Îndrumar de Laborator

pentru

Programare în MATLAB

Ionelia PANEA

Editura
Universității
din București

ISBN 978-606-16-1232-1

Îndrumar de Laboratror

pentru

Programare în MATLAB

Ionelia PANEA

ISBN 978-606-16-1232-1

Îndrumar de Laborator

pentru

Programare în MATLAB

Ionelia PANEA

editura universităţii din bucureşti
2021

REFERENȚI ȘTIINȚIFICI

CS I dr. Bogdan Grecu
Institutul Național de Cercetare Dezvolatre pentru Fizica Pământului
Conf. dr. ing. Bogdan Mihai Niculescu
Universitatea din Bucureşti
CS II dr. Sorin Anghel
GeoEcoMar

© editura universității din bucurești
Șos. Panduri nr. 90-92, 050663 București – ROMÂNIA
Tel./Fax: +40 214102384
E-mail: editura.unibuc@gmail.com
www.editura-unibuc.ro
Librăria EUB
Bd. Regina Elisabeta nr. 4-12,
030018 București – ROMÂNIA
Tel. +40 213053703
Tipografie
Bd. Iuliu Maniu, Nr. 1–3,
061071, București – România
Tel. 0799.210.566

Copertă și DTP: LAURENȚIU MUȘOIU

ISBN 978-606-16-1232-1

Cuprins

Introducere

MATLAB (Matrix Laboratory) reprezintă un mediu de programare care este folosit în universități și în industrie pentru prelucrarea datelor seismice, electrice, magnetice, geofiyică de sondă ș.a.m.d. Analiza și prelucrarea datelor seismice se poate face folosind MATLAB după instalarea pachetelor de programe grupate în tollbox-urile CREWES și SEGYMAT. Exemple de programe folosite pentru prelucrarea și modelarea datelor seismice, cu aplicații mai mult în studiile de cercetare, sunt prezentate în Margrave (2003). Folosirea acestor programe necesită un nivel mediu de cunoaștere/utilizare a MATLAB-lui.

Acest îndrumar este destinat studenților din cadrul Facultății de Geologie și Geofizică, Departamentele de Geofizică și Inginerie Geologică, anul III, și conține o serie de lucrări al căror grad de dificultate variază de la nivelul de începător până la cel de avansat. Aceste lucrări au fost definite pentru a fi folosite în analiza și prelucrarea datelor seismice. Exemple de exerciții cu sau fără rezolvare afișată sunt prezentate la sfârșitul fiecărui capitol.

În Capitolul 1 sunt prezentate noțiuni introductive despre organizarea MATLAB-ului, despre comenzile de bază și specifice acestui program și despre funcțiile predefinite care pot fi folosite în scrierea programelor. Funcțiile folosite pentru definirea scalarilor, vectorilor și a matricelor sunt descrise în Capitolele 2 și 3. Modul în care sunt realizate reprezentările grafice ale vectorilor și matricelor este prezentat în Capitolul 4. Funcțiile folosite pentru încărcarea și salvarea datelor seismice în formatele SEG-Y și Seismic Unix (SU) sunt prezentate în Capitolul 5. Informațiile necesare construirii geometriei folosite în prelucrarea și modelarea datelor seismice pot fi încărcate sau salvate în formatul text; funcțiile folosite sunt prezentate în Capitolul 5. Rezultatele prelucrării sau modelării se analizează și reprezentând datele în domeniul spectral. Transformata Fourier este folosită pentru a se obține spectrele de amplitudine și de fază ale datelor seismice. În Capitolul 6 sunt prezentate exemple de funcții și programe folosite pentru a transforma datele seismice din domeniul timp în domeniul frecvență sau din domeniul timp-distanță în domeniul frecvență-număr de undă. Analiza și prelucrarea datelor seismice se poate face cu sau fără folosirea tranformatei Fourier. Exemple de aplicații sunt prezentate în Capitolul 7. O parte din aceste programe pot fi folosite pentru pregătirea datelor seismice folosite ca date de intrare în programele propuse de van der Burg et al. (2006). În Capitolul 8 sunt prezentate o serie de programe folosite pentru modelarea datelor seismice cu aplicații în proiectarea studiilor seismice active sau pasive și în analiza datelor seismice înregistrate folosind metoda activă. Funcțiile și programele descrise în Capitolul 9 sunt utile pentru analiza și folosirea unui set mare de date seismice. Analiza,

prelucrarea și salvarea fiecărei înregistrări în parte consumă destul de mult timp iar MATLAB-ul conține o serie de funcții care pot fi folosite pentru scrierea programelor ce permit citirea automată a fiecărei înregistrări în parte, prelucrarea acesteia și apoi salvarea automată în fișiere independente sau nu.

Capitolul 1
Introducere în utilizarea programului MATLAB

1.1 Lansarea în lucru a programului MATLAB

MATLAB este un program interactiv care poate fi folosit la prelucrarea numerică a semnalelor sau înregistrărilor de diferite tipuri. De exemplu, datele seismice pot fi încărcate în programul MATLAB dacă sunt salvate în formatul SEG-Y sau SU. MATLAB este un program simplu de învăţat şi este des utilizat în mediile universitare şi în industrie (SeisSpace - Halliburton).

Elementele de bază în MATLAB sunt vectorii şi matricele. Orice semnal analizat este convertit într-un vector sau o matrice, în funcţie de dimensiunea semnalului, 1D sau 2D. De exemplu, o trasă seismică va fi întotdeauna considerată un vector coloană ($N_t \times 1$), iar o înregistrare seismică va fi citită ca o matrice ($N_t \times N_x$). În aceste exemple, N_t reprezintă numărul de eşantioane de timp conţinute de trasa seismică, iar N_x reprezintă numărul de trase conţinute de înregistrarea analizată.

Analiza şi prelucrarea datelor seismice necesită instalarea, pe lângă programul MARLAB, a unor programe grupate în aşa-numitele toolbox-uri SEGYMAT şi CREWES.

Programul se lansează în execuţie cu dublu click pe iconiţa Matlab sau prin selectarea iconiţei şi apoi Matlab.exe; pe monitor vor apărea următoarele ferestre:
- Command Window: fereastra în care utilizatorul introduce comenzile;
- Command History: fereastra în care se păstrează un istoric al comenzilor folosite;
- Current Directory: arată directorul în care se lansează programul.

Din meniul principal se pot alege ferestrele de lucru, acestea se selectează din *View*; de obicei, se păstrează numai fereastra Command Window în care se introduc comenzile de către utilizator.

După lansarea în execuţie, programul MATLAB intră în *modul de comandă*, adică afişează prompter-ul >> în fereastra de comandă. Din acest moment, utilizatorul introduce comenzile al căror răspuns este reprezentat de *variabile*; numele acestora este definit de către utilizator. În cazul în care obţinerea anumitor variabile necesită folosirea mai multor comenzi, este utilă scrierea acestor comenzi în programe salvate sub forma fişierelor cu extensia **.m**. Aceste fişiere sunt de două feluri, *script* sau *funcţie (function)*.

Un fişier *script* este un fişier extern care conţine o secvenţă de comenzi MATLAB. Rularea acestui tip de fişier se face în două moduri, prin folosirea butonului "Run" sau prin selectarea tuturor comenzilor scrise în fişier, sau o parte din ele, şi selectarea folosind butonul din dreapta al mouse-ului (MB3) a comenzii "Evaluate selection". Rezultatul execuţiei acestui tip de fişier este reprezentat de un set de variabile

care rămân în zona de memorie a aplicaţiei; toate mărimile definite în *script* apar ca variabile după executarea lui.
Un exemplu de fişier script este prezentat mai jos:

```
% program care calculează raza primei zone Fresnel:
clear
% date de intrare:
f_min = 10 % Hz
pas = 0.2 % Hz
nf = 100
V = 1250 % m/s
timp = 1 % s

% calculul razei primei zone Fresnel
for ifreq=1:nf
    f(ifreq)=f_min+(ifreq-1)*pas;
    Rf(ifreq)=(V/2)*sqrt(timp/f(ifreq));
end
plot(f,Rf)
```

Dacă se foloseşte un fişier tip *funcţie (function)*, în urma execuţiei lui se obţin numai variabilele pe care definite ca ieşiri ale programului.

```
function [Rf]=fresnel_function(f_min,V,timp);

% date de intrare:
f_min = 10 % Hz
pas = 0.2 % Hz
nf = 100
V = 1250 % m/s
timp = 1 % s

% calculul razei primei zone Fresnel
for ifreq=1:nf
    f(ifreq)=f_min+(ifreq-1)*pas;
    Rf(ifreq)=(V/2)*sqrt(timp/f(ifreq));
end
plot(f,Rf)
```

Executarea acestui tip de fişier se poate face direct în fereastra de comandă folosind următoarea expresie:

```
[Rf]=fresnel_function(10,1250,1);
```

Directorul în care porneşte MATLAB-ul se numeşte *directorul curent*. Acesta se poate schimba temporar de către utilizator în timpul folosirii programului sau permanent prin MB3 pe iconiţa Matlab, se selectează *Properties*, se introduce numele noului director în care să pornească programul apoi se selectează OK. Efectul schimbării permanente apare la noua pornire a MATLAB-ului. Se pot face mai multe shortcut-uri în care se pot selecta diferiţi directori din care se doreşte pornirea programului; această opţiune se poate

folosi în cazul în care se lucrează cu seturi de programe definite pentru seturi de date diferite.

Exerciţii*:*

E1: Setati noul director curent din care să pornească MATLAB-ul.
E2: Folosind *View* afişaţi fiecare fereastră care apare în meniu.

1.2 Funcţii MATLAB

Funcţiile predefinite din MATLAB şi comenzile pot fi folosite direct în fereastra *Command Window*. Apariţia prompter-ului >> arată că programul este gata să primească comenzile utilizatorului. Comenzile de bază, cum ar fi trecerea de la un director la altul, definire de directoare noi, listarea conţinutului unui director, aflarea directorului curent, şi cele specifice acestui program sunt descrise în Tabelele 1.1, 1.2 şi 1.3. Dacă se foloseşte expresia " whos *nume de variabilă*", se obţin informaţii numai despre acea variabilă. Dacă se foloseşte " clear *nume de variabilă*", se şterge numai acea variabilă.

Tabelul 1.1: Exemple de comenzi de bază folosite în MATLAB

Comenzi de bază	
pwd	Calea spre directorul curent
Ls	Listarea fişierelor conţinute de director
cd	Schimb între directoare
mkdir	Definire director nou

Tabelul 1.2: Exemple de comenzi specifice MATLAB-ului

Comenzi specifice Matlab-ului	
whos	Listarea variabilelor definite de utilizator, a dimensiunii şi tipului acestora (reale sau complexe)
who	Listează variabilele definite de utilizator
clear	Ştergerea tuturor variabilelor definite de utilizator
help	Tastat împreună cu numele funcţiei predefinite aduce informaţii despre folosirea acesteia

Exerciţii cu rezolvare afişată:

E1: Definiţi numere aleatoare folosind funcţia `rand`.

```
>> rand(1,1)
ans =
    0.9058

>> rand(1,1)
ans =
    0.1270

>> rand(1,1)
ans =
    0.9134
```

Tabelul 1.3: Exemple de funcţii predefinite din Matlab

Funcţii predefinite	
rand	Definire număr aleator
round	Rotunjire număr aleator
ceil	Rotunjire număr aleator la valoare mai mare
floor	Rotunjire număr aleator la valoare mai mică
fliplr	Inversare vector linie/matrice pe orizontală
flipud	Inversare vector/matrice pe verticală
resamp	Reeşantionarea unei variabile la intervalul dorit
power	Ridicarea la putere
fft	Transformata Fourier 1D
fft2	Transformata Fourier 2D
zeros	Definire vector / matrice în care toate elementele au valoarea 0
ones	Definire vector / matrice în care toate elementele au valoarea 1
eye	Definire matrice în care elementele de pe diagonala principală au valoarea 1 şi în rest 0
load	Încarcă un fişier text
save	Salvează un fişier text
plot	Reprezentare grafică a unui vector linie sau coloană
imagesc	Reprezentare grafică 2D color
surf	Reprezentare grafică 3D color

E2: Definiţi numere aleatoare folosind funcţia `rand` şi apoi determinaţi numerele întregi corespunzătoare folosind funcţia `round`.

```
>> A=rand(1,1)
```

```
A =
    0.6324

>> round(A)
ans =
    1

>> A=rand(1,1)
A =
    0.0975

>> round(A)
ans =
    0
```

E3: Definiţi numere aleatoare folosind funcţia `rand` şi apoi determinaţi numerele întregi corespunzătoare folosind funcţia `ceil`.

```
>> A=rand(1,1)
A =
    0.5469

>> ceil(A)
ans =
    1

>> A=rand(1,1)
A =
    0.9575

>> ceil(A)
ans =
    1
```

E4: Definiţi numere aleatoare folosind funcţia `rand` şi apoi determinaţi numerele întregi corespunzătoare folosind funcţia `floor`.

```
>> A=rand(1,1)
A =
    0.9649

>> floor(A)
ans =
    0

>> A=rand(1,1)
A =
    0.1576
```

```
>> floor(A)
ans =
    0
```

E5: Definiţi numere întregi şi aleatoare folosind funcţia `rand` apoi ridicaţi la puterea a 3-a folosind funcţia `power`.

```
>> A=2
A =
    2

>> B=3
B =
    3

>> C=power(A,B)
C =
    8

>> A=rand(1,1)
A =
    0.9706

>> B=3
B =
    3

>> C=power(A,B)
C =
    0.9143
```

E6: Definiţi matrice de ordinul 2 folosind funcţiile `zeros`, `ones` şi `eye`:

```
>> zeros(2,2)
ans =
     0     0
     0     0

>> ones(2,2)
ans =
     1     1
     1     1

>> eye(2,2)
ans =
     1     0
     0     1
```

Capitolul 2
Definire şi operaţii cu scalari, vectori şi matrice

2.1 Definirea variabilelor

Variabilele pot fi definite direct în fereastra *Command Window* de către utilizator sau ca urmare a executării unui fişier. Numele de variabile sunt case-sensitive (contează litera mare şi mică). Funcțiile predefinite se scriu numai cu literă mică.

Variabilele pot fi numere naturale, complexe, vectori sau matrice. Acestea pot fi definite cu afişare dacă nu se pune punct şi virgulă la definirea lor, indiferent de locul unde se face aceasta, direct în fereastra de comandă sau într-un fişier *script* sau *function.*

Variabila definită automat de mediul MATLAB este "ans" şi în ea este returnat rezultatul executării unei expresii în care nu a fost definită o variabilă. Există o serie de variabile predefinite în MATLAB cu caracter permanent şi anume:
- pi = 3.1416,
- i este variabila folosită pentru scrierea numerelor complexe,
- j este variabila alternativă la unitatea imaginară,
- inf este variabila folosită pentru reprezentarea lui plus infinit ca rezultat al împărţirii 1.0 / 0.0,
- nan este variabila folosită pentru reprezentarea lui not-a-number ca rezultat al împărţirii 0.0 / 0.0,
- realmax şi realmin reprezintă cea mai mare şi cea mai mică valoare pozitivă în virgulă mobilă.

Pentru definirea variabilelor se pot folosi operatorii aritmetici de bază, funcţii predefinite şi caractere specifice MATLAB-ului (vezi Tabelele 2.1, 2.2 şi 2.3). În cazul în care se lucrează cu vectori şi matrice, atenţie trebuie acordată dimensiunilor acestora:

```
A = [1 2 3 4]
B = [1 2 4]

A+B

A =
     1     2     3     4

B =
     1     2     4

??? Error using ==> plus
Matrix dimensions must agree.
```

Tabelul 2.1: Operatori aritmetici de bază folosiţi în definirea variabilelor

Operatori aritmetici	*Simbol*
Adunare	+
Scădere	-
Multiplicare	*
Împărţire	/
Ridicare la putere	^
Transpusă	'

Tabelul 2.2: Exemple de funcţii predefinite folosite în definirea variabilelor

Functii predefinite	*Simbol*
Valoare absolută	abs
Faza ataşată unui număr complex (radiani)	angle
Partea reală a unui număr complex	real
Partea imaginară a unui număr complex	imag
Rădăcina pătrată	sqrt
Exponenţială	exp
Logaritm	log
Funcţia sin	sin
Funcţia cos	cos
Transformata Fourier 1D	fft
Transformata Fourier 2D	fft2

Tabelul 2.3: Exemple de caractere speciale folosite în definirea variabilelor

Caractere speciale din Matlab	*Simbol*
Generarea diviziunilor	:
Definirea corpului unei funcţii	( )
Definirea unui vector / matrice	[ ]
Indicator de operaţie aritmetică element cu element	.
Continuarea unei comenzi Matlab pe linia de comandă următoare	...
Separator între instrucţiuni pe aceeaşi linie de comandă (cu rezultat pe ecran)	,
Separator între instrucţiuni pe aceeaşi linie de comandă (fără rezultat pe ecran)	;
Introducerea de comentarii în scrierea programului	%
Separator între liniile unei matrice	[;]

Exercitii cu rezolvare afişată:

E1: Definiţi variabila `A` direct în Command Window.

