Program de calcul automat al regimurilor optimale de prelucrare la debitarea prin eroziune electrică complexă

Lect.drd.ing. Tiberiu Marius Karnyanszky
Universitatea „Tibiscus" Timişoara

ABSTRACT. The complex electric erosion is a member of the very large category of unconventional processing technology. The determination of the mathematical pattern of the dependence of the output values through the input values is still partial unknown. That's the reason this paper try to introduce two methods to optimize the functions which describes the connection among the productivity of the processing or the surface quality (as outputs) and the electrical, mechanical, technological parameters (as inputs).

1 Aplicarea simulării la procesele tehnologice

Operația de simulare aplicată asupra proceslor tehnologice, în general, şi asupra prelucrării prin eroziune electrică complexă (EEC) în particular, presupune mai întâi elaborarea unui model matematic de forma unor ecuații, care să descrie relațiile dintre factorii de intrare şi cei de ieşire din sistem (proces tehnologic).

Este însă la fel de adevărat că, dacă modelul matematic nu poate fi exprimat concret (matematic), simularea poate avea loc deoarece ea poate să facă ajustarea parametrilor simulării până la apropierea de criteriile funcționale dorite, în aşa fel încât să permită elaborarea modelului matematic.

Simularea presupune parcurgerea următoarelor etape ([Poc98, SL00, Vad77]):

- **dezvoltarea modelului**, adică elaborarea dependențelor matematice care descriu funcționarea sistemului şi implementarea lor într-un mod care să permită utilizarea (prelucrarea) folosind calculatorul;
- **testarea modelului**, adică verificarea comportamentului în condiții specificate pentru a verifica modul de adecvare a modelului la realitate;
- **exploatarea modelului**, adică utilizarea sa pentru a descrie modul în care sistemul respectiv funcționează în realitate.

2 Optimizarea modelului prin simulare

Sunt două metode generale pentru căutarea optimului unei funcții utilizând simularea numerică ([VSO83]):

- metoda explorării aleatoare pure, constând în generarea unui număr de puncte uniform repartizate într-un domeniu dat şi reținerea după fiecare ciclu de simulare a valorii optime a funcției;
- metoda explorării aleatoare lente dirijate, constând în efectuarea unor calcule aritmetice şi teste comparative în cadrul unui ciclu de simulare, care devine astfel mai lung dar scade numărul total de cicluri.

Am apelat la două metode de simulare din cea de-a doua categorie, pentru funcțiile de o singură variabilă respectiv pentru cele de două variabile.

3 Metoda reducerii domeniului admisibil la funcțiile de o variabilă

Metoda se poate aplica funcțiilor continue şi unimodale ([VSO83]) ceea ce este cazul la prelucrările tehnologice prin EEC. Paşii algoritmului de simulare pentru maximizarea funcției $f:[a,b]\rightarrow R$ sunt:

Pasul 1. Se generează n numere aleatoare distincte x_i, i=1,n, uniform repartizate pe domeniul de definiție a funcției.

Pasul 2. Se ordonează crescător şirul $\{x_i\}$.

Pasul 3. Se calculează valorile $y_i=f(x_i)$, i=1,n. Dacă $y_1<y_2<y_3$ atunci calculele continuă. Calculele se opresc atunci când:

- $f(x_{i-1})<f(x_i)$ şi $f(x_{i+1})<f(x_i)$ (1)

  În această situație, se restrânge domeniul de definiție al funcției la intervalul [a', b'] unde $a'=x_{i-1}$ iar $b'=x_{i+1}$;

- $f(x_{n-1})<f(x_n)$ (2)

  caz în care se iau a'=x_n iar b'=b;

- $f(x_{i-1})=f(x_i)$ dar $f(x_i)>f(x_{i+1})$ (3)

  caz în care se iau a'=x_{i-1} iar b'=x_i.

Pasul 4. Dacă:

$$b'-a'<\varepsilon \text{ şi } |f(b')-f(a')|<\eta \quad (5)$$

unde ε şi η sunt abateri admisibile, atunci valoarea optimă este $x_{optim}=(a'+b')/2$. Dacă relația (5) nu este satisfăcută atunci se reia algoritmul de la pasul 1.

