USING INFORMATICS PROGRAMMES TO DESIGN DRAINAGE ARRANGEMENTS FOR EXCESSIVELY HUMID SOILS

FOLOSIREA PROGRAMELOR INFORMATICE PENTRU PROIECTAREA AMENAJĂRILOR DE DRENAJ PE TERENURI CU EXCES DE UMIDITATE

T. E. MAN, Claudia BURAN

Politehnica University of Timisoara
Corresponding author: Teodor Eugen MAN, e-mail:eugen@zavoi.ro

Abstract: *The paper presents a methodology to set up a software product used to design drainage arrangements for soil with excessive humidity. Due to the fact that the design of a complex software programme has to be carried out as a team, we have to strictly follow certain steps: problem analysis, application design, sub-problem analysis, programming language transposition, programme testing, drawing up the documentation and checking the final product. The paper presents a case study to achieve a software application for drainage design.*

Rezumat: *Lucrarea de faţă prezintă metodologia de elaborare a unui produs informatic pentru proiectarea amenajărilor de drenaj pe terenuri cu exces de umiditate. Datorită faptului că un program informatic complex se realizează în echipă, trebuie respectate cu stricteţe câteva etape: analiza problemei, modelarea aplicaţiei, analiza fiecărei subprobleme, transpunerea într-un limbaj de programare, testarea programului, elaborarea documentaţiei şi verificarea produsului final. Lucrarea prezintă un studiu de caz pentru realizarea unui soft folosit in calculul de proiectare al drenajului.*

Key words**:** *drainage, informatics programs, design, humidity excess soils.*
Cuvinte cheie**:** *drenaj, programe informatice, proiectare, sol cu exces de umiditate.*

INTRODUCTION AND GENERAL ISSUES

Considering the great importance of soils with excessive humidity in agriculture, as well as the great number of socio-economic objectives lying in such an area, draining-sewing works are a core issue.

Draining-sewing systems, as designed according to a specific drainage flow calculated on the level of the years 1960-1970, can no longer cope with the evacuation of excessive water, considering the high level of rainfalls leading to floods. Most of the draining-pipes are clogged up and are influenced by vegetation, while the thirty-year old pumps are not efficient anymore. All these elements harm the exploitation of the systems.

Currently, the requirements and demands on the level of the European Union impose a new approach of the global environmental issues from the point of view of the impact and tension on the environment and of all the consequences of the socio-economic development. As far as land improvement is concerned, we need to modernise the irrigation and draining-sewing systems existing on a national level, so that they may become reliable and efficient. This may be done by providing modern equipment with the adequate calculation technology and the appropriate software that facilitate optimal system design and exploitation.

MATERIALS AND METHODS

Generally, any problem that has to be solved with the assistance of the computer involves certain stages to reach the solution. We will further present the steps to set up an informatics product to calculate the distance between the drains.

RESULTS AND DISSCUTIONS

Problem analysis

At this stage, we study the problem and establish the entry data, the output data and we formulate the detailed requirements of the user (beneficiary). The final result should be the thorough understanding of the requirements – the problem specifications or field, and to eliminate any ambiguity of the general formulation.

Problem: Calculating the investment needed to plan the draining works on a hectare of soil.

The calculation formula for the problem is as follows:

$$Is = cost \times 10 : L \quad (1)$$

where Is – specific investment;
Cost – lei/km;
L – drain length for one hectare.

When analysing the problem, the problem will be analysed according to a top-down pattern, that is, the problem will be first analysed in general, then decomposed in sub-problems.

Moulding the application – it is based on a functional decomposition into modules.

A system is divided into subsystems providing one, or more, functions, called services. In a structured approach, each subsystem contains a series of resources making up a functional unit (the subsystem in itself) that actually consists of a series of procedures and functions. The application mould is reduced to a top-down type approach.

The results of the moulding procedure itself consist in fact of three models:

- Static model – suggests the constructive units of the application (modules) and establishes their levelled hierarchy relation;
- Dynamic model – suggests the succession in time of the events and the stages through which the application passes as their direct consequence;
- Functional model – describes the data flow between the application components, each processing being moulded as an independent process.

For this problem we will choose the functional model. The following information results from the beneficiary:

- After studies and lab measurements, we determine the hydraulic and geometric features of the draining tubes of the filtering materials;
- The distance between the drains is calculated for a soil profile (Figure 1) by using Ernst's formula:

$$h = \frac{q \cdot D_v}{K_1} + \frac{q \cdot L^2}{8 \cdot K_1 \cdot T_e} + \frac{q \cdot L}{K_1} \cdot \ln \frac{\alpha \cdot D_0}{U} + \frac{q \cdot L}{K_1} \cdot \zeta_{if} \quad (2)$$

where: $T_e = D_1 + D_2 \cdot \frac{K_2}{K_1}$ (3)

- We notice that for the calculation of the distance between the drains we need the hydraulic resistance quotient at entrance to the draining tube (ζ_{if}) that we calculate with I. David's relation;
- L is determined by solving the second degree equation in Ernst's formula;
- The number of drain lines is determined on a 100-meter width, as well as the needed drain length and the specific investment.