Se tastează direct în fereastra de comandă, la prompter:

```
>> A = 1-i * 2
```

se apasă tasta Enter şi se obţine:

```
A =
   1.0000 - 2.0000i
```

Răspunsul nu mai este afişat dacă se scrie punct şi virgulă când se defineşte variabila `A`:

```
>> A = 1-i * 2;
>>
```

E2: Calculaţi valoarea absolută a variabilei `A` cu afişare pe ecran.

Se tastează direct în fereastra de comandă, la prompter, apoi Enter:

```
>> B=abs(A)
B =
    2.2361
```

E3: Calculaţi faza variabilei `A` cu afişare pe ecran.

Se tastează direct în fereastra de comandă, la prompter, apoi Enter:

```
>> F = angle(A)
F =
    -1.1071
```

E4: Calculaţi partea reală şi cea imaginară a variabilei complexe `A` cu afişare pe ecran.

Se tastează direct în fereastra de comandă, la prompter, apoi Enter:

```
>> R=real(A)
R =
    1

>> C=imag(A)
C =
```

```
    -2
```

E5: Definiţi variabila `nx` cu afisare pe ecran; `nx` poate reprezenta numărul de geofoane active de pe profilul de observaţie.

Se tastează direct în fereastra de comandă, la prompter, apoi Enter:

```
>> nx = 80
```

şi rezultă:

```
nx  =
    80
```

E6: Definiţi variabila `nx` fără afişare în fereastra de comandă.

Se tastează direct în fereastra de comandă, la prompter, apoi Enter:

```
>> nx = 80;
```

şi rezultă:

```
>>
```

E7: Definiţi variabila `dt` în secunde sub ambele forme, cu sau fără afişare în fereastra de comandă.

- cu afişare în fereastra de comandă:

```
>> dt = 0.001
>> dt  =
    0. 001
```

- fără afişare în fereastra de comandă:

```
>> dt = 0.001;
>>
```

E8: Multiplicaţi două variabile, `dt` şi `nt`, fără salvarea rezultatului într-o variabilă definită; folosiţi editorul de text şi adăugaţi comentarii pentru a explica semnificaţia variabilelor.

Se scrie în editorul de text şi apoi se execută cu *Evaluate selection*:

```
dt = 0.001; % interval de eșantionare în timp
nt = 501; % număr de eșantioane de timp
dt*nt
```

iar răspunsul este afișat în *Command Window*:

```
>>
dt = 0.001; % interval de eșantionare în timp
nt = 501; % număr de eșantioane de timp
dt*nt

ans =
    0.5010
>>
```

E9: Calculaţi distanţa, `dist`, dintre două puncte cu coordonate cunoscute cu afișare pe ecran.

Definiţi variabilele:

```
x1 = 10
x2 = 15
y1 = 2
y2 = 4
dist = sqrt((x2-x1)^2+(y2-y1)^2)
```

iar rezultatul este:

```
x1 =
    10
x2 =
    15
y1 =
    2

y2 =
    4
dist =
    5.3852
```

E10: Calculaţi valoarea coeficientului de reflexie, `R`, folosind variabilele:

```
rho1=0.2
rho2=0.18
V1 = 2000
V2 = 2400
```

```
R = (rho2*V2-rho1*V1)/(rho2*V2+rho1*V1)
```

Rezultatul este:

```
R =
    0.0385
```

E11: Calculaţi sinus, cosinus şi tangentă pentru un unghi de 45^o. Funcţiile trigonometrice se folosesc pentru unghiuri definite în radiani:

```
>> rad_45=(45*3.14)/180
rad_45 =
    0.7850

>> sin_i = sin(rad_45)
sin_i =
    0.7068
```

Funcţia cosinus se determină ştiind sinus:

```
cos_i = sqrt(1-sin_i^2)
```

Funcţia tangentă se determină folosind sinus şi cosinus:

```
tg_i = sin_i/cos_i.
```

E12: Definiţi variabila `dx_rand` ca rezultat al expresiei `dx_reg + rand(1,1)`, ştiind că `dx_reg = 5`. Variabila `dx_reg` poate reprezenta distanţa dintre două geofoane alăturate.

```
>> dx_reg = 5;
>> dx_rand = dx_reg + rand(1, 1)
dx_rand =
    5.8214
```

Aplicarea funcţiei `rand` poate avea drept rezultat un scalar, un vector sau o matrice.

- un scalar:

```
>> rand(1, 1)
ans =
    0.0925
```

- un vector coloană:

```
>> rand(2,1)
ans =
```

```
    0.2311
    0.6068
```

- un vector linie:

```
>> rand(1,2)
ans =
    0.8147    0.9058
```

- o matrice:

```
>> rand(2, 2)
ans =
    0.4860     0.7621
    0.8913     0.4565
```

Funcţia `rand` poate fi folosită pentru a se obţine valori pozitive sau negative:

```
>> 2*rand(1,1)-1
ans =
    0.8268

>> 2*rand(1,1)-1
ans =
    0.2647

>> 2*rand(1,1)-1
ans =
    -0.8049
```

Dacă se doreşte obţinerea unor numere limitate de o valoare maximă, `err_max`, se foloseşte:

```
>> err_max=0.5
err_max =
    0.5000

>> (2*rand(1,1)-1)*err_max
ans =
    0.3147

>> (2*rand(1,1)-1)*err_max
ans =
    0.4058

>> (2*rand(1,1)-1)*err_max
ans =
    -0.3730
```

În mod asemănător se pot obţine vectori şi matrice cu elemente ale căror valori sunt aleatoare, pozitive şi negative, limitate sau nu de o valoare maximă.

E13: Rotunjiți variabila `dx_rand` definită la exercițiul anterior folosind funcţia predefinită `round`:

```
>> round(dx_rand)
ans   =
    6
```

E14: Rotunjiţi variabila `dx_rand` la numărul întreg mai mic decât valoarea ei.

```
>> floor(dx_rand)
ans =
    5
```

E15: Rotunjiţi variabila `dx_rand` la numărul întreg mai mare decât valoarea ei.

```
>> ceil(dx_rand)
ans =
    6
```

2.2 Definirea scalarilor, vectorilor şi a matricelor

MATLAB-ul conţine un pachet de programe care lucrează cu matrice numerice rectangulare ale căror elemente pot fi numere reale sau complexe. Astfel, scalarii sunt asimilaţi matricelor cu o linie şi o coloană (`1 × 1`), vectorii sunt asimilaţi matricelor cu o linie (`1× n`) sau o coloană (`n × 1`).

Se numește matrice cu `m` linii şi `n` coloane un tablou cu `m` linii şi `n` coloane ale cărui elemente sunt numere reale sau complexe. O matrice cu o linie şi `n` coloane se mai numeşte matrice linie; o matrice cu `m` linii şi o coloană se mai numeste matrice coloană. Dacă numărul de coloane este egal cu numărul de linii, atunci matricea se numeşte matrice pătratică. O matrice se numeşte nulă (zero) dacă toate elementele ei sunt zero. Matricea în care toate elementele de pe diagonala principală sunt egale cu 1, iar în rest sunt egale cu 0, se numeşte matricea unitate.

MATLAB-ul conţine funcţii care definesc astfel de matrice. De exemplu, ştiind valorile pentru `m` şi `n`, matricea cu toate elementele zero se obţine folosind funcţia predefinită `zeros(m, n)`. Matricea cu toate elementele de pe diagonala principală egale cu 1 se obţine folosind funcţia `eye(m, n)`. Matricea cu toate elementele egale cu 1 se

obţine folosind funcţia `ones(m, n)`. Definirea unei matrice se poate face în mai multe feluri. Cel mai simplu mod este de a introduce termenii direct în fereastra de comandă:

```
M = [ 1 2 3 4; 1 2 3 4; 1 2 3 4]
```

ce va avea ca rezultat:

```
M =
         1      2      3      4
         1      2      3      4
         1      2      3      4
```

Acest mod de definire este util în cazul în care se lucrează cu matrici de dimensiuni mici. Atunci când trebuie definite matrici de dimensiuni mari, de exemplu cu sute de elemente pe coloană şi/sau linie, se foloseşte comanda specială în MATLAB " : " sau functia `for-loop`, care va fi prezentată în subcapitolul 3.1.2. Parametrii care trebuie introduşi înainte de definirea unei matrice sunt numărul maxim de elemente pe coloană, `ncol`, şi pe linie, `nlinie`.

O matrice cu două linii şi număr de coloane egal cu `ncol` poate fi definită folosind " : " astfel:

```
>> ncol = 40;
>> M = [1:1:ncol;2:2:2*ncol]
M =
Columns 1 through 18
  1  2  3  4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  2  4  6  8  10  12  14  16  18  20  22  24  26  28  30  32  34  36
Columns 19 through 36
19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
38  40  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72
Columns 37 through 40
37  38  39  40
74  76  78  80
```

Introducerea scalarilor, vectorilor şi a matricelor se poate face direct în fereastra de comandă sau cu ajutorul fisierelor *script* sau *function*.

Exercţii cu rezolvare afişată:

E1: Introduceţi un scalar, `nx`, în fereastra de comandă cu afișarea rezultatului:

```
>> nx = 100
nx =
```

```
   100
```

E2: Definiţi un vector coloană, `Vc`, cu elementele (1 2 3 4):

```
>> Vc = [1; 2; 3; 4]
Vc =
     1
     2
     3
     4
```

şi un vector linie, `Vl`, cu elementele (1 2 3 4):

```
>> Vl = [1 2 3 4]
Vl =
     1     2     3     4
```

E3: Definiţi un vector `dist` ce reprezintă distanţa acoperită de un dispozitiv de înregistrare folosind variabilele `dx = 5`, reprezentând distanţa dintre două geofoane alăturate exprimată în metri, şi `nx = 30`, număr de geofoane. Calculaţi poziţia fiecărui receptor din dispozitivul de înregistrare, `dist_g`.

```
>> dx = 5;
>> nx = 30;
>> dist = dx*(nx-1)

dist =
   145

>> dist_g = dx*(1:nx)

dist_g =
Columns 1 through 17
5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85
Columns 18 through 30
90  95  100  105  110  115  120  125  130  135  140  145  150
```

E4: Definirea matricelor folosind funcţiile predefinite:

```
>> eye(3,3)
ans =
     1     0     0
     0     1     0
     0     0     1

>> ones(3,3)
```

```
ans =
     1     1     1
     1     1     1
     1     1     1

>> zeros(3,3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

Exerciţii fără rezolvare afişată:

E1: Ştiind valorile pentru `dt = 0.001`, `nt = 501`, `dx = 5`, `nx = 20`, calculaţi următoarele variabile:

Scalar: `timp = dt*nt`
Vector: `timp = dt*(1:nt)`
Scalar: `dist = dx*nx`
Vector: `dist = dx*(1:nx)`
Scalar: `fn = 1/(2*dt)  % frecvenţa Nyquist`
Scalar: `kn = 1/(2*dx) % numărul de undă Nyquist`

E2: Ştiind valorile pentru modulul lui Young, `E`, şi raportul lui Poisson, `υ`, calculaţi:

Modulul de incompresibilitate : $K = E / 3(1-2\upsilon)$.
Primul parametru Lamé : $\lambda = E\upsilon / (1+ \upsilon)(1-2\upsilon)$
Modulul de forfecare : $G = E / 2(1+ \upsilon)$.

Gresie: $E = 2.0*10^{11}$, $\upsilon = 0.25$
Marna : $E = 3.0*10^{11}$, $\upsilon = 0.35$
Granit: $E = 3.3*10^{11}$, $\upsilon = 0.28$
Cuartit : $E = 7.0*10^{11}$, $\upsilon = 0.23$

E3: Calculaţi sinus, cosinus şi tangentă pentru unghiurile 15, 30, 45, 75 şi 90.

E4: Calculaţi sinus, cosinus şi tangenta unui unghi critic folosind vitezele de propagare cu valorile `V1 = 2000` m/s şi `V2 = 2400` m/s.

2.3 Operaţii cu vectori şi matrice

Dimensiunile vectorilor şi matricelor implicate în diverse operaţii aritmetice sunt foarte importante; de exemplu, nu se poate aduna un vector care are trei elemente cu unul care are patru elemente.

```
V1 = [1 2 3 4]
V2 = [1 2 3]

>> V1+V2
??? Error using ==> +
Matrix dimensions must agree.

>> V1-V2
??? Error using ==> -
Matrix dimensions must agree.

>> V1*V2
??? Error using ==> *
Inner matrix dimensions must agree.

>> V1'*V2    ! V1' este transpusa lui V1
ans =
     1     2     3
     2     4     6
     3     6     9
     4     8    12

>> V1*V2'   !!!! V2' este transpusa lui V2
??? Error using ==> *
Inner matrix dimensions must agree.
```

Aceeaşi regulă se aplică şi în cazul operaţiilor cu matrice. De exemplu, adunarea a două matrice se face numai dacă au aceeaşi dimensiune. Dacă se definesc matricele `M1(2×3)`, `M2(2×3)` şi `M3(3×3)`, se pot aduna matricele `M1` şi `M2` dar nu se pot aduna `M3` cu `M1` sau cu `M2` datorită mărimii lor diferite, număr de linii diferit. La fel se întâmplă şi în scăderea a două matrice.

```
M1 = [1 2 3; 1 2 3]
M2 = [2 3 4; 2 3 4]
M3 = [ 1 2 3; 2 3 4; 3 4 5]

M = M1+M2
N = M1+M3
O = M2+M3
```

```
M1 =
     1     2     3
     1     2     3

M2 =
     2     3     4
     2     3     4

M3 =
     1     2     3
     2     3     4
     3     4     5

M =
     3     5     7
     3     5     7

??? Error using ==> plus
Matrix dimensions must agree.
```

Multiplicarea directă a matricelor `M1` şi `M2` nu este posibilă; în schimb, se pot multiplica dacă se foloseşte transpusa uneia dintre ele, de exemplu `M2'`.

```
M1 = [1 2 3; 1 2 3]
M2 = [2 3 4; 2 3 4]

M = M1*M2'

M1 =
     1     2     3
     1     2     3

M2 =
     2     3     4
     2     3     4

M =
    20    20
    20    20
```

Determinantul unei matrice, `M`, se calculează folosind expresia `det(M)` . Inversa unei matrice, `M`, se obține scriind `inv(M)` .

```
M = [1 2 3; 1 2 3]
det_M = det(M)
```

```
M =
     1     2     3
     1     2     3

??? Error using ==> det
Matrix must be square.

M = [ 1 5 3; 4 3 1; 1 9 5]

det_M = det(M)
inv_M = inv(M)

M =
     1     5     3
     4     3     1
     1     9     5

det_M =
    10

inv_M =
    0.6000    0.2000   -0.4000
   -1.9000    0.2000    1.1000
    3.3000   -0.4000   -1.7000
```

Împărţirea a două matrice `M` şi `N` se efectuează în mai multe moduri:

- Împărţirea la dreapta, `M / N`, este soluţia sistemului liniar compatibil determinat `X * M = N`, adică `N * inv(M)`, unde `inv(M)` este inversa matricei `M`, calculată în MATLAB folosind expresia `inv(M)`.
- Împărţirea la stânga, `M \ N`, este soluţia sistemului liniar compatibil determinat `M * X = N`, adică `X = inv(M) * N`.
- Împărţirea element cu element la dreapta, `M./N`, se realizează împărţind elementele de pe aceeaşi poziţie.
- Împărţirea element cu element la stânga, `M.\N`, se face împărţind elementele de pe aceeaşi poziţie.

Ridicarea la putere, M^a, se obţine scriind expresia `M^a`, unde a > 1. Ridicarea la putere element cu element se face folosind `M.^a`.

Transpusa unei matrice se calculează folosind `M'`.

Două matrice sunt egale dacă sunt de acelaşi tip, adică au acelaşi număr de elemente pe linii şi coloane, şi dacă elementele lor sunt egale.

Capitolul 3
Folosirea funcţiilor for-end şi if/elseif/else-end

3.1 Definirea vectorilor şi a matricelor folosind funcţia for-end

3.1.1 Definirea vectorilor

Ştiind dimensiunea şi valorile elementelor unui vector, acesta poate fi definit folosind:

```
nx = 21 % număr de geofoane
dxg = 5 % distanţa între geofoane exprimată în metri
for ix=1:nx
    dist(ix)=(ix-1)*dxg;
end
dist

dist =
Columns 1 through 17
0  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80
Columns 18 through 21
85  90  95  100
```

Aceste valori pot însemna poziţiile a 21 geofoane plasate pe un profil seismic liniar într-un dispozitiv cu împuşcare la capăt.

Un vector care conţine elementele `-100:5:100` poate fi definit astfel:

```
clear
nx = 41
dxg = 5 % metri
for ix=1:nx
    dist(ix)=(ix-1)*dxg-((nx-1)*dxg)/2;
end
dist

dist =
Columns 1 through 14
-100  -95  -90  -85  -80  -75  -70  -65  -60  -55  -50  -45  -40  -35
Columns 15 through 31
-30  -25  -20  -15  -10  -5  0  5  10  15  20  25  30  35  40  45  50
Columns 32 through 41
55  60  65  70  75  80  85  90  95  100
```

Aceste valori pot reprezenta poziţiile geofoanelor într-un dispozitiv cu împuşcare la centru.

Un vector care conţine elementele `30:5:230` poate fi definit astfel:

```
clear
nx = 41
dxg = 5 % metri
ofset = 30 % metri
for ix=1:nx
    dist(ix)=(ix-1)*dxg+ofset;
end
dist

dist =
Columns 1 through 17
30  35  40  45  50  55  60  65  70  75  80  85  90  95  100  105  110
Columns 18 through 31
115  120  125  130  135  140  145  150  155  160  165  170  175  180
Columns 32 through 41
185  190  195  200  205  210  215  220  225  230
```

Aceste valori pot reprezenta poziţiile geofoanelor într-un dispozitiv cu împuşcare cu un offset de 30 metri.

În cazul în care distanţele dintre `nx` geofoane variază cu o eroare pozitivă maximă cunoscută, de exemplu `err_max = 0.1`, valoare care înseamnă 10% din distanţa regulată dintre geofoane, `dxg = 1`, se foloseşte următorul program:

```
clear
nx = 41
dxg = 1 % metri
err_max = 0.1
for ix = 1:nx
    dist(ix) = (ix-1)*dxg+rand(1,1)*err_max;
end
dist

dist =
Columns 1 through 9
0.0815  1.0906  2.0127  3.0913  4.0632  5.0098  6.0278  7.0547  8.0958
Columns 10 through 17
9.0965  10.0158  11.0971  12.0957  13.0485  14.0800  15.0142  16.0422
Columns 18 through 25
17.0916  18.0792  19.0959  20.0656  21.0036  22.0849  23.0934  24.0679
Columns 26 through 33
25.0758  26.0743  27.0392  28.0655  29.0171  30.0706  31.0032  32.0277
Columns 34 through 41
33.0046  34.0097  35.0823  36.0695  37.0317  38.0950  39.0034  40.0439
```

Dacă eroarea adăugată distanţei regulate dintre geofoane este negativă se scrie:

```
clear
nx = 41
dxg = 1 % metri
err_max = 0.1
for ix = 1:nx
    dist(ix) = (ix-1)*dxg - rand(1,1)*err_max;
end
dist

dist =
Columns 1 through 10
   -0.0382    0.9234    1.9205    2.9813    3.9510    4.9554    5.9354
6.9291    7.9245    8.9724
  Columns 11 through 20
    9.9320   10.9345   11.9837   12.9881   13.9502   14.9040   15.9660
16.9415   17.9776   18.9249
  Columns 21 through 30
   19.9745   20.9494   21.9301   22.9109   23.9041   24.9453   25.9861
26.9851   27.9742   28.9159
  Columns 31 through 40
   29.9746   30.9186   31.9756   32.9071   33.9650   34.9803   35.9749
36.9384   37.9527   38.9648
  Column 41
   39.9169
```

Dacă eroarea adăugată distanţei regulate dintre geofoane este negativă şi pozitivă se scrie:

```
clear
nx = 41
dxg = 1 % metri
err_max = 0.1
for ix = 1:nx
    dist(ix) = (ix-1)*dxg + (2*rand(1,1)-1)*err_max;
end
dist

dist =
  Columns 1 through 10
    0.0171    1.0099    2.0834    2.9572    4.0514    5.0507    5.9761
7.0136    7.9152    8.9108
  Columns 11 through 20
   10.0062   11.0558   12.0868   12.9260   14.0138   14.9939   15.9024
16.9674   17.9324   19.0589
  Columns 21 through 30
   19.9622   21.0057   21.9331   23.0204   23.9526   25.0308   26.0378
27.0496   27.9901   28.9168
  Columns 31 through 40
```

```
   29.9458    31.0827    31.9305    33.0652    34.0077    35.0992    35.9156
36.9885   37.9213   39.0924
  Column 41
   39.9009
```

Un vector care conţine poziţiile geofoanelor într-un dispozitiv cu împuşcare centrală cu distanţe neregulate între geofoane cu o eroare maximă de +/-10% din distanţa regulată `dxg = 5` poate fi definit astfel:

```
clear
nx = 21
dxg = 5 % metri
err_max=0.1
for ix=1:nx
    dist(ix)=(ix-1)*dxg-((nx-1)*dxg)/2+(2*rand(1,1)-1)*err_max;
end
dist

dist =
  Columns 1 through 10
  -49.9450  -44.9365  -39.9263  -35.0831  -30.0200  -25.0480  -19.9400
-15.0137   -9.9179   -5.0636
  Columns 11 through 20
   -0.0472    4.9291    9.9272   15.0739   20.0159   25.0100   29.9290
35.0706   40.0244   44.9702
  Column 21
   50.0026
```

3.1.2 Definirea matricelor

În definirea unei matrice, funcţia `for` se foloseşte pentru definirea liniilor şi coloanelor din matricea dorită. O matrice, `M`, cu două linii şi 40 coloane se poate defini astfel:

```
ncol = 40;
nlinie = 2;

for ilinie=1:nlinie
    for icol=1:ncol
        M(ilinie,icol)=icol*ilinie;
    end
end
M

M =
```

```
Columns 1 through 11
   1     2     3     4     5     6     7     8     9    10    11
   2     4     6     8    10    12    14    16    18    20    22

Columns 12 through 22
  12    13    14    15    16    17    18    19    20    21    22
  24    26    28    30    32    34    36    38    40    42    44

Columns 23 through 33
  23    24    25    26    27    28    29    30    31    32    33
  46    48    50    52    54    56    58    60    62    64    66

Columns 34 through 40
  34    35    36    37    38    39    40
  68    70    72    74    76    78    80
```

Valoarea fiecărui element din matrice se calculează folosind expresiile matematice corespunzătoare.

Exercitii fără rezolvare afişată:

E1: Definiţi un vector ce reprezintă distanţa acoperită de un dispozitiv de înregistrare cu `nx` geofoane spaţiate la distanţa `dxg` folosind `for-loop`. Calculaţi distanţe regulate, distanţe neregulate cu o eroare maximă de 25% din valoarea regulată `dxg`, cu erori pozitive şi negative şi numai cu erori pozitive sau cu erori negative.

E2: Calculaţi vectorul `dist` pentru care valorile elementelor variază de la `-500` la `500` cu un pas regulat `dxg = 5` şi neregulat cu o eroare maximă de 10% din `dxg`.

E3: Calculaţi timpii de propagare ai undelor longitudinale folosind următoarele relaţii:

```
t = dist/V
t = (1/V1)*sqrt(dist^2 + 4*h^2)
t = (1/V1)*sqrt(dist^2+4*h^2+4*h*dist*sin_fi)
```

şi variabilele:

```
V = 450 % m/s
V1 = 1450 % m/s
nx =  72
dxg = 2.5 % metri
h = 100 % metri
fi = 30 % unghi exprimat în grade.
```

Efectuaţi calculele considerând variabila `dist` un scalar sau un vector cu număr de elemente egal cu `nx`.

E4: Definiţi o matrice în care valoarea fiecărui element este egală cu suma poziţiei lui pe linie şi coloană.

E5: Definiţi o matrice cu 10 coloane şi trei linii în care valorile de pe prima linie să fie 1-10, de pe a doua linie 11-20 şi a treia linie 21-30.

E6: Definiţi o matrice cu cinci coloane şi trei linii în care prima linie începe cu 1, a doua cu 11 şi a treia cu 21. Diferenţa dintre elementele de pe fiecare linie este 2.

E7: Definiţi o matrice cu cinci linii şi șase coloane în care elementele au valori mai mari sau mai mici decat 10 cu o eroare maximă de 50% din 1.

3.2 Definirea vectorilor şi a matricelor folosind funcţiile for-end şi if/elseif/else-end

3.2.1 Definirea vectorilor

Funcţiile `if`, `if-else` şi `if-elseif-else` se folosesc pentru a modifica o singură valoare sau mai multe valori dintr-un vector.

Modificarea valorii unui singur element dintr-un vector se poate face folosind funcţia `if`:

```
clear
nx = 21
dxg = 2.5 % metri

for ix = 1:nx
    if ix == 15
        dist(ix) = 0.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =

Columns 1 through 7
  2.5000    5.0000    7.5000   10.0000   12.5000   15.0000   17.5000
Columns 8 through 14
  20.0000   22.5000   25.0000  27.5000   30.0000   32.5000   35.0000
Columns 15 through 21
  0   40.0000   42.5000   45.0000   47.5000   50.0000   52.5000
```

Pentru a modifica valorile a două elemente dintr-un vector se foloseşte `if-elseif`:

```
clear
nx = 21
dxg = 2.5 % metri

for ix = 1:nx
    if ix == 15
        dist(ix) = 0.0;
    elseif ix == 20
        dist(ix) = 0.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =

Columns 1 through 7
2.5000    5.0000    7.5000   10.0000   12.5000   15.0000   17.5000
Columns 8 through 14
20.0000   22.5000   25.0000  27.5000   30.0000   32.5000   35.0000
Columns 15 through 21
0   40.0000   42.5000   45.0000   47.5000         0        52.5000
```

Pentru a modifica valorile mai multor elemente dintr-un vector, funcţia `if-else` se foloseşte astfel:

```
clear
nx = 21
dxg = 2.5 % metri

for ix = 1:nx
    if ix > 15
        dist(ix) = 0.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =
Columns 1 through 7
2.5000    5.0000    7.5000   10.0000   12.5000   15.0000   17.5000
Columns 8 through 14
20.0000   22.5000   25.0000  27.5000   30.0000   32.5000   35.0000
```

```
Columns 15 through 21
37.5000         0         0         0         0         0         0
```

Pentru a modifica valorile elementelor dintr-un vector folosind două condiţionări, funcţia `if-elseif` se foloseşte astfel:

```
clear
nx = 21
dxg = 2.5 % metri

for ix = 1:nx
    if ix < 5
        dist(ix) = 0.0;
    elseif ix > 15
        dist(ix) = 0.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =
Columns 1 through 8
0         0         0         0   12.5000   15.0000   17.5000   20.0000
Columns 9 through 16
22.5000   25.0000   27.5000   30.0000   32.5000   35.0000   37.5000   0
Columns 17 through 21
0         0         0         0         0
```

Funcţia `rand` poate fi folosită în definirea vectorilor cu `if-else-elseif` astfel:

```
clear
nx = 21
dxg = 2.5 % metri

for ix = 1:nx
    if ix < 5
        dist(ix) = rand(1,1);
    elseif ix > 15
        dist(ix) = 0.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =
Columns 1 through 7
```

```
0.7749    0.8173    0.8687    0.0844   12.5000   15.0000   17.5000
Columns 8 through 14
20.0000   22.5000   25.0000   27.5000   30.0000   32.5000   35.0000
Columns 15 through 21
37.5000         0         0         0         0         0         0
```

Definirea unor valori aleatoare cu o eroare maximă `err_max` numai pentru anumite elemente dintr-un vector se poate face astfel:

```
clear
nx = 21
dxg = 2.5 % metri
err_max = 0.25

for ix = 1:nx
    if ix <= 5
        dist(ix) = (2*rand(1,1)-1)*err_max;
    elseif ix > 18
        dist(ix) = 1.0;
    else
        dist(ix) = ix*dxg;
    end
end
dist

dist =
Columns 1 through 7
-0.0501   -0.1201    0.1500   -0.0343    0.2053   15.0000   17.5000
Columns 8 through 14
20.0000   22.5000   25.0000    27.5000   30.0000   32.5000   35.0000
Columns 15 through 21
37.5000   40.0000   42.5000   45.0000    1.0000    1.0000    1.0000
```

3.2.2 Definirea matricelor

Funcţiile `if`, `if-else` şi `if-elseif` se folosesc pentru a defini matrice în care valorile pe o linie sau coloană sau valorile de pe un anumit număr de linii sau coloane diferă după o formulă definită.

Modificarea valorii unui singur element dintr-o matrice se poate face astfel:

```
clear
ncol = 4
nlinie = 5

for ilinie=1:nlinie
```

```
    for icol=1:ncol
        if icol == 2
            if ilinie == 3
                M(ilinie,icol) = 100;
            else
                M(ilinie,icol) = ilinie;
            end
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1     1     1     1
     2     2     2     2
     3   100     3     3
     4     4     4     4
     5     5     5     5
```

Modificarea valorilor de o singură linie dintr-o matrice se poate face astfel:

```
clear
ncol = 4
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if ilinie == 2
            M(ilinie,icol) = 100;
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1     1     1     1
   100   100   100   100
     3     3     3     3
     4     4     4     4
     5     5     5     5
```

Modificarea valorilor de o singură coloană dintr-o matrice se poate face astfel:

```
clear
ncol = 4
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if icol == 2
            M(ilinie,icol) = 100;
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1   100     1     1
     2   100     2     2
     3   100     3     3
     4   100     4     4
     5   100     5     5
```

Modificarea valorilor de o linie şi o coloană dintr-o matrice se poate face astfel:

```
clear
ncol = 4
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if icol == 2
            M(ilinie,icol) = 100;
        elseif ilinie == 3
            M(ilinie,icol) = 100;
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1   100     1     1
     2   100     2     2
   100   100   100   100
     4   100     4     4
     5   100     5     5
```

Exerciţii cu rezolvare afişată:

E1: Definiţi o matrice cu cinci linii şi patru coloane în care valorile de pe fiecare linie sunt egale cu numărul liniei cu condiţia ca elementele de pe coloana a doua şi liniile 3 – 5 sa aibă valoarea 100.

```
clear
ncol = 4
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if icol == 2
            if ilinie > 2
                M(ilinie,icol) = 100;
            else
                M(ilinie,icol) = ilinie;
            end
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1     1     1     1
     2     2     2     2
     3   100     3     3
     4   100     4     4
     5   100     5     5
```

E2: Definiţi o matrice cu cinci linii şi patru coloane în care elementele de pe toate liniile şi coloanele 3 şi 4 au valoarea 100 iar cele de pe coloanele 1 şi 2 au valorile egale cu numărul liniilor.

```
clear
ncol = 4
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if icol > 2
            M(ilinie,icol) = 100;
        else
```

```
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1     1   100   100
     2     2   100   100
     3     3   100   100
     4     4   100   100
     5     5   100   100
```

E3: Definiţi o matrice cu cinci linii şi patru coloane în care elementele de pe toate coloanele şi liniile 4 şi 5 au valoarea 100 iar cele de pe liniile 1 - 3 şi toate coloanele au valorile egale cu numărul liniilor.

```
clear
ncol = 4
nlinie = 5

for ilinie = 1: nlinie
    for icol = 1:ncol
        if ilinie > 3
            M(ilinie,icol) = 100;
        else
            M(ilinie,icol) = ilinie;
        end
    end
end
M

M =
     1     1     1     1
     2     2     2     2
     3     3     3     3
   100   100   100   100
   100   100   100   100
```

E4: Definiţi o matrice cu cinci linii şi patru coloane în care elementele de pe primele două coloane au valoarea 100 iar cele de pe restul coloanelor au valoarea 1000.