Algoritmul care implementează această problemă este:

```
program Simulare1;
uses crt;
type vectorr=array[0..10]of real;
var coef,x,y:vectorr;
    i,j,m,n:integer;
    a,b,epsilon,ita,pas,fa,fb,xoptim:real;
    gata:boolean;
function f(k:integer;coef,x:real):real;
var i:integer; g:real;
begin
  g:=coef;
  for i:=1 to k do g:=g*x;
  f:=g;
end;
begin
  clrscr;
  writeln('Introduceti datele despre functia f:');
  write('Gradul polinomului n=');
  readln(n);
  writeln('Introduceti coeficientii polinomiali:');
  for i:=n downto 0 do
    begin
    write('a[',i,']=');
    readln(coef[i]);
    end;
  writeln('Introduceti domeniul de definitie a functiei f:');
  write('a='); readln(a);
  write('b='); readln(b);
  write('Introduceti epsilon='); readln(epsilon);
  write('Introduceti ita='); readln(ita);
  gata:=false; m:=10;
```

```
  repeat
    pas:=(b-a)/m;
    for i:=0 to m do x[i]:=a+i*pas;
    for i:=0 to m do
      begin
      y[i]:=coef[0];
      for j:=1 to n do y[i]:=y[i]+f(j,coef[j],x[i]);
      end;
    for i:=0 to m do
      begin
      if (y[i-1]<y[i]) and (y[i+1]<y[i]) then
        begin
        a:=x[i-1];
        b:=x[i+1];
        end;
      if (y[m-1]<y[m]) then
        begin
        a:=x[m];
        end;
      if (y[i-1]=y[i]) and (y[i]>y[i+1]) then
        begin
        a:=x[i-1];
        b:=x[i];
        end;
      end;
    fa:=coef[0];
    for j:=1 to n do fa:=fa+f(j,coef[j],a);
    fb:=coef[0];
    for j:=1 to n do fb:=fb+f(j,coef[j],b);
    if ((b-a)<epsilon) and (abs(fb-fa)<ita) then
      begin
      xoptim:=(a+b)/2;
      writeln('Solutia optima este xoptim=',xoptim);
      gata:=true;
      readln;
      end;
  until gata;
end.
```

Programul a fost verificat experimental pentru dependența între productivitatea debitării şi puterea indusă în spațiul de lucru $Q_p=f(P)$ iar rezultatele au fost confirmate prin comparație cu cele obținute prin interpolare folosind metoda celor mai mici pătrate.

4 Metoda reducerii domeniului admisibil la funcţiile de două variabile

Pentru optimizarea funcţiei de două variabile f:[a,b]x[c,d]→R se parcurg următorii paşi:

Pasul 1. Se alege un punct (x,y) ∈ [a,b]x[c,d].

Pasul 2. Se menţine y constant şi se determină valoarea x' pentru care:

$$f(x',y) = \text{optim } f(x,y) \quad (6)$$

Pasul 3. Se menţine x' constant şi se determină y' pentru care:

$$f(x',y') = \text{optim } f(x',y) \quad (7)$$

Pasul 4. Se verifică dacă:

$$|x-x'|\leq\varepsilon_1,\ |y-y'|\leq\varepsilon_2,\ |f(x,y)-f(x',y')|\leq\eta \quad (8)$$

unde ε_1, ε_2 şi η sunt abaterile admisibile. Dacă da, atunci optimul este $x_{optim}=(x+x')/2$ şi $y_{optim}=(y+y')/2$ şi algoritmul se încheie. Dacă nu, se trece la pasul 5.

Pasul 5. Se compară x cu x' şi y cu y'. Dacă y'<y se ia d=y; dacă y'>y se ia c=y; dacă x'<x se ia b=x; dacă x'>x se ia a=x şi se revine la pasul 2.

