

Fizică experimentală

Laborator 10

Victor E. Ambruș

*Facultatea de fizică, Universitatea de Vest din Timișoara,
Bd. Vasile Pârvan nr. 4, Timișoara, RO 300223, România*

19 decembrie 2017

Rezumat

Pe parcursul acestui laborator vor fi trecute în revistă următoarele tehnici de interpolare și extrapolare: regresia liniară, polinomul de interpolare Lagrange și interpolarea spline.

1 Datele experimentale

Să considerăm setul de date din tab. 1, pe care le putem introduce în **Maxima** cu următoarele comenzi:

```
valx:[0.0, 0.1, 0.2, 0.3, 0.4,0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$  
valf:[0.03,0.098,0.197,0.305,0.4,0.492,0.61,0.689,0.81,0.899,1.01]$
```

Putem reprezenta aceste date cu comanda:

```
plot2d([discrete, valx, valf],
```

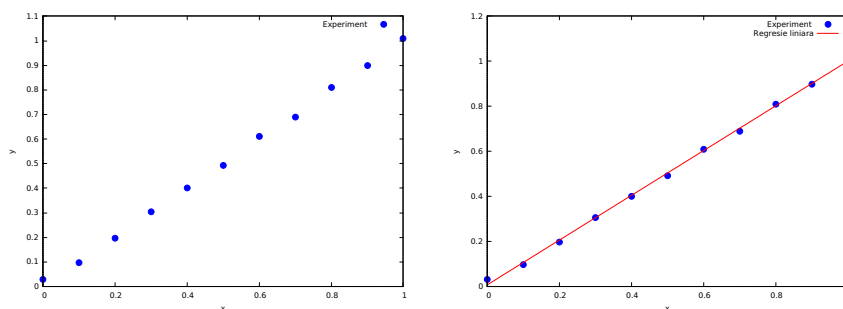


Figura 1: (a) Datele experimentale. (b) Dreapta de regresie liniară.

x	y
0.0	0.03
0.1	0.098
0.2	0.197
0.3	0.305
0.4	0.4
0.5	0.492
0.6	0.61
0.7	0.689
0.8	0.81
0.9	0.899
1.0	1.01

Tabela 1: Datele experimentale.

```
[legend, "Experiment"],
[style, points],
[xlabel, "x"],
[ylabel, "y"]);
```

Rezultatul poate fi vizualizat în fig. 1.

2 Regresia liniară

Regresia liniară este una dintre cele mai uzitate metode de interpolare și extrapolare a datelor experimentale. Regresia liniară are la bază metoda celor mai mici pătrate, putând fi extinsă cu ușurință la cazuri mai complexe (regresie pătratică, etc).

Regresia liniară pleacă de la presupunerea că dependența $y(x)$ este de tip liniar:

$$y(x) = ax + b. \quad (1)$$

În acest caz, metoda celor mai mici pătrate dă următoarele valori optime pentru a și b :

$$a = \frac{\overline{xy} - \bar{x} \bar{y}}{\overline{x^2} - (\bar{x})^2}, \quad b = \frac{\overline{x^2} \bar{y} - \bar{x} \overline{xy}}{\overline{x^2} - (\bar{x})^2}. \quad (2)$$

Pentru aflarea coeficienților a și b avem nevoie de calcularea unor medii. Calculul mediilor poate fi făcut utilizând comanda `mean` din pachetul `descriptive`. Începem prin încărcarea acestui pachet:

```
load (descriptive);
```

Comanda `mean` ia ca și parametru un șir sau o matrice și returnează media acestuia. Mediile lui `valx` și `valy` le vom stoca în variabilele `mx` și `my`:

```
mx:mean(valx); my:mean(valy);
```

Putem calcula celelalte medii din ec. (2) ținând cont că șirurile se înmulțesc element cu element utilizând operația `*`:

```
mxy:mean(valx*valy); mxx:mean(valx*valx);
```

Problema 1. Să se definească variabilele `rega` și `regb` având valorile lui a și b din ec. (2). [R: $a \simeq 0,993$, $b \simeq 0,007$]

Problema 2. Să se reprezinte grafic dreapta de regresie liniară împreună cu punctele experimentale.