```
clear
ncol = 4
```

```
nlinie = 5

for ilinie = 1:nlinie
    for icol = 1:ncol
        if icol < 3
            M(ilinie,icol) = 100;
        else
            M(ilinie,icol) = 1000;
        end
    end
end
M

M =
         100         100        1000        1000
         100         100        1000        1000
         100         100        1000        1000
         100         100        1000        1000
         100         100        1000        1000
```

Exercitii fără rezolvare afişată:

E1: Definiţi un vector cu zece elemente în care primele patru elemente au valoarea 10 iar restul au valori egale cu poziţia elementului în vector.

E2: Definiţi un vector cu 21 elemente cu valori `0:5:100` iar al 17-lea element să aibă valoarea zero.

E3: Definiţi un vector cu 21 elemente, valoare minimă 0, valoare maximă 100, pas 5, iar la primele 5 elemente se adaugă/scad valori aleatoare cu o eroare maximă de 30% din 5.
E4: Definiţi un vector coloană cu 21 elemente ale căror valori au semnificaţie de viteză de propagare a undei P cu o valoare minimă de 2000 m/s şi pas egal cu 10 m/s.

E5: Definiţi un vector coloană cu 31 elemente în care doar elementele cu poziţiile 3 şi 10 au valori diferite de zero, -0.01 şi 0.01, iar restul sunt zero. Acest vector descrie o funcţie de reflectivitate.

E6: Definiţi o matrice cu cinci linii şi patru coloane cu valoarea 1 pe primele două coloane şi 10 pe restul coloanelor.

E7: Definiţi o matrice cu șase linii şi patru coloane cu valori ce variază de la -0.5 la 0.5 pe primele trei linii şi cu valoarea 1 pe restul liniilor.

E8: Definiţi o matrice cu șase linii şi șase coloane în care avem următoarele valori:

```
ncol=1:6, nlinie=1:2: valoare minima = 100, pas = 5
ncol=1:3, nlinie=3:4, valoarea 2000
ncol=4:6, nlinie=3:4, valoarea 2500
ncol=1:6, nlinie=5:6, valoarea 1500
```

3.3 Definirea vectorilor şi a matricelor folosind fişierele function

Vectorii şi matricele se pot defini folosind fisierele *function*. Modelele de viteze notate cu variabilele `vel1`, `vel2` şi `vel3` au fost obţinute folosind programul:

```
function [vel1,vel2,vel3]=veloc(nlinie,ncol)

for ilinie=1:nlinie
    for icol=1:ncol
        vel1(ilinie,icol)=400+ilinie;
        vel2(ilinie,icol)=500+icol;
        vel3(ilinie,icol)=300+ilinie*icol;
    end
end
vel1
vel2
vel3
```

executând în *Command Window* linia:

```
[vel1,vel2,vel3]=veloc(5,3);

vel1 =
   401   401   401
   402   402   402
   403   403   403
   404   404   404
   405   405   405

vel2 =
   501   502   503
   501   502   503
   501   502   503
   501   502   503
   501   502   503

vel3 =
   301   302   303
   302   304   306
   303   306   309
```

```
   304   308   312
   305   310   315
```

Avantajul folosirii fişierului *function* este că salvează numai variabilele definite de către programator ca date de ieşire între parantezele drepte:

```
>> whos
Name      Size            Bytes  Class     Attributes
vel1      5x3               120  double
vel2      5x3               120  double
vel3      5x3               120  double
```

Capitolul 4
Reprezentări grafice 1D, 2D şi 3D

4.1 Reprezentarea vectorilor. Editarea graficelor

Orice vector se poate reprezenta grafic folosind funcţia `plot`. Pentru a obţine informaţii despre folosirea acestei funcții se foloseşte expresia `help plot` executată direct în *Command Window*:

```
>> help plot
 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, disconnected
    line objects are created and plotted as discrete points vertically
at
    X.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

           b     blue          .     point              -     solid
           g     green         o     circle             :     dotted
           r     red           x     x-mark             -.    dashdot
           c     cyan          +     plus               --    dashed
           m     magenta       *     star             (none)  no line
           y     yellow        s     square
           k     black         d     diamond
           w     white         v     triangle (down)
                               ^     triangle (up)
                               <     triangle (left)
                               >     triangle (right)
                               p     pentagram
                               h     hexagram

    For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
    at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
    point but does not draw any line.
```

În cazul în care se doreşte reprezentarea mai multor vectori pe acelaşi grafic, de exemplu pentru comparaţie, se foloseşte funcţia `hold on` (Figura 4.1):

```
% definire şi reprezentare vectori distanţe:
dx1 = 5
dx2 = 10
dx3 = 15
nx  = 80

for ix=1:nx
    dist1(ix) = ix*dx1;
    dist2(ix) = ix*dx2;
    dist3(ix) = ix*dx3;
end
plot(dist1)
hold on
plot(dist2,'r')
plot(dist3,'g')
xlabel('numar de geofoane')
ylabel('distanta (m)')
```

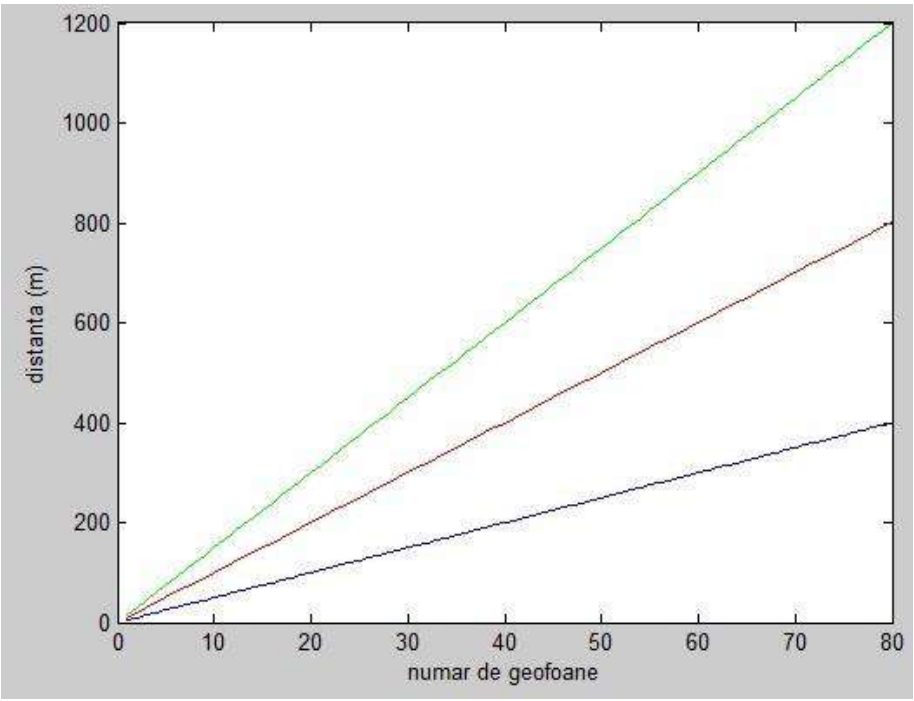

Figura 4.1: Reprezentarea vectorilor `dist1`, `dist2` şi `dist3`

În cazul în care se doreşte reprezentarea vectorilor în două figuri separate întrerupem continuitatea reprezentărilor prin folosirea funcţiilor `hold off` şi `hold on`:

```
% definire şi reprezentare vectori distanţe:
dx1 = 5
```

```
dx2 = 10
dx3 = 15
nx  = 80

for ix=1:nx
    dist1(ix) = ix*dx1;
    dist2(ix) = ix*dx2;
    dist3(ix) = ix*dx3;
end
plot(dist1)
hold on
plot(dist2,'r')
xlabel('numar de geofoane')
ylabel('distanta (m)')
hold off
figure(2)
plot(dist3,'g')
xlabel('numar de geofoane')
ylabel('distanta (m)')
```

Rezultatele executării programului de mai sus sunt prezentate în Figurile 4.2a şi 4.2b.

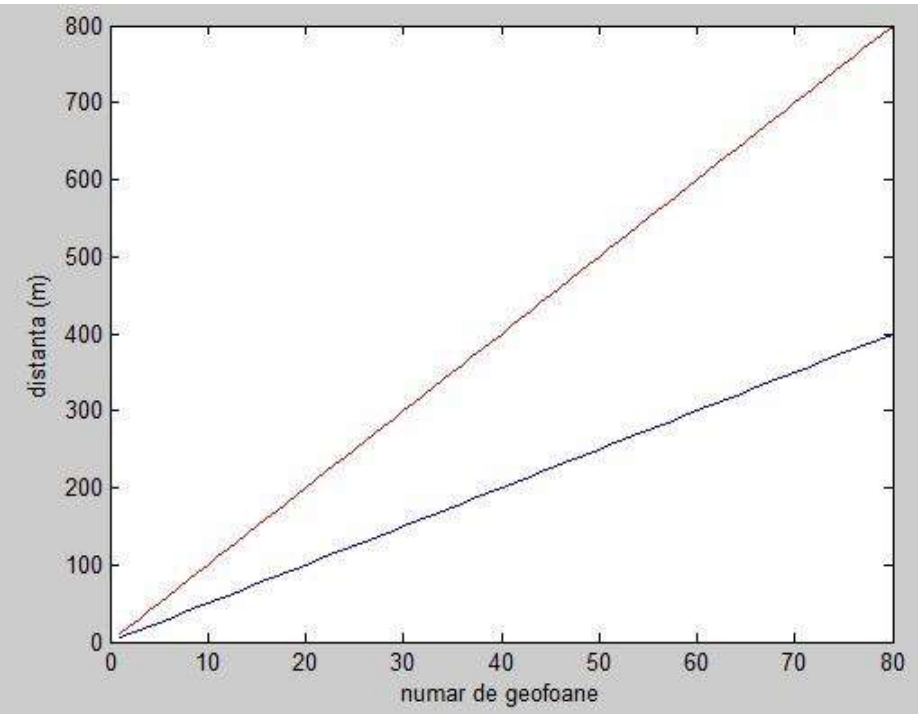

Figura 4.2a: Reprezentarea vectorilor `dist1` şi `dist2`

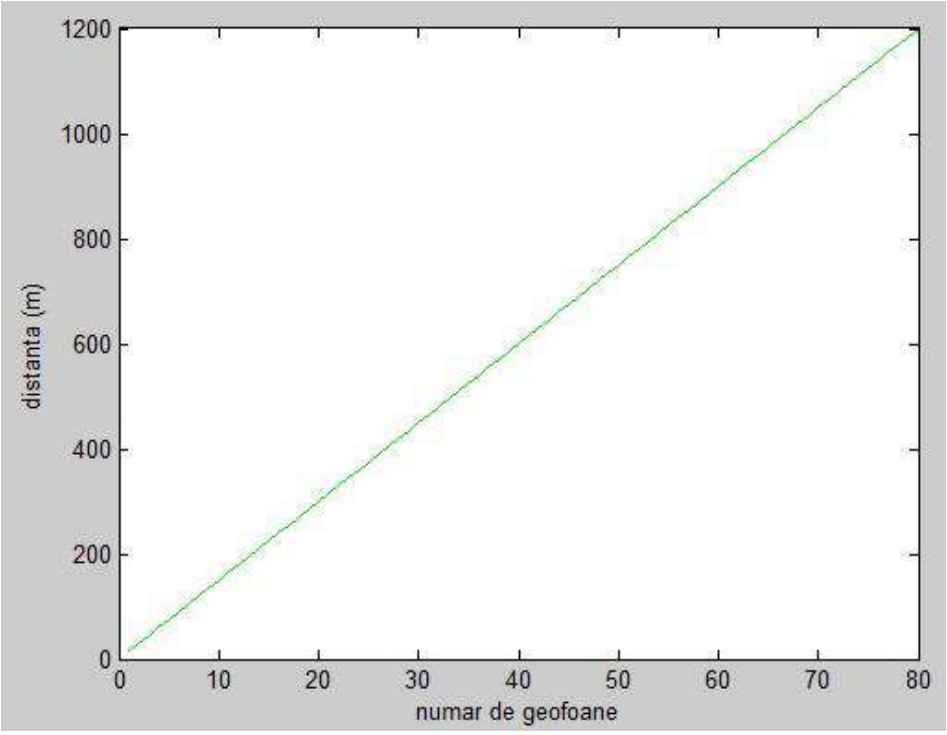

Figura 4.2b: Reprezentarea vectorului `dist3`

O figură nouă se deschide folosind funcţia `figure`. Editarea axelor de coordonate şi a titlului se poate face folosind funcţiile `xlabel`, `ylabel`, `axis` şi `title`.

4.2 Reprezentarea matricelor. Editarea graficelor

O seismogramă ce conţine cel puţin două trase este încărcată în MATLAB sub forma unei matrice in care numărul de eşantioane conţinute de fiecare trasă reprezintă numărul de linii, iar numărul de trase reprezintă numărul de coloane din matrice. Pentru reprezentarea acestei matrice se foloseste funcţia `imagesc` din MATLAB sau funcţiile `plotseis` sau `plotimage` din CREWES (Figurile 4.3 – 4.5). Matricea prezentată în Figurile 4.3 – 4.5 a fost obţinută folosind programul de mai jos:

```
clear
nlinie = 6
ncol = 6
for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie < 3
            M(ilinie,icol)=100+(icol-1)*5;
        elseif ilinie > 4
            M(ilinie,icol)=1500;
```

```
        else
            if icol<4
                M(ilinie,icol)=2000;
            else
                M(ilinie,icol)=2500;
            end % if icol
        end % if ilinie
    end % for icol
end % for ilinie
M
imagesc(M)
```

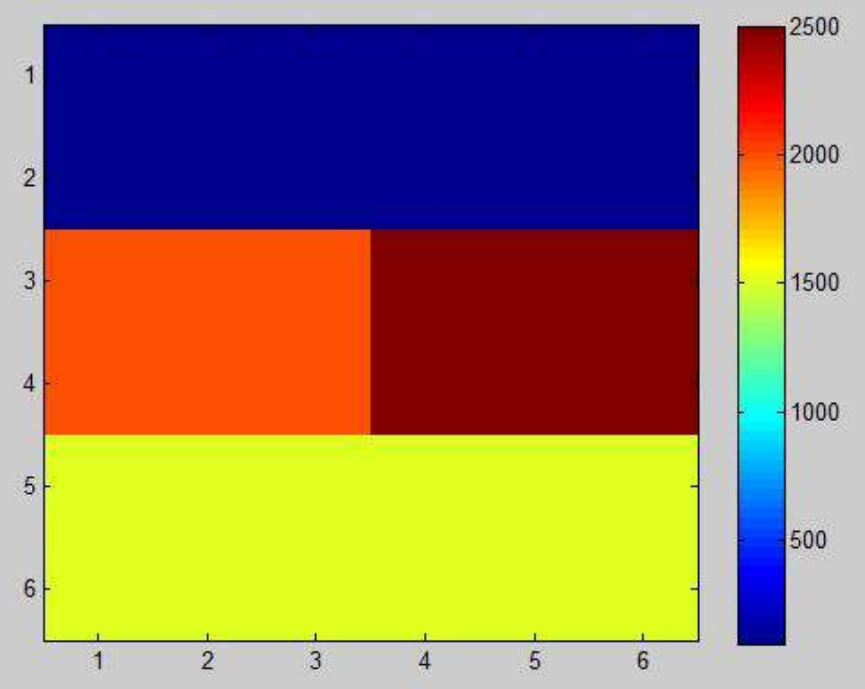

Figura 4.3: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

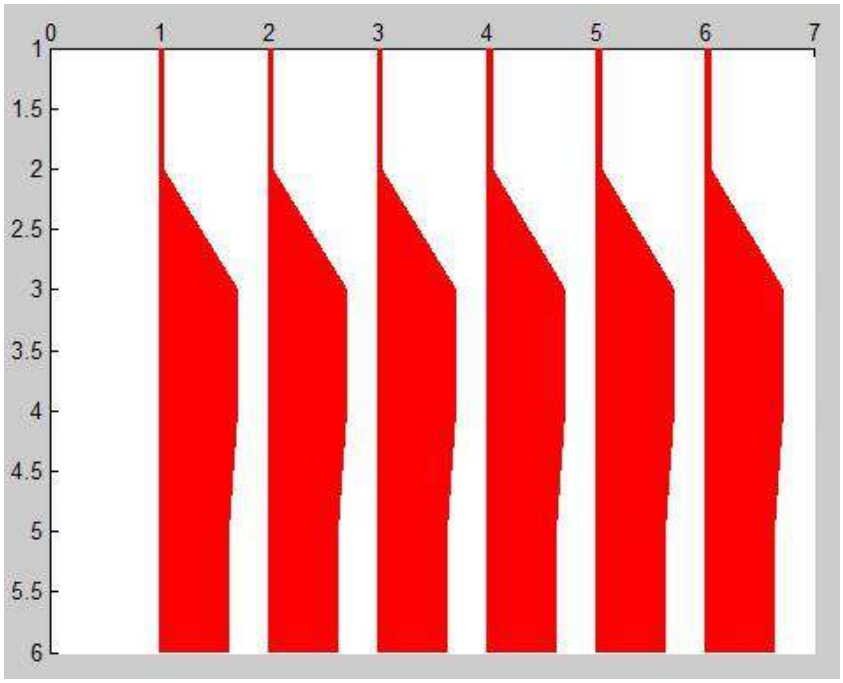

Figura 4.4: Reprezentarea valorilor matricei `M` folosind funcţia `plotseis`

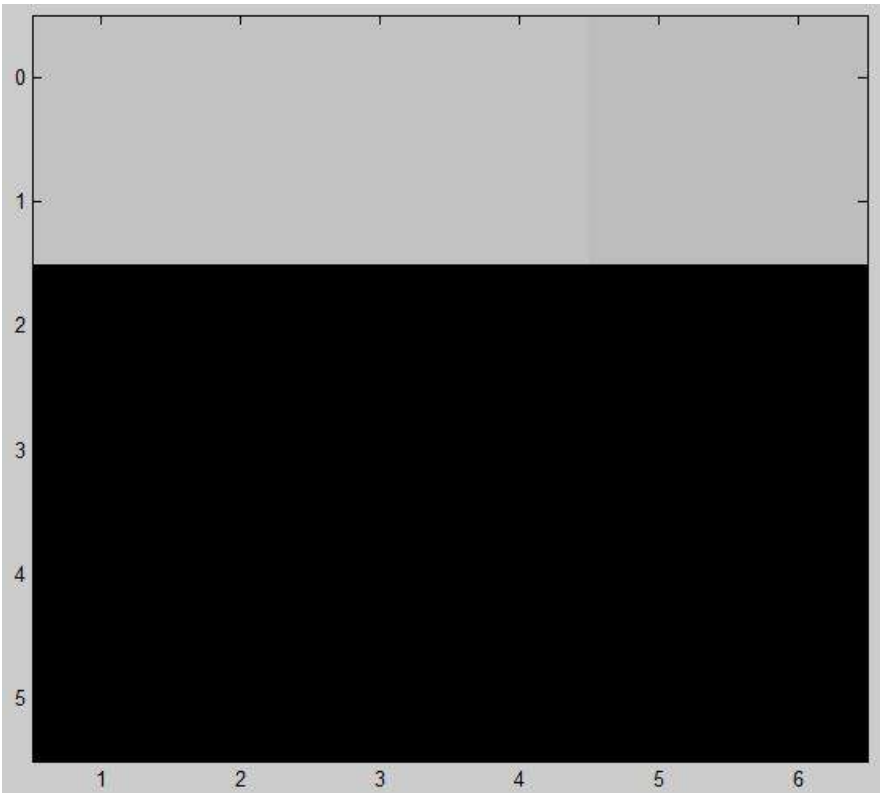

Figura 4.5: Reprezentarea valorilor matricei `M` folosind funcţia `plotimage`

Editarea axelor se face folosind funcţiile `xlabel` şi `ylabel` iar editarea valorilor de pe axe se face definind vectorii corespunzători acestor valori (`dist`, `adancime`). Rezultatele sunt prezentate în Figurile 4.6, 4.7 şi 4.8.