Algoritmul care implementează această problemă este:

```
program Simulare2;
uses crt;
type matricer=array[0..10,0..10]of real;
     vectorr=array[0..10]of real;
var x,y:vectorr;
    coef,z:matricer;
    i,j,k,l,m,n:integer;
    a,b,c,d,epsilon,ita,pasx,pasy,fa,fb,fc,fd,xoptim,yoptim:real;
    gata:boolean;
function f(k,l:integer;x,y:real):real;
var i:integer; g:real;
begin
  g:=1;
  for i:=1 to k do g:=g*x;
  for i:=1 to l do g:=g*y;
  f:=g;
end;
begin
  clrscr;
  writeln('Introduceti datele despre functia f:');
```

```
write('Gradul maxim al polinomului n='); readln(n);
writeln('Introduceti coeficientii polinomiali:');
for i:=0 to n do
  for j:=0 to n do
    begin
    write('coeficientul lui x^',i,'*y^',j,'='); readln(coef[i,j]);
  end;
writeln('Introduceti domeniul de definitie a functiei f:');
write('a='); readln(a);
write('b='); readln(b);
write('c='); readln(c);
write('d='); readln(d);
write('Introduceti epsilon='); readln(epsilon);
write('Introduceti ita='); readln(ita);
gata:=false; m:=10;
repeat
  pasx:=(b-a)/m;
  pasy:=(d-c)/m;
  for i:=0 to m do x[i]:=a+i*pasx;
  for i:=0 to m do y[i]:=c+i*pasy;
  for i:=0 to m do
    for j:=0 to m do
      begin
      z[i,j]:=1;
      for k:=0 to n do
        for l:=0 to n do
          z[i,j]:=z[i,j]+coef[k,l]*f(k,l,x[i],y[j]);
      end;
  for i:=0 to m do
    begin
   if (z[i-1,m div 2]<z[i,m div 2]) and (z[i+1,m div 2]<z[i,m div 2])
       then
      begin
      a:=x[i-1];
      b:=x[i+1];
      end;
    if (z[m-1,m div 2]<z[m,m div 2]) then
      begin
      a:=x[m];
      end;
   if (z[i-1,m div 2]=z[i,m div 2]) and (z[i,m div 2]>z[i+1,m div 2])
      then
      begin
      a:=x[i-1];
      b:=x[i];
      end;
    end;
  fa:=coef[0,0];
 for i:=1 to n do fa:=fa+coef[i,m div 2]*f(i,m div 2,x[i],y[m div 2]);
  fb:=coef[0,0];
 for i:=1 to n do fb:=fb+coef[i,m div 2]*f(i,m div 2,x[i],y[m div 2]);
  if ((b-a)<epsilon) and (abs(fb-fa)<ita) then
    begin
    xoptim:=(a+b)/2;
    writeln('Solutia optima este xoptim=',xoptim);
```

```
      gata:=true;
      readln;
      end;
  until gata;
  gata:=false;
  repeat
    pasy:=(d-c)/m;
    for i:=0 to m do y[i]:=c+i*pasy;
    for j:=0 to m do
      begin
      z[1,j]:=1;
      for k:=0 to n do
        for l:=0 to n do
          z[1,j]:=z[1,j]+coef[k,l]*f(k,l,xoptim,y[j]);
      end;
    for j:=0 to m do
      begin
      if (z[1,j-1]<z[1,j]) and (z[1,j+1]<z[1,j]) then
        begin
        c:=y[j-1]; d:=y[j+1];
        writeln('S-a modificat c=',c:10:4,' d=',d:10:4); readln;
        end;
      if (z[1,m-1]<z[1,m]) then
        begin
        c:=y[m];
        writeln('S-a modificat c=',c:10:4); readln;
        end;
      if (z[1,j-1]=z[1,j]) and (z[1,i]>z[1,j+1]) then
        begin
        c:=y[j-1]; d:=y[j];
        writeln('S-a modificat c=',c:10:4,' d=',d:10:4); readln;
        end;
      end;
    fc:=coef[0,0];
    for j:=1 to n do fc:=fc+coef[1,i]*f(1,j,xoptim,y[j]);
    fd:=coef[0,0];
    for j:=1 to n do fd:=fd+coef[1,j]*f(1,j,xoptim,y[j]);
    if ((d-c)<epsilon) and (abs(fd-fc)<ita) then
      begin
      yoptim:=(c+d)/2;
      writeln('Solutia optima este yoptim=',yoptim);
      gata:=true;
      readln;
      end;
  until gata;
end.
```

Programul a fost verificat experimental pentru dependența între timpul de debitare şi perechea de parametri de intrare formată din densitatea de curent pe spațiul de lucru şi viteza relativă $t_p=f(j, v_r)$ iar rezultatele au fost confirmate prin comparație cu cele obținute prin interpolare folosind metoda celor mai mici pătrate.

5 Concluzii

Aplicarea în practică a celor două metode de determinare a valorilor optime ale unei funcții de una sau două variabile, utilizată ca model matematic al dependenței între valorile de ieşire şi cele de intrare la prelucrarea prin EEC, permite obținerea valorilor optime ale parametrilor de intrare care duc la maximizarea rezultatelor.

Metoda a fost verificată prin comparare atât cu rezultatele altor metode de optimizare cât şi cu cele obținute experimental şi de aceea se poate aplica atunci când se caută regimul optim de funcționare al maşinii de prelucrare prin EEC.

Bibliografie

[Poc98] **Pocinog, Grigorie,** *Modele şi metode de simulare*, Editura Eurobit, Timişoara, 1998

[SL00] **Slavii, Heorge G, Luchin, Milenco,** *Modelare şi simulare*, Editura Eurostampa, Timişoara, 2000

[Vad77] **Văduva, Ion,** *Modele de simulare cu calculatorul*, Editura Tehnică, Bucureşti, 1977

[VSO83] **Văduva, Ion, Stoica, Marcel, Odăgescu, Ioan,** *Simularea proceselor economice*, Editura Tehnică, Bucureşti, 1983