Soluție:

```
plot2d([[discrete, valx, valy],
       rega*x+regb], [x,0,1],
       [legend, "Experiment", "Regresie liniara"],
       [style, points, lines],
       [xlabel, "x"],
       [ylabel, "y"]]);
```

Rezultatul poate fi consultat în fig. 1(b).

Metoda celor mai mici pătrate poate fi implementată și analitic. Pentru aceasta, definim funcția $S(\{x\}, \{y\})$:

$$S(\{x\}, \{y\}) = \sum_{i=1}^N [y_i - y(x)]^2, \quad (3)$$

aceasta reprezentând suma pătratelor distanțelor dintre valorile y_i măsurate experimental și cele obținute prin regresia liniară (1). Metoda celor mai mici pătrate caută valorile a și b pentru care $S(\{x\}, \{y\})$ are valoare minimă:

$$\frac{\partial S}{\partial a} = 0, \quad \frac{\partial S}{\partial b} = 0. \quad (4)$$

În Maxima, putem construi `valS` ca fiind valoarea funcției S (3):

```
valS : sum((valy[i] - (a * valx[i] + b))^2, i, 1, length(valx));
```

Mai sus am folosit funcția `sum`, care însumează valorile expresiei din primul argument obținute pentru toate valorile contorului de pe poziția a doua, care sunt cuprinse între limitele date pe pozițiile 3 și 4. Se vede că `valS` conține variabilele `a` și `b` la nivel simbolic.

Problema 3. Să se găsească valorile lui a și b rezolvând sistemul de ecuații $dS/da = 0$, $dS/db = 0$. [$a \simeq 0.993$, $b \simeq 0.007$]

3 Polinomul de interpolare Lagrange

Orice funcție care este determinată ca urmare a unui proces de măsurare are un număr finit N de perechi (x_k, y_k) , unde $k = 1, 2, \dots, N$, denumite puncte

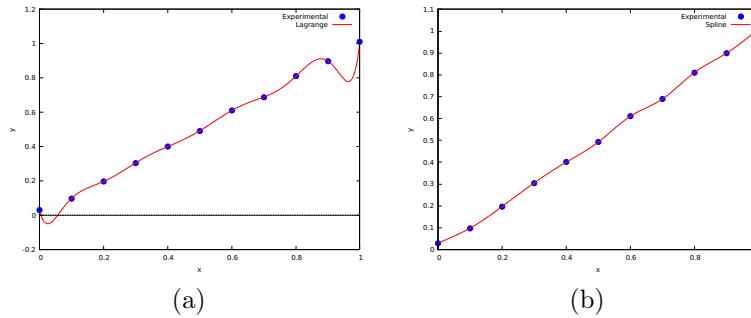


Figura 2: (a) Polinomul de interpolare Lagrange. (b) Interpolarea folosind funcții spline de ordinul 3 (cubice).

experimentale. Polinomul de interpolare Lagrange $L(x)$ se scrie:

$$L(x) = \sum_{k=1}^N y_k u_k(x), \quad (5)$$

unde $u_k(x)$ sunt polinoame de ordinul $N - 1$, definite prin:

$$u_k(x) = \prod_{j \neq k} \frac{x - x_j}{x_k - x_j}. \quad (6)$$

Se observă că $u_k(x_j) = 0$ pentru toate valorile lui j diferite de k (când $j \neq k$), în timp ce $u_k(x_k) = 1$. O proprietate specială a polinomului de interpolare Lagrange este că valoarea acestuia în punctele $x = x_k$ este cea măsurată experimental, adică $L(x_k) = y_k$.