```
clear
nlinie = 6
ncol = 6
dxg = 5
dz = 10
dist = dxg*(1:ncol);
adancime = dz*(1:nlinie);

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie < 3
            M(ilinie,icol)=100+(icol-1)*5;
        elseif ilinie > 4
            M(ilinie,icol)=1500;
        else
            if icol<4
                M(ilinie,icol)=2000;
            else
                M(ilinie,icol)=2500;
            end % if icol
        end % if ilinie
    end % for icol
end % for ilinie
M

imagesc(dist,adancime,M)
xlabel('distanta (m)')
ylabel('adancime (m)')

plotseis(M,adancime,dist)
xlabel('distanta (m)')
ylabel('adancime (m)')

plotimage(dist,adancime,M)
xlabel('distanta (m)')
ylabel('adancime (m)')
```

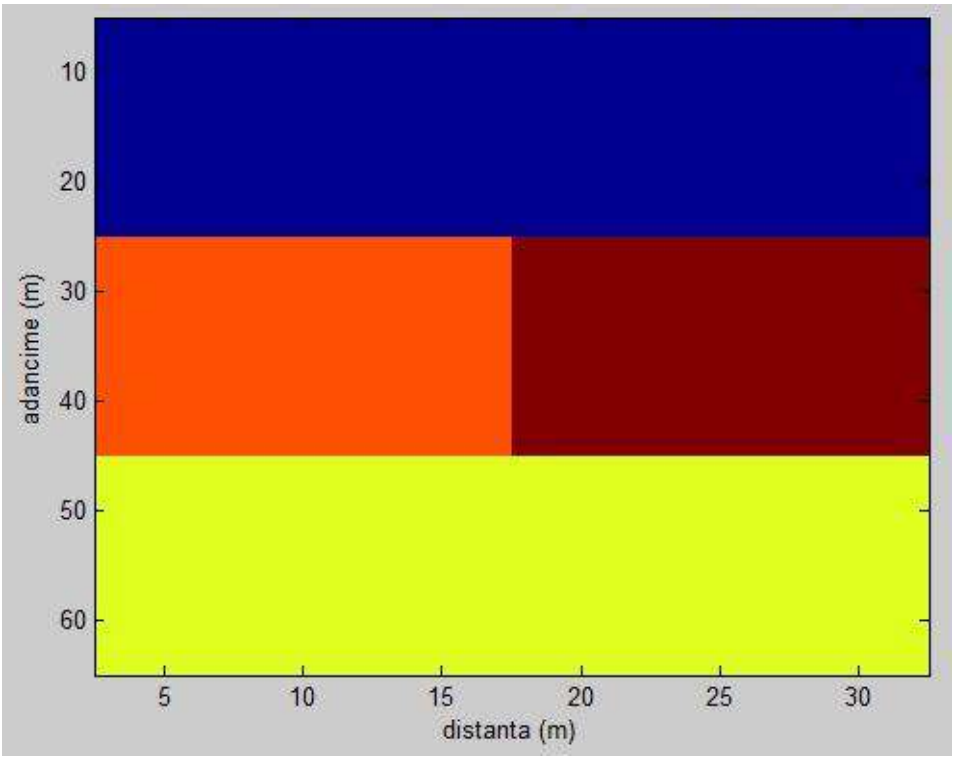

Figura 4.6: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc` cu editarea axelor

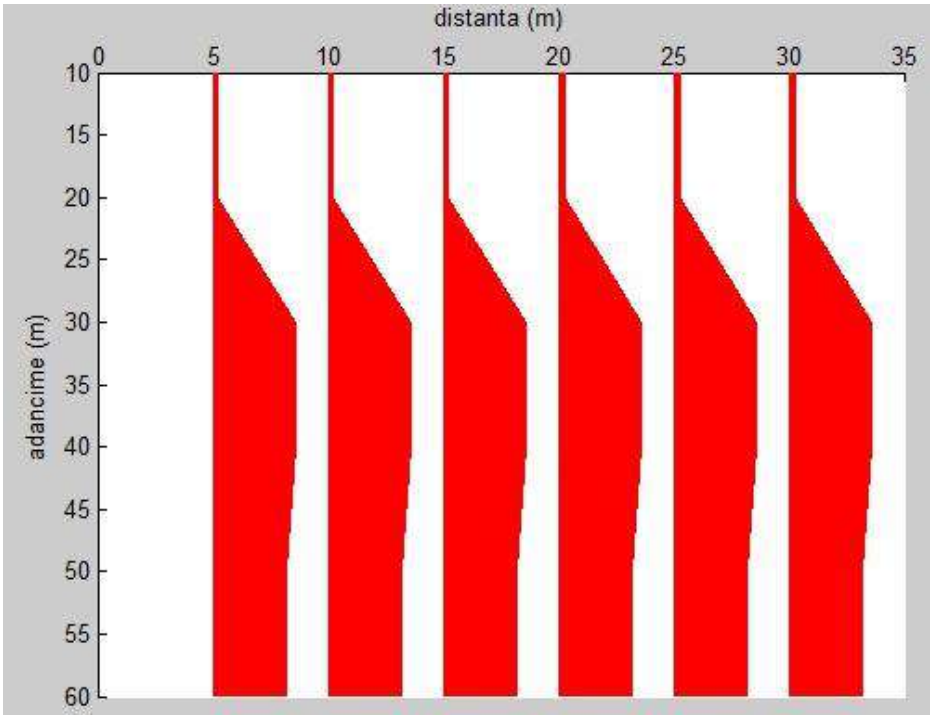

Figura 4.7: Reprezentarea valorilor matricei `M` folosind funcţia `plotseis` cu editarea axelor

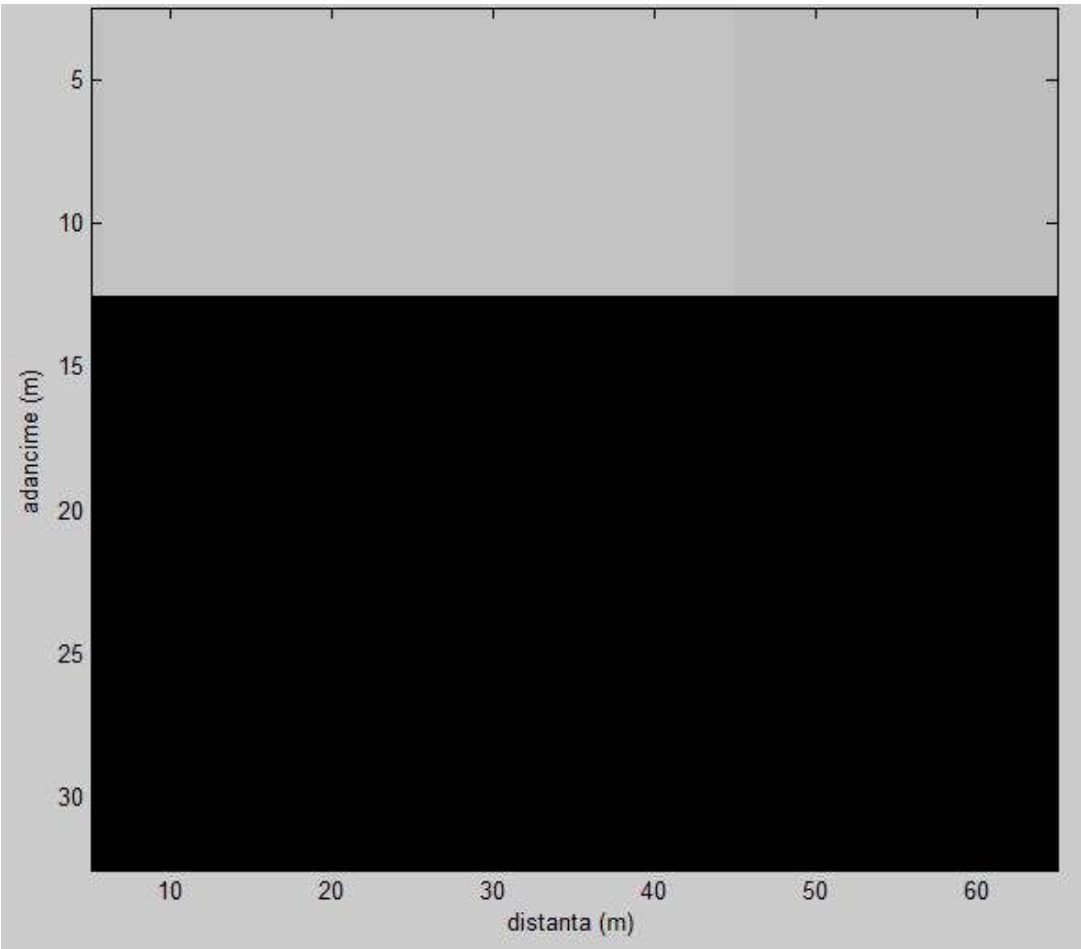

Figura 4.8: Reprezentarea valorilor matricei `M` folosind funcţia `plotimage` cu editarea axelor

4.3 Reprezentarea 3D. Editarea graficelor

Orice matrice poate fi reprezentată 3D color folosind funcţia `surf`. În Figura 4.9 este prezentat modelul de viteză calculat folosind programul de mai jos.

```
clear
nlinie = 6
ncol = 6
dxg = 5
dz = 10
dist = dxg*(1:ncol);
adancime = dz*(1:nlinie);

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie < 3
```

```
            M(ilinie,icol)=100+(icol-1)*5;
        elseif ilinie > 4
            M(ilinie,icol)=1500;
        else
            if icol<4
                M(ilinie,icol)=2000;
            else
                M(ilinie,icol)=2500;
            end % if icol
        end % if ilinie
    end % for icol
end % for ilinie
M
surf(M)
xlabel('numar linii')
ylabel('numar coloane')
zlabel('Viteza de propagare (m/s)')
```

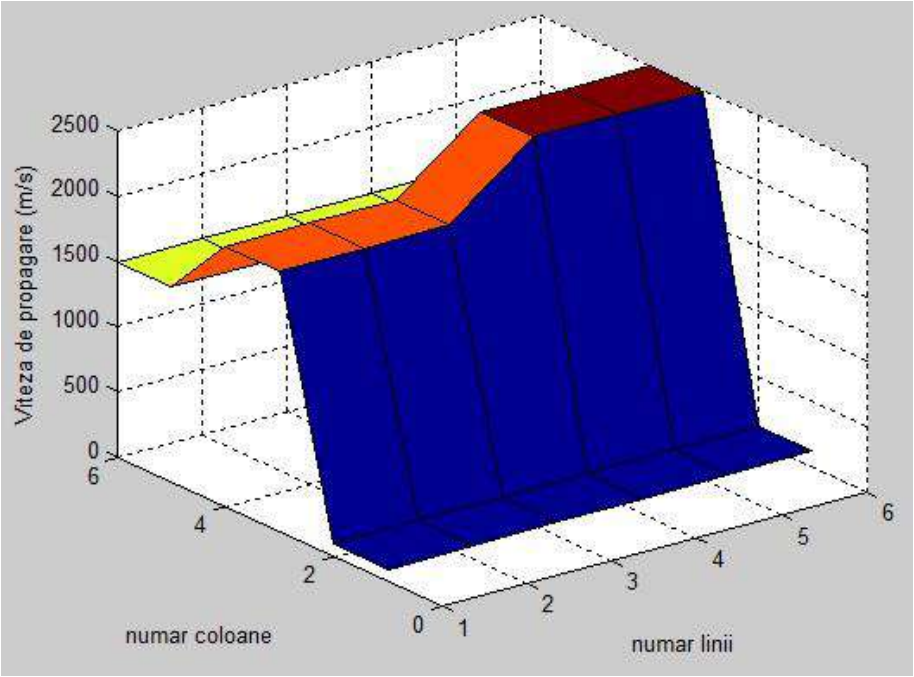

Figura 4.9: Reprezentarea valorilor matricei `M` folosind funcţia `surf`

Orice matrice cu trei coloane poate fi reprezentată 3D folosind această funcţie. Coordonatele surselor seismice se pot încărca în MATLAB şi reprezenta folosind funcţia `surf`:

```
S = load('D:\Matlab\REC_Matlab.prn');
S_10 = S(1:10,2:4);
```

```
surf(S_10)
```

Valorile conţinute de matricea `S_10` pot fi prezentate numeric sau grafic (Figura 4.10).

```
>> S_10=S(1:10,2:4)

ans =

      459362      683571         693
      459352      683561         696
      459341      683550         698
      459330      683539         700
      459320      683529         700
      459310      683518         701
      459299      683507         700
      459289      683497         701
      459280      683483         704
      459263      683479         707
```

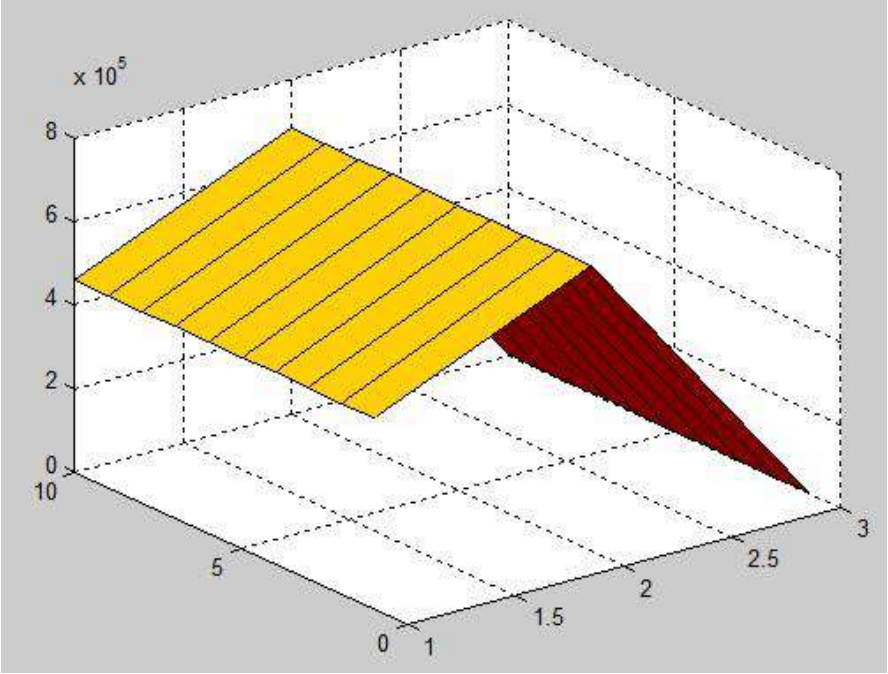

Figura 4.10: Reprezentarea valorilor matricei `S_10`

Exerciţii cu rezolvare afişată:

E1: Definiţi o matrice cu 10 coloane şi 10 linii ce reprezintă un model de viteză 2D în care viteza variază numai pe direcţie orizontală, valoare minimă de 2000 m/s şi pas 50 m/s şi este constantă în timp (direcţie verticală). Modelul este prezentat în Figura 4.11.

```
clear
nlinie = 10
ncol = 10
pas = 50

for ilinie=1:nlinie
    for icol=1:ncol
        M(ilinie,icol)=2000+(icol-1)*pas;
    end % for icol
end % for ilinie
M
imagesc(M)
```

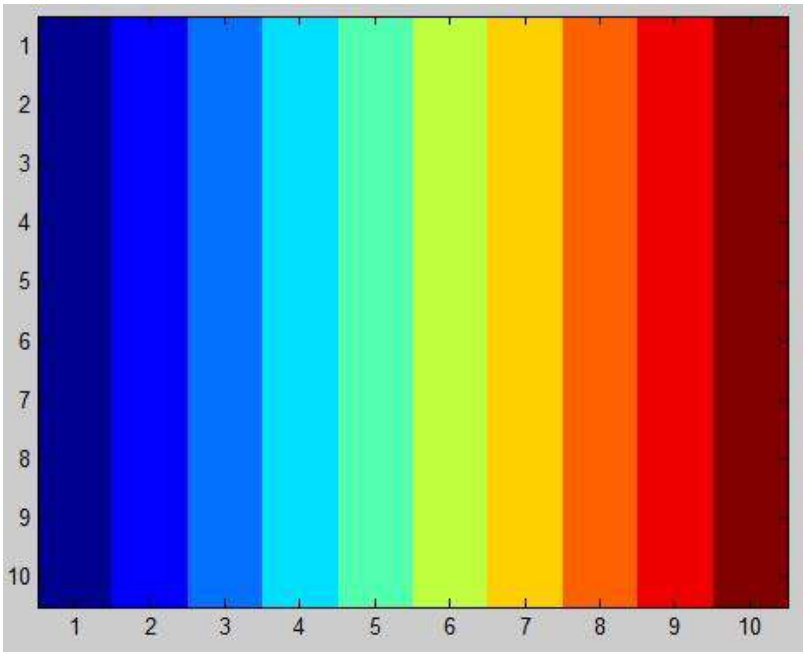

Figura 4.11: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E2: Definiţi o matrice cu 10 coloane şi 10 linii ce reprezintă un model de viteză 2D în care viteza variază numai pe direcţie verticală, valoare minimă de 1000 m/s, pas 25 m/s şi este constantă în lungul întinderii de geofoane (pe direcţie orizontală). Modelul este prezentat în Figura 4.12.