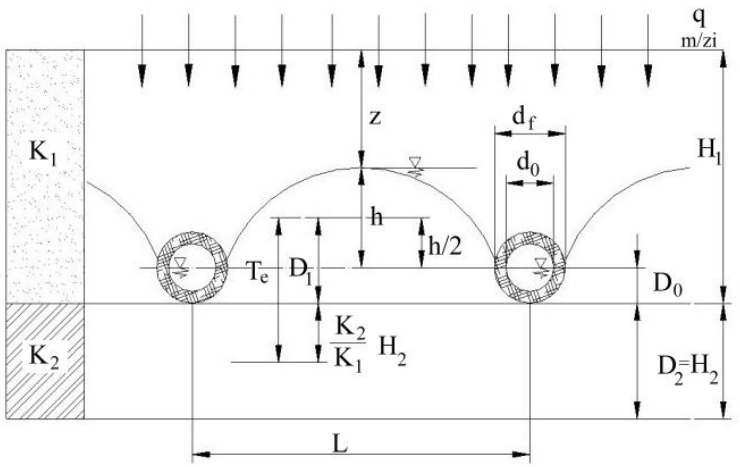
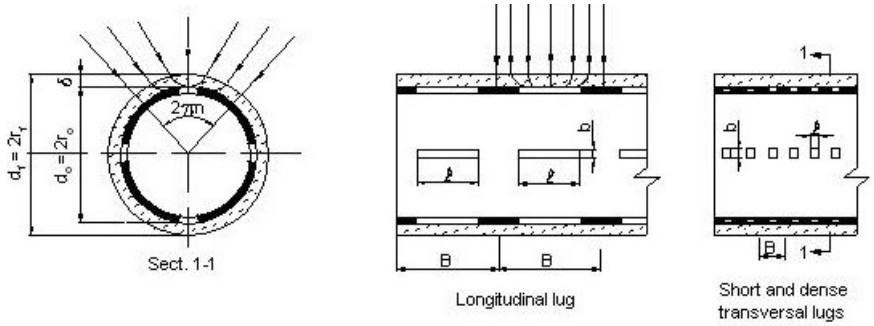


Figure 1 – Schema to calculate draining in the case of a layered soil, when using a filtering material

Analysis of each sub-problem

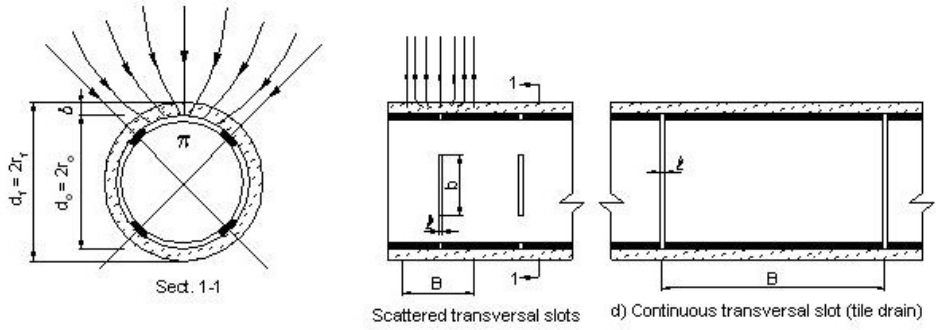
At this stage, we set up a method to solve each sub-problem. We then set up an algorithm on the basis of each and every method. At this stage, it is better to check each algorithm after conceiving it.

The following is an example for the calculation of the hydraulic resistance quotient at water entrance to the draining tube (ς_{if}), with I. David's relation:



a) $\ell << b$ b) $\ell < b$; $\ell > b$

$$B < \frac{\pi \cdot d_o}{n}$$



c) $b << \ell$

$B << \ell$

Figure 2 – Typical scheme of slots and lugs positioning on the draining tube

The calculation of (ς_{if}) according to I. David for the case of the four geometries of the draining tube slots in figure 2, provided the existence of a certain width around the filter exists or not, is the following:

$$\zeta_{if} = \alpha \cdot \left[\ln \frac{1}{\sin \frac{nb}{2d_0}} + \frac{1-\chi}{2\chi} \cdot \ln \left(A_1 + \sqrt{A_1^2 + 1} \right) \cdot \left(A_2 + \sqrt{A_2^2 + 1} \right) \right] +$$

$$+ \beta \cdot \left[\ln \frac{1}{\sin \frac{\ell}{2B}} + \frac{1-\chi}{2\chi} \cdot \ln \left(B_1 + \sqrt{B_1^2 + 1} \right) \cdot \left(B_2 + \sqrt{B_2^2 + 1} \right) \right] \quad (4)$$

where: - for the holes (lugs) along the generator α and β have the following expressions:

$$\alpha = \frac{2 \cdot B}{n \cdot \pi \cdot \ell}; \qquad \beta = \frac{2 \cdot B}{\pi^2 \cdot d_0} \quad (5)$$