3.1 Folosind pachetul `interp`

Maxima are deja implementată funcția care dă polinomul Lagrange pentru un set de puncte. În primul rând, trebuie încărcat pachetul `interp`, unde e definită această funcție:

```
load(interp)$
```

Funcția care ne interesează se numește `lagrange` și ia ca argument un singur parametru, structurat sub forma unei liste de perechi de valori (x_k, y_k) . Creăm această listă după cum urmează:

```
puncte : block([pct,k],
  pct:[],
  for k : 1 thru length(valx) do (
    pct : append(pct, [[valx[k], valy[k]]])
  ),
```

```

    pct
);

```

Mai sus am introdus structura `for`. Polinomul Lagrange se obține folosind:

```
lag : lagrange(puncte)$
```

Problema 4. Să se reprezinte grafic polinomul de interpolare Lagrange, împreună cu punctele experimentale. [Vezi fig. 2(a)]

3.2 Folosind definiția

În Maxima, aceste polinoamele $u_k(x)$ (6) se pot defini folosind funcția:

```

func_uk(valx,k) := block([uk, j],
    uk : 1,
    for j : 1 thru length(valx) do (
        if notequal(j, k) then uk : uk * (x - valx[j]) / (valx[k] - valx[j])
    ),
    return (uk)
)$

```

Polinomul de interpolare Lagrange (5) se scrie:

```

func_lag(valx, valy) := block([k, lag],
    lag : 0,
    for k : 1 thru length(valx) do (
        lag : lag + valy[k] * func_uk(valx, k)
    ),
    lag
)$

```

Problema 5. Să se verifice dacă rezultatul obținut mai sus coincide cu cel returnat de funcția `lagrange`.

4 Interpolarea cu funcții spline cubice

Pentru funcții nemonotone, interpolarea Lagrange poate duce la abateri importante ale funcției $L(x)$ față de $f(x)$. În astfel de cazuri, o aproximare mai bună se obține prin interpolarea cu ajutorul funcțiilor *spline*. Funcțiile spline de ordinul m reprezintă polinoame de ordinul m astfel construite încât derivatele acestora până la ordinul $m - 1$ să fie continue în punctele x_k (noduri).

Pentru construirea interpolării spline aferente unui set de $N = n + 1$ puncte experimentale, se utilizează un număr de n funcții S_k ($k = 1, 2, \dots, n$), fiecare fiind definită pe intervalul $[x_k, x_{k+1}]$ corespunzător.

În cazul funcțiilor spline cubice ($m = 3$), se folosește notația:

$$S_k(x) = p_k + q_k(x - x_k) + r_k(x - x_k)^2 + s_k(x - x_k)^3, \quad x \in [x_k, x_{k+1}]. \quad (7)$$

Pentru determinarea celor $4n$ coeficienți constanți p_k , q_k , r_k și s_k avem nevoie de $4n$ relații independente. În primul rând, funcțiile $S_k(x)$ trebuie să treacă prin punctele care mărginesc intervalul pe care îl interpolează:

$$S_k(x_k) = y_k, \quad S_k(x_{k+1}) = y_{k+1}. \quad (8)$$

Rezultă $2n$ ecuații. Alte $2n - 2$ ecuații se pot obține impunând ca derivatele de ordinul 1 și 2 să fie continue în punctele x_k ($2 \leq k \leq n$):

$$\begin{aligned} S'_k(x_{k+1}) &= S'_{k+1}(x_{k+1}), & \text{continuitatea pantei} \\ S''_k(x_{k+1}) &= S''_{k+1}(x_{k+1}), & \text{continuitatea pantei,} \end{aligned} \quad (9)$$

Adunând constrângerile enumerate mai sus, rezultă că rămânem cu două grade de libertate, care sunt fixate de condițiile:

$$\begin{aligned} S''_1(x_1) &= 0 & \Rightarrow r_1 &= 0, \\ S''_n(x_{n+1}) &= 0 & \Rightarrow s_n &= -\frac{r_n}{3h_n}, \end{aligned} \quad (10)$$

unde s-a introdus notația:

$$h_k = x_{k+1} - x_k, \quad 1 \leq k \leq n. \quad (11)$$

4.1 Folosind pachetul interpol

Folosind `puncte`, interpolul ne dă spline-ul folosind comanda:

```
spl : cspline(puncte)
```

Problema 6. Să se reprezinte grafic funcția de interpolare spline, împreună cu punctele experimentale. [Vezi fig. 2(b)]

4.2 Folosind gnuplot

În `gnuplot`, reprezentarea unui set de date folosind interpolarea cu funcții spline cubice se face folosind opțiunea `smooth mcsplines`:

```
plot "date.dat" u 1:2 w p ps 2 pt 7, "" w l lw 3 smooth mcsplines
```

4.3 Folosind recurența

Folosind $S_k(x_k) = y_k$, rezultă imediat:

$$p_k = y_k, \quad 1 \leq k \leq n. \quad (12)$$

Mai departe, condiția $S''_k(x_{k+1}) = S''_{k+1}(x_{k+1})$ permite eliminarea lui s_k :

$$s_k = \frac{r_{k+1} - r_k}{3h_k}, \quad 1 \leq k \leq n - 1, \quad (13)$$

unde $s_n = -r_n/3h_n$.

