APPLICATIONS
THE MODEL SOLVATION OF COMPLETE SYSTEM

In the next part we expand the numerical method from the chapter [2,2] of rezolving a nonlinear system of equations:

We can consider a nonlinear system:

$$F_j(x_1,\ldots,x_i,\ldots,x_n)=0 \qquad (j=1,\ldots,m) \qquad \mathbf{1}$$

We costruct the equivalent problem for the calculus of the values $x_1,\ldots,x_n$, which provide the functional minimization:

$$\min\left(\Phi=\sum_{j=1}^{m}F_j^2(x_1,\ldots,x_i,\ldots,x_n)\right) \qquad \mathbf{2.}$$

Where the functions are given under the form (1). We have to observe that the solution $x_1,\ldots,x_n$, of the system (1) provide in the same time the functional minimum (2), is true the reciprocal too. In these conditions the solvation method can be replaced with the problem of minimization of a functional without supplementary restrictions. The last problem is more general and easier, using one from the known methods of improvement without restrictions, for example:

-the method of unidimensional variation
-the Box method
-the Hooke-Jeeves method (pattern search)
-the Rosenbrock method
-methods of gradient type
-the method of type Newton-Rapson
-methods of contraction, etc.

The Box method (in literature known as different denominations: the Spandley method or Hex method, or the *n*-dimensional simplex), does not presume the continuity, the derivation of the functions. In the *n*-dimensional space we construct a regular figure with *n*+1 vertexs with equal distances between vertexs named simplex (for example in the three-dimensional a tetrahedron, etc.). We calculate the Φ in the simplex vertexs. The afferent vertex to the highest value, repetitive we replace with a vertex of opposite sense, symmetrical, obtaining a new simplex. Repeating the method we skip the anterior selected vertexes, in case of need for precision we reduce the dimension of the simplex side, obtaining the vertex of expected value. In literature are indicated the expressions of obtaining the simplex vertexes, of choosing the new vertex.

The model solving steps are:

- We read the number of variables (n), the initial size of the n-dimensional (a) simplex side, the error threshold (ε) for the Φ calculus, the maximum number of iterations (t)
- We construct the vertexes V_1, ..., V_i, ..., V_{n+1} of the simplex, in this way:

$$V_i = V_i\left(x_i^1, \ldots, x_i^j, \ldots, x_i^n\right)$$

3. where $x_i^j = \begin{cases} 0 & \text{dacă } i = 1 \\ q & \text{dacă } j+1 \neq i,\ i > 1 \\ p & \text{dacă } j+1 = i,\ i > 1 \end{cases}$

$$p = \frac{a}{n\sqrt{2}}\left(n - 1 + \sqrt{n+1}\right)$$

$$q = \frac{a}{n\sqrt{2}}\left(-1 + \sqrt{n+1}\right)$$

- We calculate the Φ value in every vertex of the simplex
- We choose the highest value (the worst possibility) of Φ in the simplex vertexes, vertex that was not selected before and we keep its value (x_i^R)
- We calculate the new symmetrical vertex in the place of the selected one for elimination (this will be the image of the vertex that have to be eliminated in the mirror of the other vertexes) using the relation (3,4)
- $$x_i^N = \left[\frac{2}{n}\left(\sum_{j=1}^{n+1} x_i^j - x_i^R\right)\right] - x_i^R \qquad (i=1, .\ n) \qquad 4.$$

- We repete the algorithm from the point 3 until the way out condition in one of the vertexes is satisfied, $\Phi \leq \varepsilon$ or the iterations number has reached the t value.
- We post the point coordinates, the Φ value, the iterations number, ε, t, a values

If the approximation is not acceptable, we reduce the size of the simplex side, and we continue from point 3 omitting the selected vertexes choosing a less bad vertex.

There are not restrictions between the number of relations and the number of variables. If the problem has more solutions, in literature are often introduced supplimentary aim functions, obtaining optimum solutions.

NUMERICAL SIMULATION

In the presentation we use the research results realised with the support of Cluj Sapientia University in 2004. We are foreshadowing the solvation of a nonlinear electric network open with the relations system:

$$\begin{cases} 4x^3 - y^3 - 3x^2y + 3xy^2 + y - 5 = 0 \\ 3x^3 + y^2 + x - 9 = 0 \end{cases} \qquad \textbf{2.1.}$$

We construct the functional .2):

$$\begin{aligned} \Phi &= F_1^2(x,y) + F_2^2(x,y) = \\ &= \left[4x^3 - y^3 - 3x^2y + 3xy^2 + y - 5\right]^2 + \left[3x^3 + y^2 + x - 9\right]^2 \end{aligned} \qquad \textbf{2.2.}$$

Procedure MATLAB :

```
a = 0.0001;
x = zeros(3,1);
y = zeros(3,1);
F = zeros(3,1);
maxx = 0;
maxy = 0;
imax = 0;
tx = 0;
ty = 0;
nx = 0;
ny = 0;
x(2) = 0.0001;
y(3) = 0.0001;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j = 1:100000
F(1) = (4*x(1)^3-y(1)^3-3*x(1)^2*y(1)+3*x(1)*y(1)^2+y(1)-
5)^2+(3*x(1)^3+y(1)^2+x(1)-9)^2;
F(2) = (4*x(2)^3-y(2)^3-3*x(2)^2*y(2)+3*x(2)*y(2)^2+y(2)-
5)^2+(3*x(2)^3+y(2)^2+x(2)-9)^2;
F(3) = (4*x(3)^3-y(3)^3-3*x(3)^2*y(3)+3*x(3)*y(3)^2+y(3)-
5)^2+(3*x(3)^3+y(3)^2+x(3)-9)^2;
maxF = max(F);
imax = 0;
for i = 1:3
if maxF == F(i)
imax = i;
end
end
tx = sum(x);
ty = sum(y);
nx = tx - 2*x(imax);

ny = ty - 2*y(imax);
```

```
x(imax) = nx;
y(imax) = ny;
F(imax) = 0;
end
F(imax)=(4*x(imax)^3-y(imax)^3-3*x(imax)^2*y(imax)+3*x(imax)*y(imax)^2+y(imax)-
5)^2+(3*x(imax)^3+y(imax)^2+x(imax)-9)^2;
maxF = min(F);
for i = 1:3
if maxF == F(i)
imax = i;
end
end
display(F(imax))
display(x(imax))
display(y(imax))
```

BIBLIOGRAFY

SZÉL A. and GACSADI A. and KAKAS S., *Általános rendszermodell fejlesztése az inercialitás függvényében és alkalmazásai Kolozsvár Sapientia Alapitvány, (K/449-2003. contract)* 2004

2. SZÉL A. VOL THE THEORY OF VARIATION IN STUDY SYSTEMS EMT Cluj Napoca 2010 ISBN 978-973-7840-18-9