- for holes (slots) along the circumference, we have as follows:

$$\alpha = \frac{2}{n \cdot \pi}; \qquad \beta = \frac{2 \cdot B}{\pi \cdot n \cdot b} \quad (6)$$

These differences originate in the acceptance of a debit concentration on the two directions depending on the overweight of lugs (slots) after the generator, circumference respectively.

In the relation (4) A_1, A_2, B_1, B_2, and (χ) have the following expressions:

$$A_1 = \frac{\left(\frac{d_f}{d_0}\right)^n - 1}{2\left(\frac{d_f}{d_0}\right)^{\frac{n}{2}} \sin\frac{nb}{2d_0}}; A_2 = \frac{\sqrt{\left(\frac{d_f}{d_0}\right)^{2n} - 1}}{2\left(\frac{d_f}{d_0}\right)^{\frac{n}{2}} \sin\frac{nb}{2d_0}} \quad (7)$$

$$B_1 = \frac{\text{sh}\frac{\pi(d_f - d_0)}{B}}{\sin\frac{\pi \cdot \ell}{2B}}; \quad B2 = \sqrt{\frac{1}{2}\left[\sqrt{1 + 4\left(\frac{\text{sh}\frac{\pi(d_f - d_0)}{2B}}{\sin\frac{\pi \cdot \ell}{2B}}\right)^2\left(\frac{\text{ch} \cdot \frac{\pi(d_f - d_0)}{2B}}{\sin\frac{\pi \cdot \ell}{2B}}\right)^2} - 1\right]} \quad (8)$$

$$\chi = \frac{K_{fc}}{K_{sol}} \quad (9)$$

Where:

ℓ - lugs length along the generator (width of the slots on circumference);

b – lugs width along the generator (respectively slots length on circumference);

B – distance between the lugs (slots along the generator);

n – number of lugs (slots) on circumference;

d_o – external diameter of the draining tube;

d_f – external diameter of the filter;

k_{fc} – permeability quotient of the filtering material;

k_{sol} – permeability quotient of the surveyed soil.

For particular cases, in the classical calculation, the formulae are simplified, but in the case of the programme, it is enough to analyse the general case, as the others are obtained for specific data samples.

Transposing the algorithm in a programming language

At this stage, special attention will be paid to variable declaration. It is better to avoid using global variables, except for entry-output elements that will clearly be used by most sub-programmes.

Each module will be implemented through a separate sub-programme (function or procedure). Thus, to calculate the hydraulic resistance quotient Zif, we will use a function (implemented under C++) that will return a corresponding real value with the following prototype:

float coef_rez_hidr()

To determine L, a function carrying the parameter transmitted through the hydraulic resistance quotient value will be used; it will have the following prototype:

float calcullung (float *zif);

The implementation of the function calculating the hydraulic resistance quotient is the following:

```
float coef_rez_hidr()
{float pi=3.14159;
int opt, n;
float lambda, b, B, d0, df, kfc, ksol, alfa, beta, hi;
float  A1, A2, B1, B2, zita1, zita2; zita;
clrscr();
cout<<"\n Calculul coeficientului de rezistenta hidraulica";
cout<<"\n Date de intrare";
cout<<"\n Lungimea sliturilor in lungul generatoarei"; cin>>lambda;
cout<<"\n Latimea sliturilor in lungul generatoarei"; cin>>b;
cout<<"\n Distanta intre sliturI"; cin>> B;
cout<<"\n Numarul sliturilor (fantelor) pe circumferinta;  cin>n;
cout<<"\n Diametrul exterior al tubului de dren " cin>>d0;
cout<<"\n Diametrul exterior al filtrului "; cin>>df;
cout<<"\n Coeficientul de permeabilitate a materialului filtrant"; cin>>kfc;
cout<<"\n Coeficientul de permeabilitate a solului studiat"; cin>>ksol;
clrscr();
cout<<"\n Relatia de calcul dupa David \n";
cout<<"Alegeti varianta \n";
cout<<"\n  1. pentru orificiile practicate in lungul generatoarei";
cout<<"\n  2. pentru orificiile practicate in lungul circumferintei";
cin>>opt;
switch(opt)
{case 1: alfa= 2*B/(n*pi*lambda); beta=2*B/(pi*pi*d0); break;
  case 2: alfa=2/(n*pi); beta=2*B/(pi*n*b); break;
default: cout<<"\n Varianta incorecta ";
}
if ((opt==1)|| (opt==2))
{ hi=kfc/ksol;
A1=(pow(df/d0,n)-1)/(2*pow((df/d0), n/2)*sin((n*b)/(2*d0)));
A2=sqrt((pow(df/d0,2*n)-1))/(2*pow((df/d0),n/2)*sin((n*b)/(2*d0)));
B1=(sinh((df-d0))/B)/sin(pi*lambda/(2*B));
B2=sqrt(sqrt(1+4*pow((sinh(pi*(df-
d0)/(2*B))/sin(pi*lambda/(2*B))),2)*pow((cosh(pi*(df-d0)/(2*B))
     /sin(pi*lambda/(2*B))),2)-1)/2;
zita1 = alfa*(log(1/sin((n*b)/(2*d0)))+(1-
hi)/(2*hi)*log((A1+sqrt(A1*A1+1))*(A2+sqrt(A2*A2+1))));
zita2 = beta*(log(1/sin(lambda/(2*B)))+(1-
hi)/(2*hi)*log((B1+sqrt(B1*B1+1))*(B2+sqrt(B2*B2+1))));
zita=zita1+zita2;
return zita;}
```