Din condiția $S_k(x_{k+1}) = y_{k+1}$ rezultă:

$$q_k = b_k + \frac{r_{k+1} + 2r_k}{3}h_k, \quad 1 \leq k \leq n-1, \quad (14)$$

unde s-a introdus notația:

$$b_k = \frac{y_{k+1} - y_k}{h_k}. \quad (15)$$

Când $k = n$, avem:

$$q_n = b_n - \frac{2}{3}r_n h_n. \quad (16)$$

În fine, condiția $S'_k(x_{k+1}) = S'_{k+1}(x_{k+1})$ impune:

$$h_{k+1}r_{k+2} + 2(h_k + h_{k+1})r_{k+1} + h_k r_k = 3(b_{k+1} - b_k), \quad 1 \leq k \leq n-2, \quad (17)$$

împreună cu condițiile:

$$r_1 = 0, \quad 2r_n(h_n + h_{n-1}) + h_{n-1}r_{n-1} = 3(b_n - b_{n-1}). \quad (18)$$

Ultima ecuație e compatibilă cu ec. (17) și (14) dacă se consideră $r_{n+1} = 0$ (această alegere este formală, întrucât funcțiile spline nu sunt definite pentru $k = n+1$, însă ne ajută la scrierea recurențelor). Drept urmare, coeficienții r_k se pot găsi rezolvând ecuația (17) cu $1 \leq k \leq n-1$, împreună cu condițiile $r_1 = r_{n+1} = 0$.

Dezavantajul ec. (17) este că condițiile la limită sunt în capetele opuse $k = 1$ și $k = n+1$. Deoarece r_k satisface astfel de condiții la limită, se poate arăta că ec. (17) este echivalentă cu o recurență cu doi termeni, care poate fi scrisă astfel:

$$h_k r_{k+1} + \alpha_k r_k = \beta_k, \quad 2 \leq k \leq n-1. \quad (19)$$

Pentru $k = 2$ avem:

$$h_2 r_3 + \alpha_2 r_2 = \beta_2. \quad (20)$$

Comparând cu ec. (17) pentru $k = 1$, rezultă condițiile inițiale pentru α_k și β_k :

$$\alpha_2 = 2(h_1 + h_2), \quad \beta_2 = 3(b_2 - b_1). \quad (21)$$

Scriind ec. (19) pentru $k \rightarrow k+1$ rezultă:

$$h_{k+1}r_{k+2} + \alpha_{k+1}r_{k+1} = \beta_{k+1}. \quad (22)$$

Înmulțind ec. (19) cu $[2(h_k + h_{k+1}) - \alpha_{k+1}]/h_k$ și adunând cu relația de mai sus, se obține:

$$\begin{aligned} h_{k+1}r_{k+2} + 2(h_k + h_{k+1})r_{k+1} + \frac{\alpha_k}{h_k}[2(h_k + h_{k+1}) - \alpha_{k+1}]r_k \\ = \beta_{k+1} + \frac{\beta_k}{h_k}[2(h_{k+1} + h_k) - \alpha_{k+1}]. \end{aligned} \quad (23)$$

Prin comparație cu ec. (17), deducem relațiile de recurență pentru α_k și β_k :

$$\alpha_{k+1} = 2(h_k + h_{k+1}) - \frac{h_k^2}{\alpha_k}, \quad \beta_{k+1} = 3(b_{k+1} - b_k) - \frac{h_k}{\alpha_k} \beta_k, \quad 2 \leq k \leq n-1. \quad (24)$$

După determinarea șirurilor α_k și β_k , șirul r_k se poate determina pornind de la $k = n$ (cunoscând $r_{n+1} = 0$) folosind ec. (19). Având șirul r_k , se pot construi s_k și q_k cu ajutorul ec. (13) și (14). În fine, ec. (12) implică $p_k = y_k$. Mai jos prezentăm pașii unui algoritm Maxima pentru construcția interpolării cu funcții spline cubice.