```
clear
nlinie = 10
ncol = 10
pas = 25

for ilinie=1:nlinie
    for icol=1:ncol
        M(ilinie,icol)=1000+(ilinie-1)*pas;
    end % for icol
end % for ilinie
M
imagesc(M)
xlabel('Numar receptoare')
ylabel('Timp (s)')
```

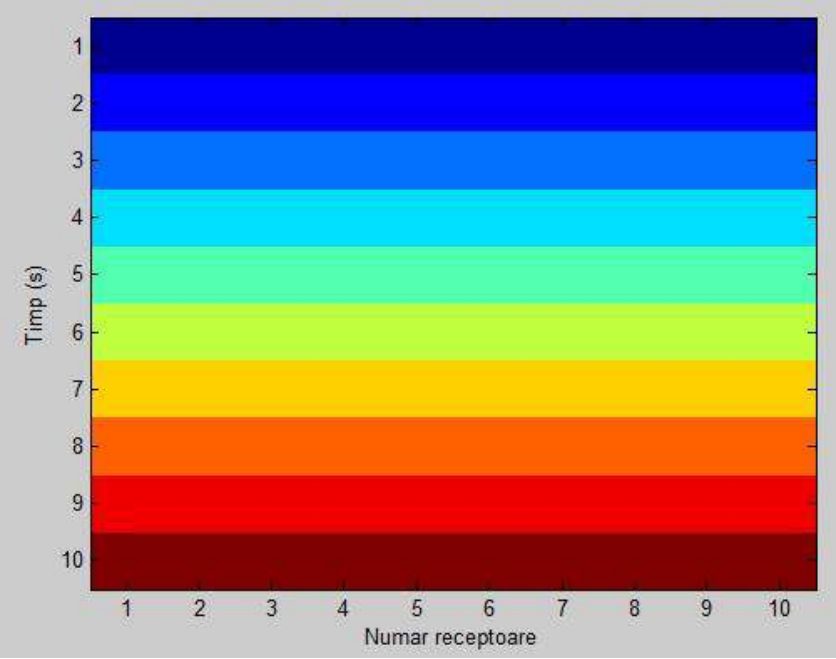

Figura 4.12: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E3: Definiţi o matrice cu 10 coloane şi 10 linii ce reprezintă un model de viteză 2D în care viteza variază în lungul întinderii de receptoare şi în timp. Valoarea minimă de viteză este 1000 m/s, pasul de variaţie în timp este 100 m/s iar cel de variaţie în lungul întinderii de geofoane este 25 m/s. Modelul este prezentat în Figura 4.13.

```
clear
nlinie = 10
```

```
ncol = 10
pas_linie = 100
pas_col = 25

for ilinie=1:nlinie
    for icol=1:ncol
        M(ilinie,icol)=1000+(ilinie-1)*pas_linie+(icol-1)*pas_col;
    end % for icol
end % for ilinie
M
imagesc(M)
xlabel('Numar receptoare')
ylabel('Timp (s)')
```

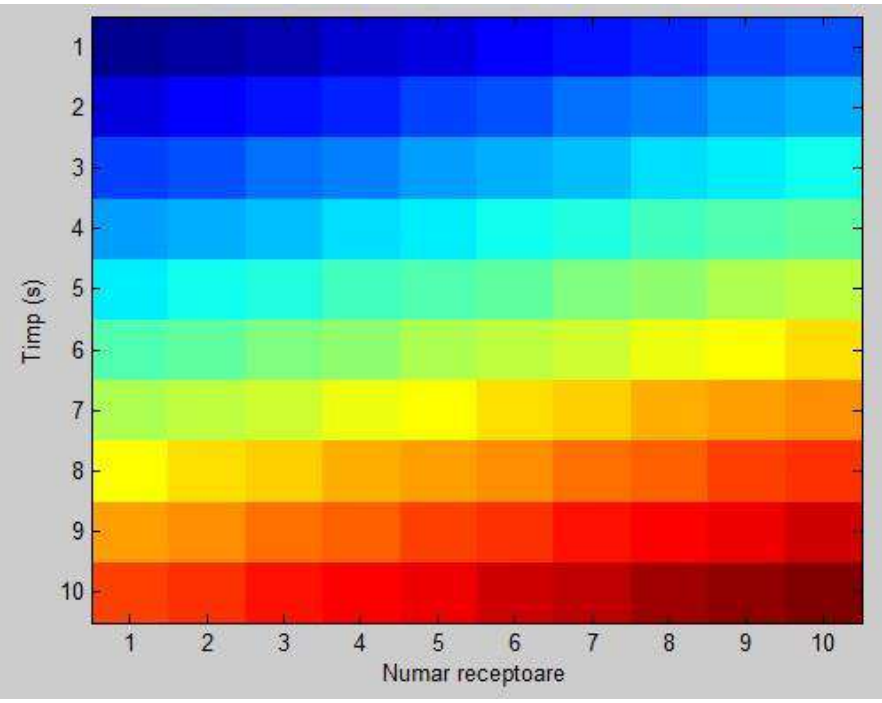

Figura 4.13: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E4: Definiţi o matrice cu 100 linii, 70 coloane, viteza minimă de 1000 m/s. Pe primele 49 coloane şi toate liniile, pasul de variaţie pe orizontală al vitezei este de 100 m/s. Pe coloanele 50-70 şi liniile 1-100, viteza este constantă si egală cu 500 m/s. Modelul este prezentat în Figura 4.14.

```
clear
nlinie = 100
ncol = 70
pas_col = 100
```

```
for ilinie=1:nlinie
    for icol=1:ncol
        if icol<50
            M(ilinie,icol)=1000+(icol-1)*pas_col;
        else
            M(ilinie,icol) = 500;
        end
    end % for icol
end % for ilinie
M
imagesc(M)
xlabel('Numar receptoare')
ylabel('Timp (s)')
```

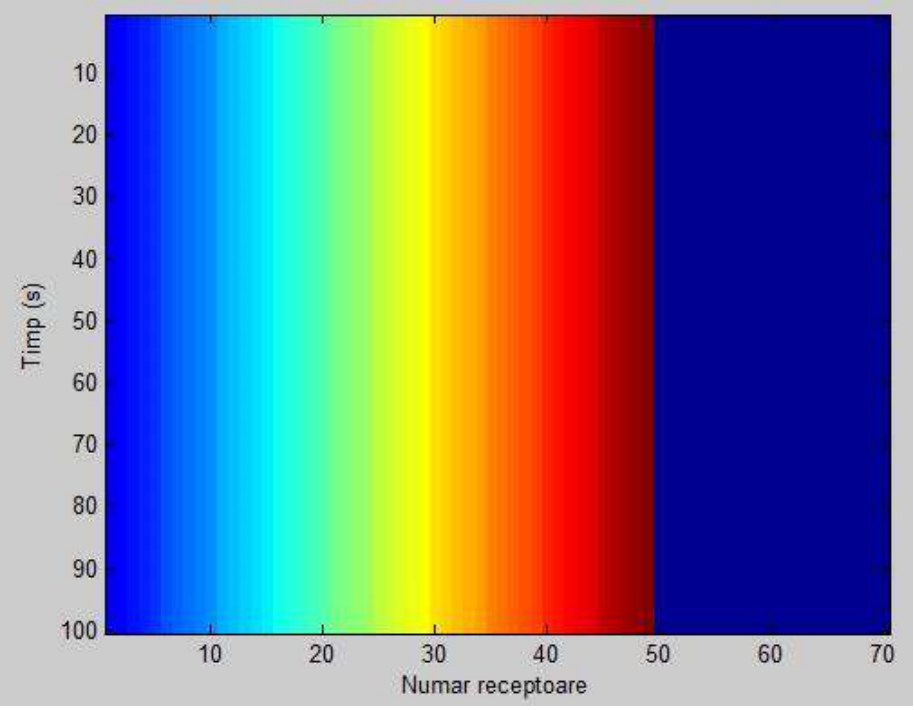

Figura 4.14: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E5: Definiţi o matrice care să reprezinte un model de viteză bistratificat dar cu variaţie continuă de viteză în funcţie de timp în fiecare strat.

Date descriere model:

`ncol = 100`, `nlinie = 80`, `V1_min = 1000`, `pas1 = 25`, `V2_min = 2000`, `pas2 = 50`.

Limita de strat este la linia 45. Modelul este prezentat în Figura 4.15.

```
clear
nlinie = 80
ncol = 100
limita = 45
V1_min = 1000
pas1 = 25
V2_min = 2000
pas2 = 50

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie < limita
            M(ilinie,icol)=1000+(ilinie-1)*pas1;
        else
            M(ilinie,icol) = 2000+(ilinie-1)*pas2;
        end
    end % for icol
end % for ilinie
M
imagesc(M)
xlabel('Numar receptoare')
ylabel('Timp (s)')
```

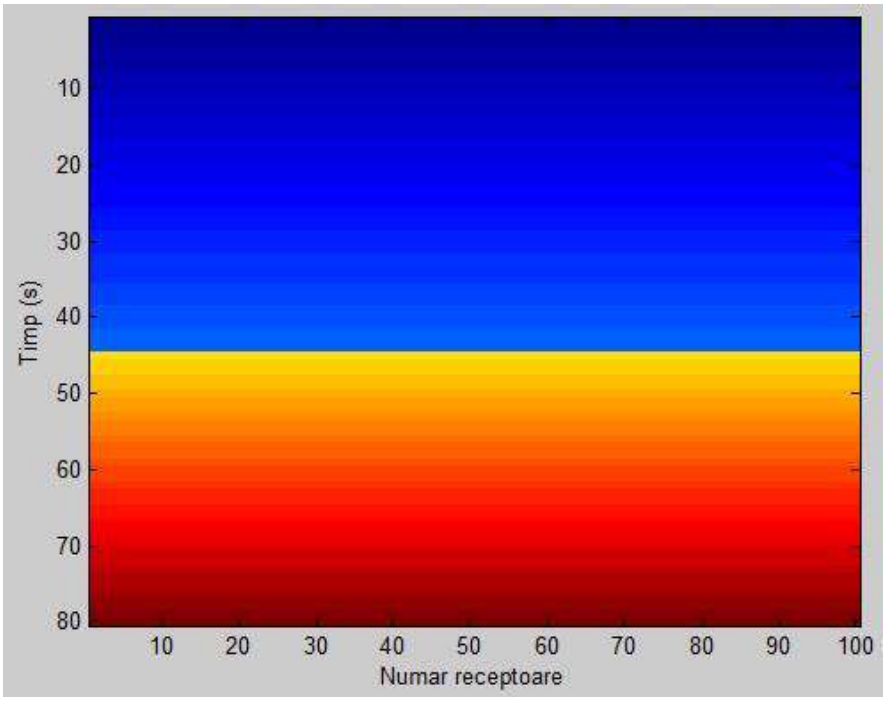

Figura 4.15: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E6: Definiţi o matrice care să reprezinte un model de viteză cu trei strate orizontale în care viteza variază în primul strat în lungul întinderii de geofoane, este constantă în stratul al doilea şi variază cu timpul în stratul al treilea.
Date descriere model:
`ncol = 100, nlinie = 90, V1_min = 1000, pas1 = 25, V2 = 2000, V3_min = 4000, pas3 = 40.`
Prima limită de strat, L1, este la linia 35 iar a doua limită de strat, L2, este la linia 65.
Modelul este prezentat în Figura 4.16.

```
clear
nlinie = 90
ncol = 100
L1 = 35
L2 = 65
V1_min = 1000
pas1 = 25
V2 = 2000
V3_min = 4000
pas3 = 40

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie < L1
            M(ilinie,icol)=V1_min+(icol-1)*pas1;
        elseif ilinie > L2
            M(ilinie,icol) = V3_min+(ilinie-1)*pas3;
        else
            M(ilinie,icol) = V2;
        end
    end  % for icol
end % for ilinie
M
imagesc(M)
xlabel('Numar receptoare')
ylabel('Timp (s)')
```

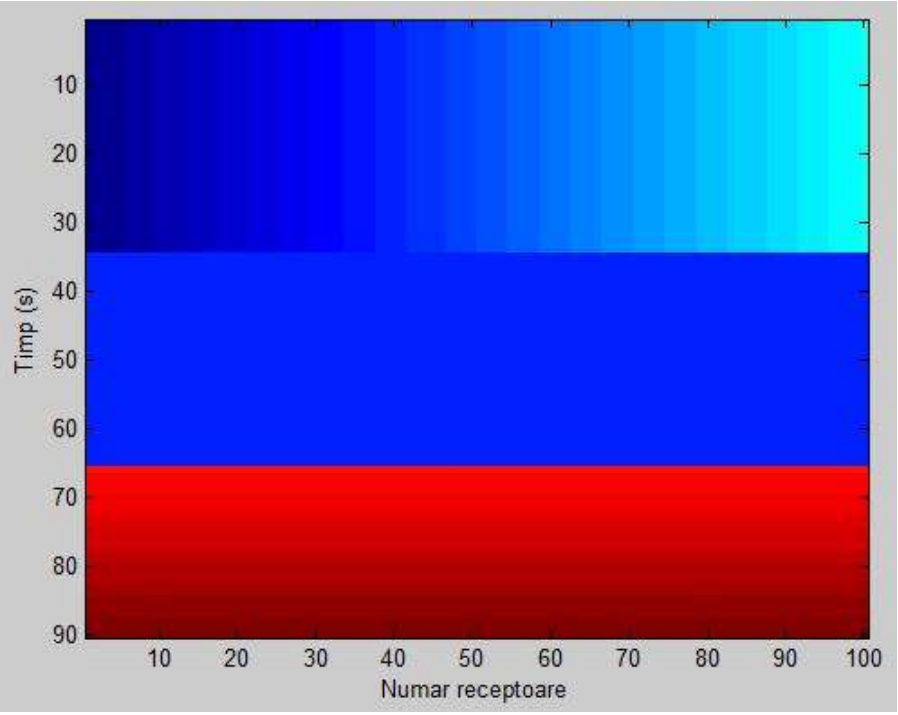

Figura 4.16: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E7: Definiți o matrice care să reprezinte un model de viteză cu două strate orizontale afectate de o falie verticală. Mărimea modelului: distanță de 100 m, pas 1 m, adâncime de 200 m, pas 1 m, falia verticală la distanța de 45 m, limita de strat la 100 m și săritura faliei de 50 m. Viteza V1 = 1000 m/s și V2 = 2000 m/s. Modelul este prezentat în Figura 4.17.

