ being Euclidean space, the determination of an element is required $x_0 \in D$, such that $f(x_0) \leq f(x)$, $\forall x \in D$, in the case of the minimization criterion, respectively, $f(x_0) \geq f(x)$, $\forall x \in D$, for the maximization criterion.

An element x of D is called ***an admissible solution***. f iscalled ***objective function***. An admissible solution for which the optimum of the objective function is obtained (minimumor maximum, depending on the desired criterion) is called an ***optimal solution*** (also called ***the global minimum point***, in the case of a minimization problem, respectively, ***the global maximum point***, in the case of a maximization problem). D is called ***the space of admissible solutions***.

If $D = \boldsymbol{R^n}$, the optimization problem is ***a problem without restrictions***.

An element $x_l \in D$, with the property that there is a vicinity $V \subset D$ of it such that $f(x_l) \leq f(x)$, $\forall x \in V$, in the case of a minimization criterion, respectively, $f(x_l) \geq f(x)$, $\forall x \in V$, in the case of a maximization criterion, is called ***the local optimal point*** (***local minimum point***, respectively ***local maximum point***), and the value of the function f in x_l is called ***the local optimum***.

Solving an optimization problem involves using ***an optimization method*** to determine the optimal solution.It can be observed that a maximization problem can be transformed into a minimization problem, and vice versa, by replacing the objective function f with the opposite, $-f$.Hereinafter, only minimization problems without restrictions are considered.

b. Optimization methods for optimization problems without restrictions

The classical optimization methods in the case of optimization problems without restrictionscan be grouped into the following categories:

- ***non-iterative methods*** (in one step), which is based on certain properties of the objective function, such as derivability, convexity, etc.;
- ***iterative methods***(in several steps), starting with an initial solution and determining at each step a new solution; the most popular methods in this category are methods of decreasing(of relaxation, of descent), characterized in that the value of the objective function decreasesat each step, which can be classified in the following subcategories:
    - zero order methods - require only calculating the values of the objective function(for example, genetic algorithms, the golden section method);
    - first order methods – requires both the calculation of the values of the objective function and its gradient (for example, the gradient method, conjugate gradient methods);
    - second order methods – requires the calculation of the values of the objective function as well as its gradient and hessian (for example, Newton's classic method).