1. Se definește setul de numere $h_k = x_{k+1} - x_k$ ($1 \leq k \leq n$):

```
hk : block([hk],
hk : [],
for k : 1 thru length(valx) - 1 do (
hk : append(hk, [valx[k + 1] - valx[k]])),
hk)$
```

2. Se definește setul de numere $b_k = (y_{k+1} - y_k)/h_k$:

```
bk : block([bk],
bk : [],
for k : 1 thru length(valx) - 1 do (
bk : append(bk, [(valy[k + 1] - valy[k]) / hk[k]])),
bk)$
```

3. Mai departe, avem nevoie de șirul auxiliar α_k ($2 \leq k \leq n$), definit prin (24):

```
alphak : block([auxalpha, alphak],
auxalpha : 2 * (hk[2] + hk[1]),
alphak : [0, auxalpha],
for k : 3 thru length(hk) do (
auxalpha : 2 * (hk[k] + hk[k-1]) - hk[k-1]^2 / auxalpha,
alphak : append(alphak, [auxalpha])),
alphak)$
```

4. Al doilea șir auxiliar, β_k ($2 \leq k \leq n$), este definit tot prin (24):

```
betak : block([auxbeta, betak],
betak : [auxbeta : 0],
for k : 2 thru length(hk) do (
auxbeta : 3 * (bk[k] - bk[k - 1]) - auxbeta *
(2 * (hk[k] + hk[k-1]) - alphak[k]) / hk[k - 1],
betak : append(betak, [auxbeta])),
betak)$
```


5. În sfârșit, putem determina coeficienții r_k ($1 \leq k \leq n$) rezolvând ec. (19) pornind de la $k = n$, descendent către $k = 1$:

```
rk : block([auxr, rk],
  rk : [auxr : 0],
  for k : length(hk) thru 2 step -1 do (
    auxr : (betak[k] - auxr * hk[k]) / alphak[k],
    rk : append([auxr], rk)),
  append([0], rk))$
```

6. Coeficienții p_k se determină din condiția (12):

```
pk : valy$
```

7. Coeficienții s_k se determină din relația (13):

```
sk : block([sk],
  sk : [],
  for k : 1 thru length(hk) do (
    sk : append(sk, [(rk[k + 1] - rk[k]) / 3 / hk[k]])),
  sk)$
```

8. Coeficienții q_k se determină din relația (14):

```
qk : block([qk],
  qk : [],
  for k : 1 thru length(hk) do (
    qk : append(qk, [bk[k] - (2 * rk[k] + rk[k + 1]) * hk[k] / 3])),
  qk)$
```

După urmarea pașilor de mai sus, se poate scrie o funcție care evaluează funcția spline în orice punct:

```
spline_eval(x) := block([startk, dx],
  for k : 2 thru length(valx) do (
    startk : k - 1,
    if (x <= valx[k]) then return (startk)
  ),
  dx : x - valx[startk],
  pk[startk] + qk[startk] * dx + rk[startk] * dx^2 + sk[startk] * dx^3
)$
```

Mai departe, generăm o listă cu `npuncte` puncte în care evaluăm funcția spline:

```

npuncte:100$
puncte_spline : (block[auxpuncte, auxx],
  auxpuncte: [],
  for k : 1 thru (npuncte + 1) do (
    auxx : valx[1] + (valx[length(valx)] - valx[1]) * (k - 1) / npuncte,
    auxpuncte : append(auxpuncte, [[auxx, spline_eval(auxx)]])),
  auxpuncte
)$

```

În fine, afișăm capodopera:

```

plot2d([[discrete, valx, vally], [discrete, puncte_spline]],
  [x, valx[1], valx[length(valx)]],
  [legend, "Exp", "Spline"],
  [style, points, lines]);

```

Rezultatul poate fi vizualizat în fig. 2(b).