The function should be checked right after implementation by using at least 10 sets of data experimentally obtained as entries and data provided by calculation as outputs. If the data have not close values, the algorithm should be checked step-by-step to eliminate possible errors (of conception or elaboration). A correct algorithm will provide more precise data than the classical method of calculation.

Testing the programme

The action of testing a programme is different from the other stages due to its rather destructive character, as the aim is to set forth the malfunction of the output. From a psychological point of view, the programmer has to adopt a "hostile" attitude towards the programme and determine as many errors as possible.

The data experimentally obtained will be introduced and the results obtained will be checked, in order to see if they fit the error limit. This should be done in two ways:

- functional testing (black box method) – knowing the functions the programme has to fulfil, the samples will be conceived in such a manner to confirm that each function is fully completed;

- structural testing (transparent box method) – knowing the structure (instructions) of the programme, the samples are conceived in such a manner to provide a convincing testing of all parts of the programme.

The testing should be witnessed by both the programmer and the specialist in the field of draining.

Elaborating the user's guide

At this stage, a user's guide, or a tutorial CD, should be drawn up for the programme. This should be complete and easy to understand by an ordinary user.

Checking the final product is the last step in making an informatics product and aims at determining and eliminating inconveniences noticed by the beneficiary after a certain period of using the informatics product, called trial period.

The possible alterations should be written down in the user's guide as "further notes", or "version notes".

After this step, the product is delivered together with the necessary documents and eventually, upon request from the beneficiary, training of the staff to be in charge with it.

CONCLUSIONS

The introduction of the computer in the design activity has increased its efficiency. In Romania, after 1990, several companies specialised in soft in different fields have been created. Programming is a team work and clearly respects the steps as mentioned in this paper.

One of the greatest problems that may come up in conceiving a complex informatics product is communication with the beneficiary. Should a small detail be misunderstood by the programmer, the product is no longer reliable.

For these reasons, it is better that the informatics products be made by a team; at least one person in the team should have double specialisation (both programming and land management).

This soft facilitates a quick design calculation of distance between drains by calculating the value of the hydraulic resistance quotient at water entrance to the drain, or in the drain complex + filter (ς_{if}) by using the results of the lab draining surveys.

REFERENCES

1. Blidaru V., Wehry A., Pricop Gh. - *Amenajări de irigaţii şi drenaje*, Interprint Press, Bucharest, 1997
2. Frenţiu, M., Pârv, B. - *Elaborarea programelor. Metode şi tehnici moderne,* ProMedia Press, Cluj-Napoca, 1994
3. Man T.E., Wehry A., David I., Popescu F. – *Drainage Studies for Ground Arrangement Solutions of Soils with Humidity Excess from the Western Part of Romania (Timiş, Arad, Bihor,*

Maramureş and Satu-Mare Counties), International Drainage Symposium of ASAE Sheraton Grand Hotel & Sacramento Convention Center Sacramento USA, 21-24.03.2004, pp. 272 – 280.
4. Man T.E. : *Studiul rezistentelor hidraulice la intrarea apei in tubul de dren* – Doctoral Thesis, I.P.T.V.T. 1983
5. Man E.T., Halbac – Cotoara R.*: Metode clasice si moderne de proiectare a amenajarilor de drenaje folosite in tara noastra sip e plan mondial*, pag. 146 – 154, Scientific Newsletter of the Polytechnic University of Timisoara, Hidrotehnica Series, Tom. 49 (63), Fabc. 1/2005, ISSN 1224 – 6042, Politehnica Press
6. Wehry A, David I, Man T.E., *Probleme actuale in tehnica drenajului*, Facla Press, Timisoara, 1982