```
clear
nlinie = 200
ncol = 100
F = 45 % poziția faliei verticale
V1 = 1000
V2 = 2000
L1 = 150
L2 = 100

for ilinie=1:nlinie
    for icol=1:ncol
        if icol < F
            if ilinie < L1
                M(ilinie,icol)=V1;
            else
                M(ilinie,icol)=V2;
```

```
            end
        else
            if ilinie < L2
                M(ilinie,icol)=V1;
            else
                M(ilinie,icol)=V2;
            end
        end
    end
end
M
imagesc(M)
xlabel('Distanta (m)')
ylabel('Adancime (m)')
```

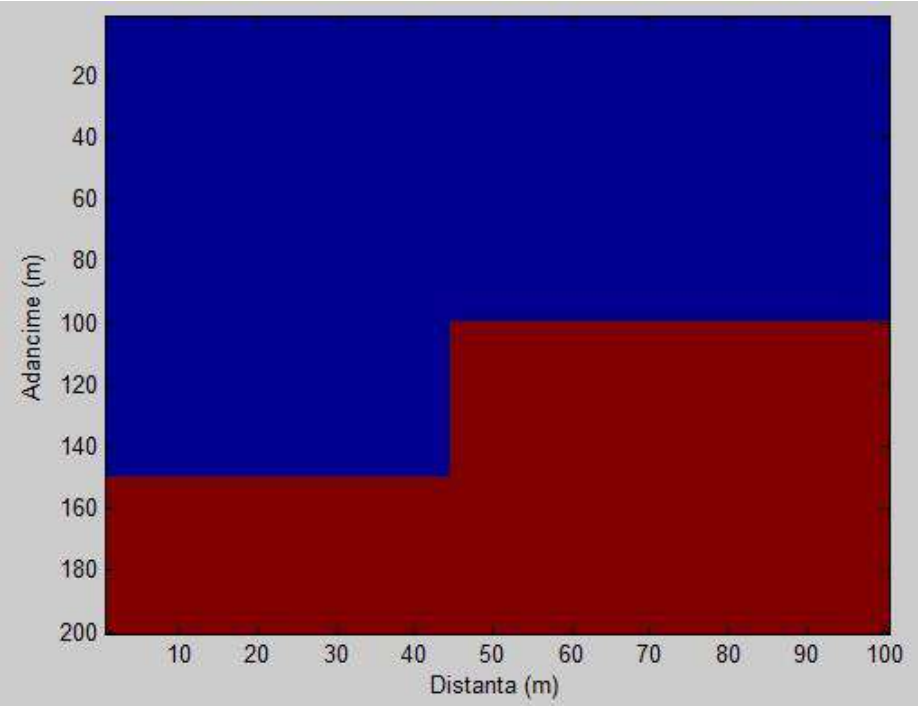

Figura 4.17: Reprezentarea valorilor matricei `M` folosind funcţia `imagesc`

E8: Definiţi un model de viteză 2D cu patru strate în care viteza variază neliniar pe verticală folosind datele scrise mai jos:

```
clear
nlinie=50
ncol=10
L1=10
L2=25
```

```
L3=40
V1=1500
V2=2000
V3=3000
V4=4000
dvel=10

for iL=1:nlinie
    L_exp(iL)=dvel*exp(0.5+0.1*iL);
end
L_exp
plot(L_exp)

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie<=L1
            vel(ilinie,icol)=V1+L_exp(ilinie);
        else
            if ilinie<=L2
                vel(ilinie,icol)=V2+L_exp(ilinie);
            elseif ilinie>=L3
                vel(ilinie,icol)=V4;
            else
                vel(ilinie,icol)=V3+L_exp(ilinie);
            end
        end
    end
end
figure(2)
imagesc(vel)
figure(3)
plot(vel(:,5))
```

Modelele de viteze 1D şi 2D sunt prezentate în Figurile 4.18 şi 4.19.

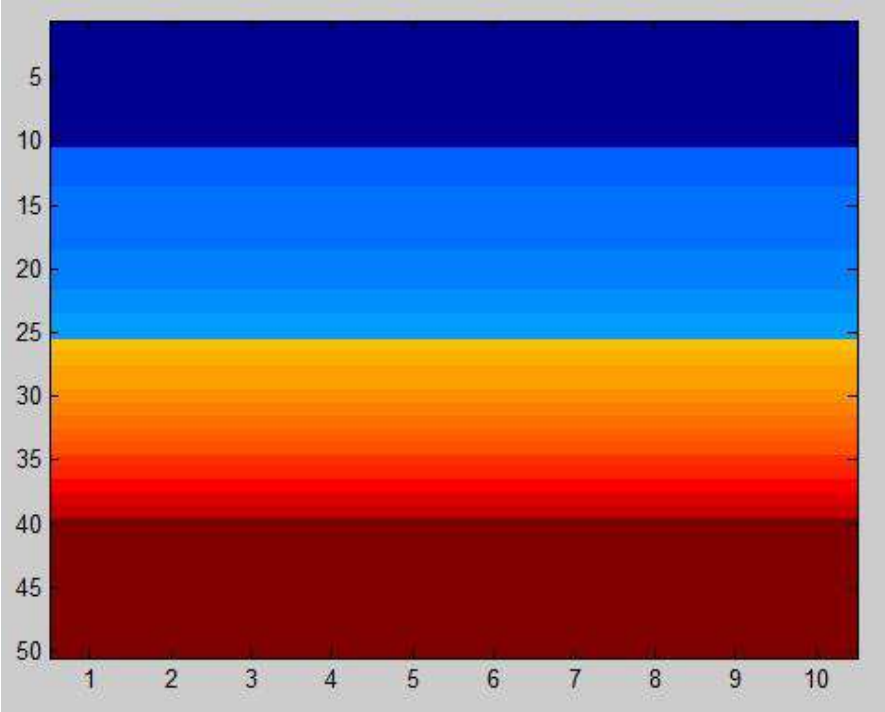

Figura 4.18: Model de viteză 2D cu patru strate cu viteză constantă pe orizontală şi variabilă neliniar pe verticală

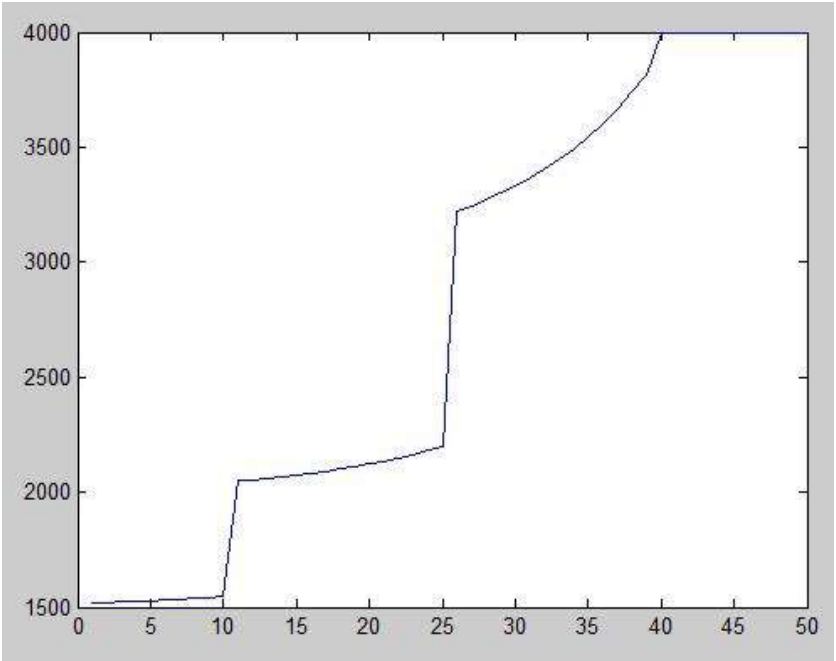

Figura 4.19: Model de viteză 1D selectat din modelul prezentat în Figura 4.18

E9: Un model cu două strate cu viteze constante separate de o limită înclinată se poate defini astfel:

```
clear
nlinie = 50
ncol = 100
V1 = 1600
V2 = 2400
L_stg = 20 % număr de linie în stânga modelului
L_dr = 25
dip = (L_dr-L_stg)/ncol

for icol=1:ncol
    L_dip(icol)=L_stg+round((icol-1)*dip);
end
L_dip

for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie >=L_dip(icol)
            vel(ilinie,icol)=V2;
        else
            vel(ilinie,icol)=V1;
        end
    end
end
vel
imagesc(vel)
xlabel('Distanta (m)')
ylabel('Adancime (m)')
```

Modelul este prezentat în Figura 4.20.

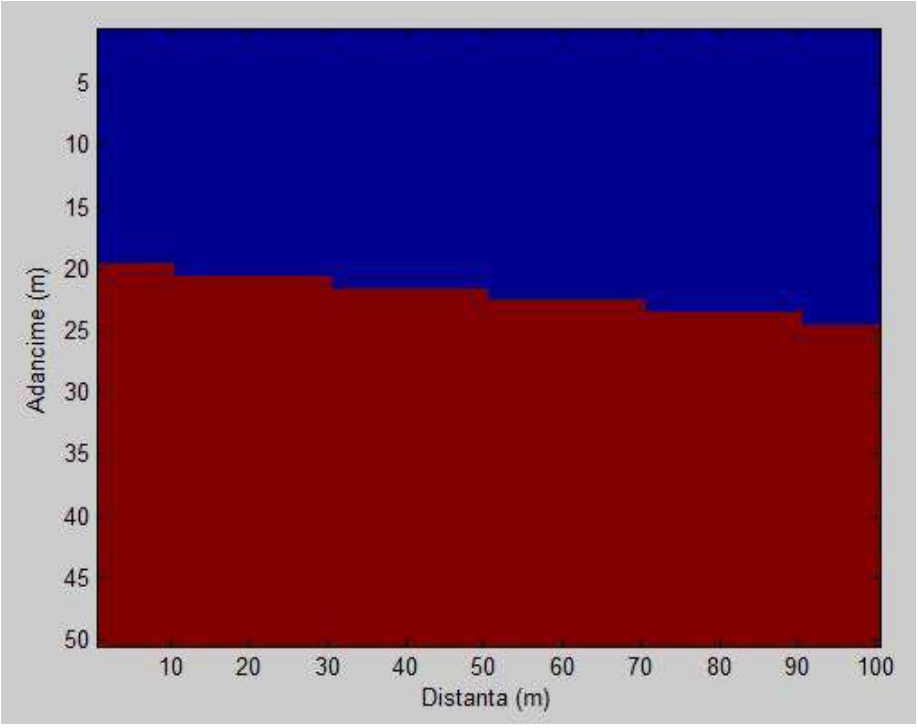

Figura 4.20: Model cu două strate cu viteze constante separate de o limită înclinată

E10: Un model cu trei strate cu viteze constante separate de două limite cu înclinări diferite se poate defini astfel:

```
clear
nlinie = 100
ncol = 100
V1 = 1600
V2 = 2400
V3 = 3200
L_stg1 = 20 % număr de linie în stânga modelului
L_dr1 = 25
dip1 = (L_dr1-L_stg1)/ncol
L_stg2 = 50 % număr de linie în stânga modelului
L_dr2 = 75
dip2 = (L_dr2-L_stg2)/ncol

for icol=1:ncol
    L_dip1(icol)=L_stg1+round((icol-1)*dip1);
    L_dip2(icol)=L_stg2+round((icol-1)*dip2);
end
L_dip1
L_dip2
```

```
for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie <=L_dip1(icol)
            vel(ilinie,icol)=V1;
        elseif ilinie >=L_dip2(icol)
            vel(ilinie,icol)=V3;
        else
            vel(ilinie,icol)=V2;
        end
    end
end
vel
imagesc(vel)
xlabel('Distanta (m)')
ylabel('Adancime (m)')
```

Modelul este prezentat în Figura 4.21.

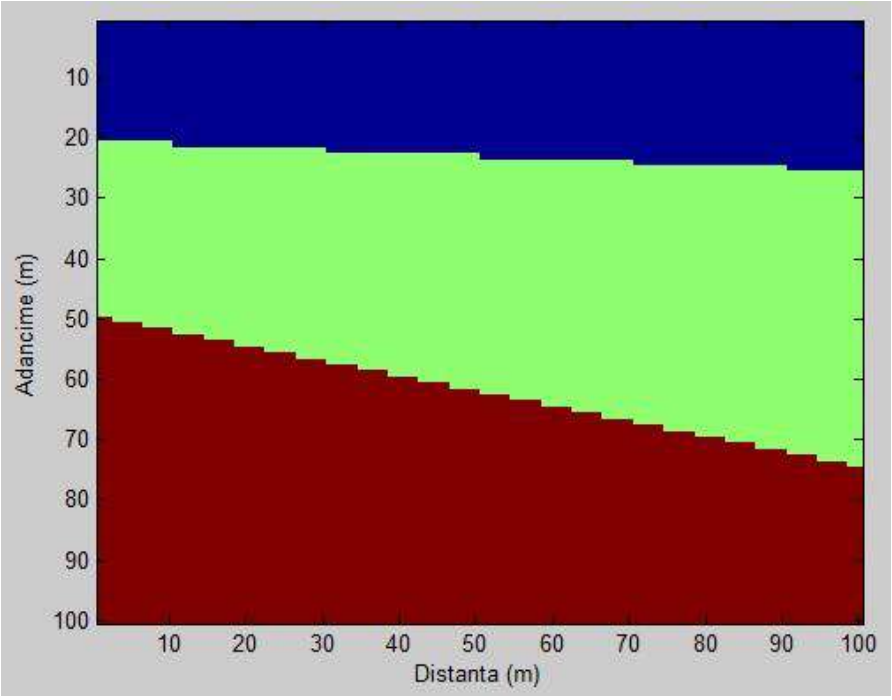

Figura 4.21: Model cu trei strate cu viteze constante separate de limite cu înclinări diferite

E11: Un model cu trei strate orizontale cu viteze constante pe orizontală şi variabile pe verticală afectat de o falie normală poate fi obţinut folosind programul prezentat mai jos. Se defineşte mai întâi modelul cu stratificaţie orizontală apoi se construieşte falia normală. Modelele de viteze sunt prezentate în Figurile 4.22 – 4.23,

```
clear Vp_1 Vp_2 Vp_3 Vp_4 Vp_5
z1=5; dz=0.5; nz1=z1/dz
Vp_1=zeros(nz1,1);
for iz1=1:nz1
    Vp_1(iz1)=10 + (iz1-1)*3;
end

z2=15; h2=z2-z1; dz=0.5; nz2=h2/dz
Vp_2=zeros(nz2,1);
for iz2=1:nz2
    Vp_2(iz2)=110 + 2.5*(iz2-1);
end

z3=55; h3=z3-z2;dz=0.5; nz3=h3/dz
Vp_3=zeros(nz3,1);
for iz3=1:nz3
    Vp_3(iz3)=170 + 16*(iz3-1);
end

z4=105; h4=z4-z3;dz=0.5; nz4=h4/dz
Vp_4=zeros(nz4,1);
for iz4=1:nz4
    Vp_4(iz4)=2410 + iz4;
end

z5=205; h5=z5-z4; nz5=h5/dz
Vp_5 = 3210*ones(201,1);
Vp_tot=[Vp_1;Vp_2;Vp_3;Vp_4;Vp_5];

nx=801 % G.dx = 0.5 m
nz=411 % G.dz = 0.5 m, z = 105 m
Vp_801=zeros(411,nx);
for ix=1:nx
    Vp_801(:,ix)=Vp_tot;
end

% săritură după falie normală
nz = 411;
Vp_sh = zeros(nz,nx);
zref = 15;
x1 = 220;
x2 = 250;
z0 = 200;
for ix = 1:nx
    for ix = 1:x1
        disp('for '), ix
        for iz = 1: nz-zref
            Vp_sh(iz+zref,ix) = Vp_801(iz,ix);
```

```
        end
    end
    for ix = x1+1 : x2
        z1 = -z0*((ix - x2)/(x2 - x1));
        for iz = 1: nz
            if iz < z1
               Vp_sh(iz+zref,ix) = Vp_801(iz,ix);
           else
               Vp_sh(iz,ix) = Vp_801(iz,ix);
           end
        end
    end
    for ix = x2+1 : nx
        for iz = 1:nz
           Vp_sh(iz,ix) = Vp_801(iz,ix);
        end % for -loop
    end % ix window - loop
end % ix -loop

Vp_shift = Vp_sh(1:411,:)+390;

imagesc(Vp_801)
figure(2)
imagesc(Vp_shift)
```

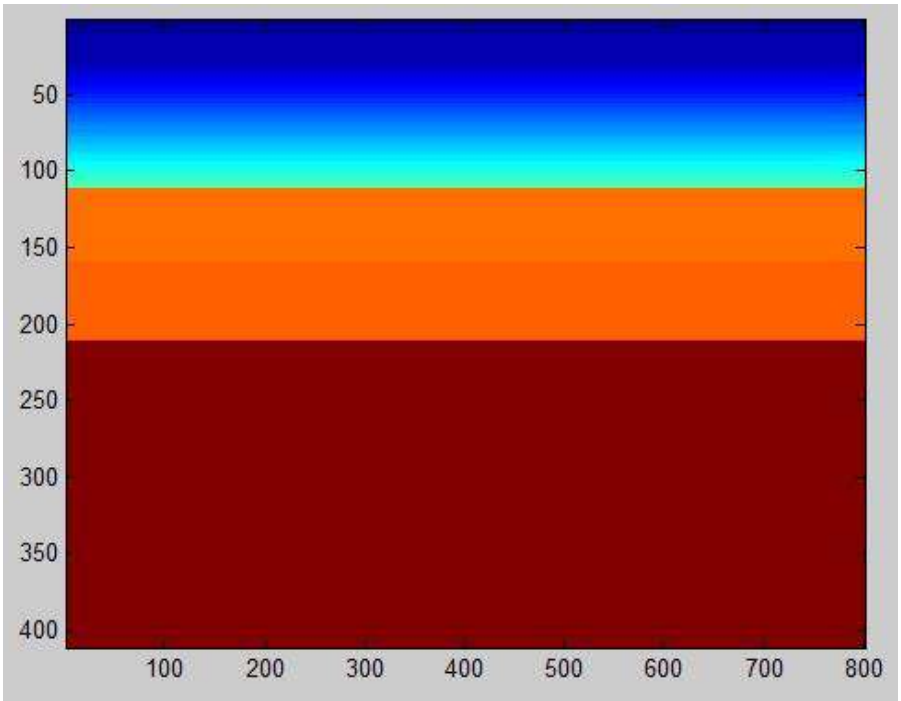

Figura 4.22: Model cu trei strate cu stratificaţie orizontală nefaliat

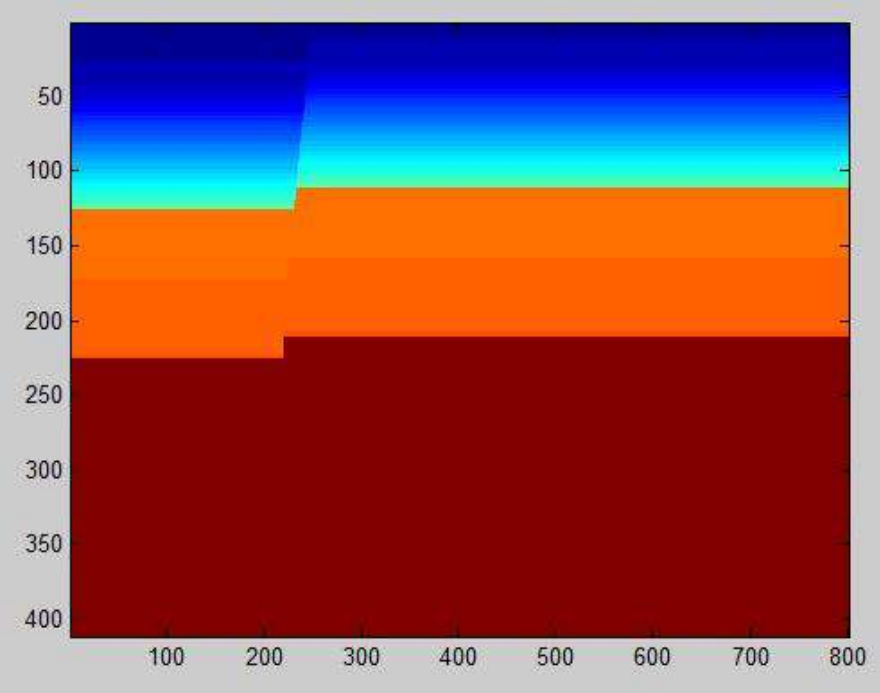

Figura 4.23: Model cu trei strate cu stratificaţie orizontală faliat

E12: Modelul prezentat în Figura 4.24 cu datele de intrare de mai jos se poate defini astfel:

```
clear
nlinie=50
ncol=10
L1=10
L2=25
L3=40
V1=1500
V2=2000
V3=3000
V4=4000
dvel=10
L_stg1 = 20 % număr de linie în stânga modelului
L_dr1 = 35
dip1 = (L_dr1-L_stg1)/ncol
dcol = 5 % metri
for iL=1:nlinie
    L_exp(iL)=dvel*exp(0.5+0.1*iL);
end
L_exp

for icol=1:ncol
```

```
    dist(icol)=(icol-1)*dcol;
    L_dip1(icol)=L_stg1+round((icol-1)*dip1);
end
L_dip1
for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie<=L_dip1(icol)
            vel(ilinie,icol)=V1+L_exp(ilinie);
        else
            if ilinie<=L2
                vel(ilinie,icol)=V2+L_exp(ilinie);
            elseif ilinie>=L3
                vel(ilinie,icol)=V4;
            else
                vel(ilinie,icol)=V3+L_exp(ilinie);
            end
        end
    end
end
imagesc(vel)
xlabel('Distanta (m)')
ylabel('Adancime (m)')
```

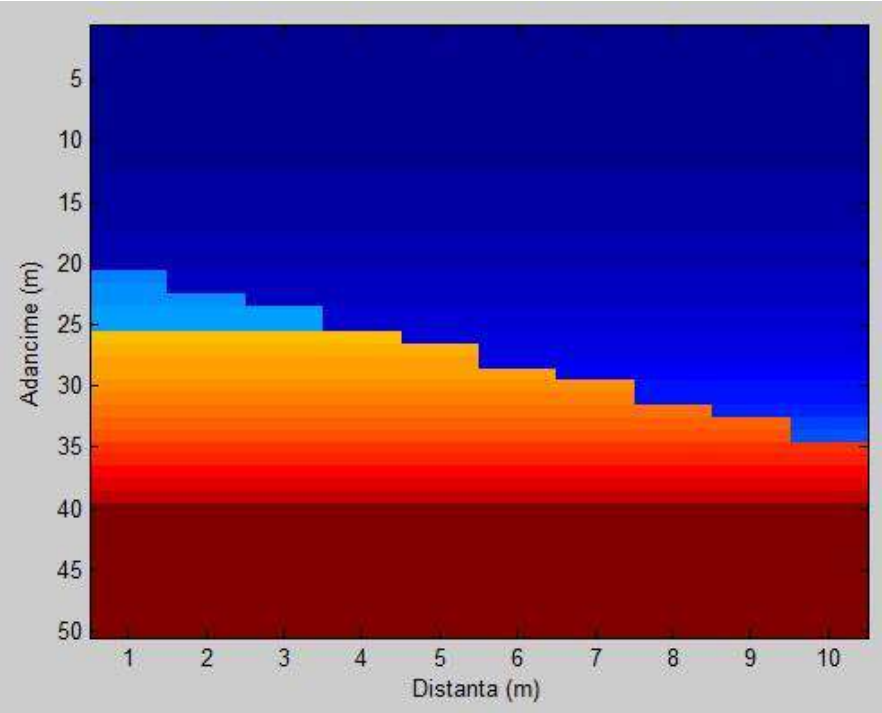

Figura 4.24: Model de viteză cu variaţie pe verticală a vitezei

E13: Reprezentaţi funcţia de reflectivitate pentru un model cu prima limită orizontală și a doua înclinată. Coeficientul de reflexie pentru prima limită (R1) este -0.7 iar pentru a doua limită (R2) este 0.7. Funcţia este prezentată în Figura 4.25.

```
clear
nlinie=50
ncol=10
L_stg1 = 20 % număr de linie în stânga modelului
L_dr1 = 35
dip1 = (L_dr1-L_stg1)/ncol
for icol=1:ncol
    L_dip1(icol)=L_stg1+round((icol-1)*dip1);
end
for ilinie=1:nlinie
    for icol=1:ncol
        if ilinie==L_dip1(icol)
            FR(ilinie,icol)=0.7;
        elseif ilinie==5
            FR(ilinie,icol)=-0.7;
        else
            FR(ilinie,icol)=0.;
        end
    end
end
plotseis(FR)
xlabel('Distanta (m)')
ylabel('Adancime (m)')
```

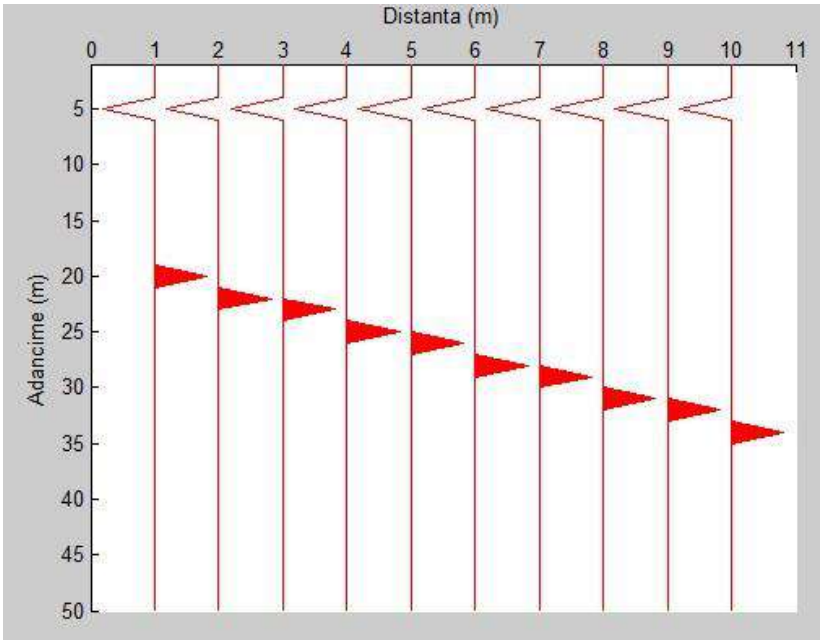

Figura 4.25: Funcţia de reflectivitate pentru un model cu trei strate

Capitolul 5
Încărcarea şi salvarea datelor în format text, SEG-Y şi SU

5.1 Încărcarea datelor în format text

Încărcarea fişierelor text se face folosind funcţia `load`:

```
text=load('D:\Matlab\REC_MATLAB.prn');
```

Pentru verificare în *Command Window* se tastează:

```
whos
```

şi se obţine:

```
>> text=load('D:\Matlab\REC_MATLAB.prn');
>> whos
  Name        Size            Bytes  Class     Attributes
  text      466x4             14912  double
```

Selectarea şi reprezentarea tuturor valorilor de pe coloana a 4-a se face folosind (Figura 5.1):

```
plot(text(:,4))
xlabel('Număr receptoare')
ylabel('Elevaţie (m)')
```

5.2 Încărcarea datelor în format SEG-Y

Un fişier salvat în format SEG-Y se încarcă folosind funcţia `segyread`:

```
>> St=segyread('D:\Matlab\SH3_agc.sgy');

>> whos
  Name          Size              Bytes  Class     Attributes
  St         2000x72            1152000  double
```

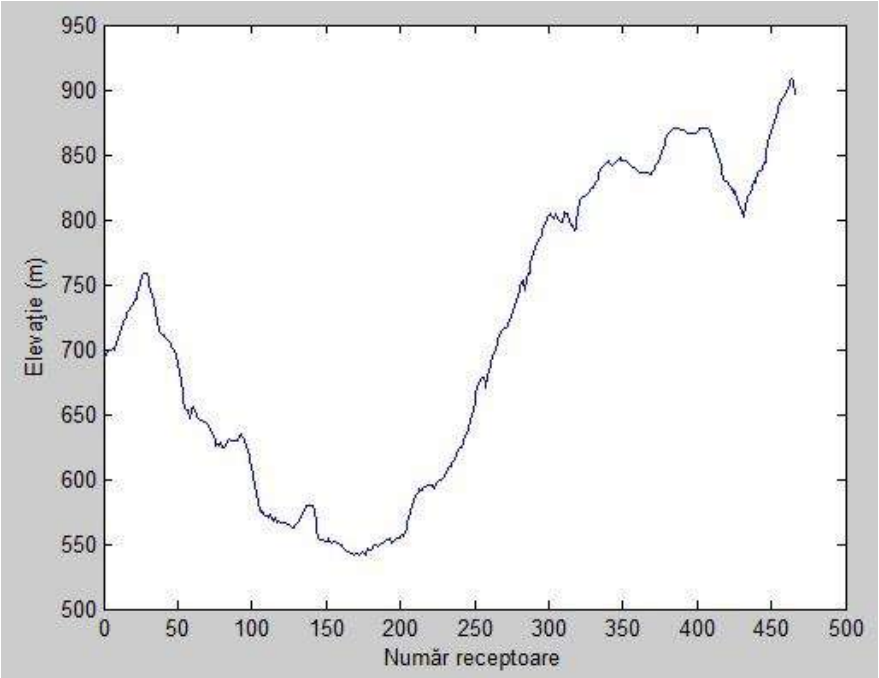

Figura 5.1: Reprezentarea valorilor de pe coloana a 4-a din fisierul „text”

Pentru reprezentarea grafică se foloseşte funcţia `plotimage` (Figura 5.2):

```
plotimage(St)
axis([0 72 0 500])
```

Seismograma prezentată în Figura 5.2 necesită aranjarea traselor după încărcarea lor în variabila `St` după poziţia lor corectă pe profil folosind:

```
St=segyread('D:\Matlab\SH3_agc.sgy');
plotimage(St)
St0=[St(:,1:24) fliplr(St(:,25:48))];
plotimage(St0)
xlabel('Număr receptoare')
ylabel('Timp (ms)')
```

Seismograma după aranjarea traselor este prezentată în Figura 5.3.

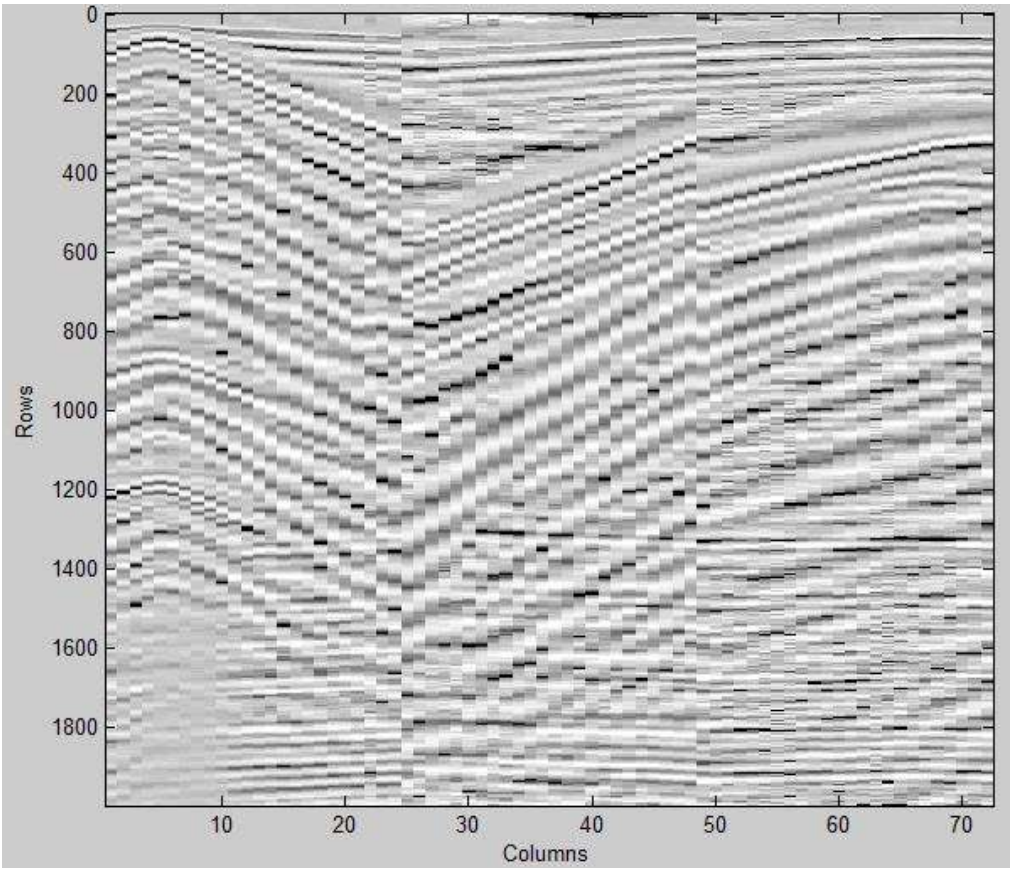

Figura 5.2: Seismograma `SH3_agc.sgy` înainte de aranjarea traselor după poziţia lor corectă pe profil

5.3 Încărcarea datelor în format Seismic Unix (SU)

Un fişier salvat în format SU se încarcă folosind funcţia `ReadSu` din CREWES:

```
St=ReadSu('D:\Matlab\shot101_inline.su');
plotimage(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

Panoul de zgomot ambiental salvat în variabila `St` este prezentat în Figura 5.4.

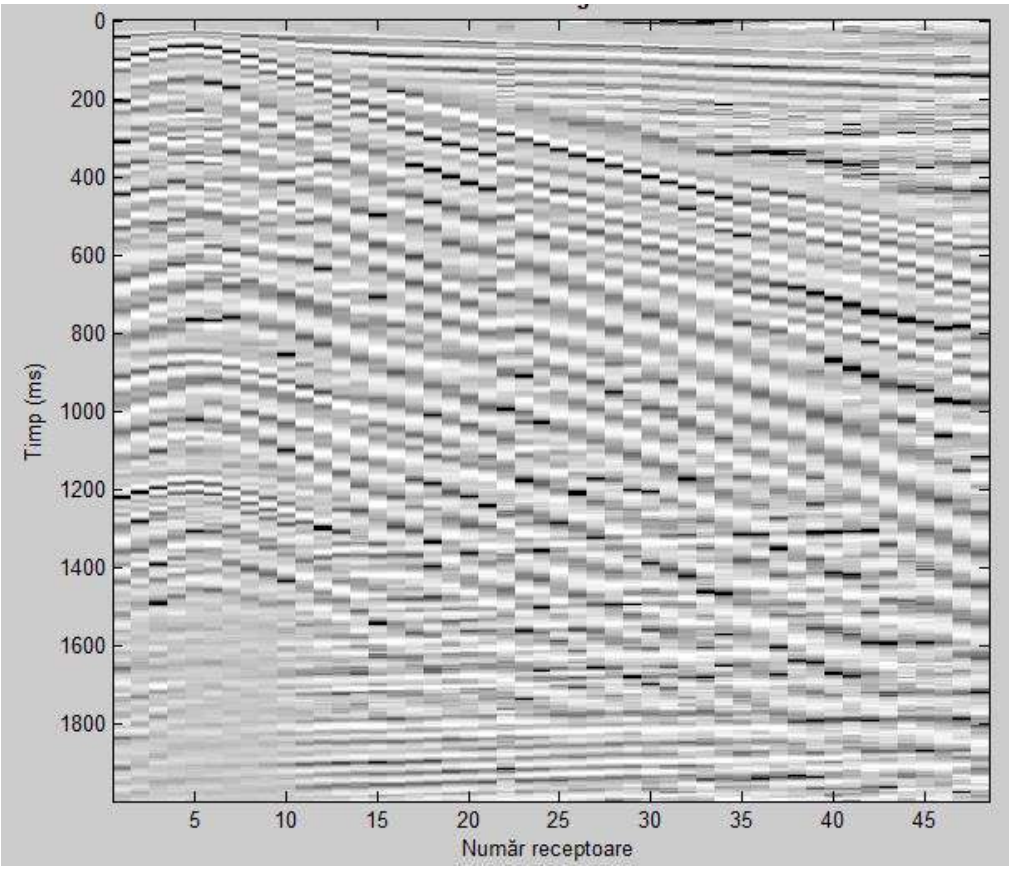

Figura 5.3: Seismograma `SH3_agc.sgy` după aranjarea traselor după poziţia lor corectă pe profil

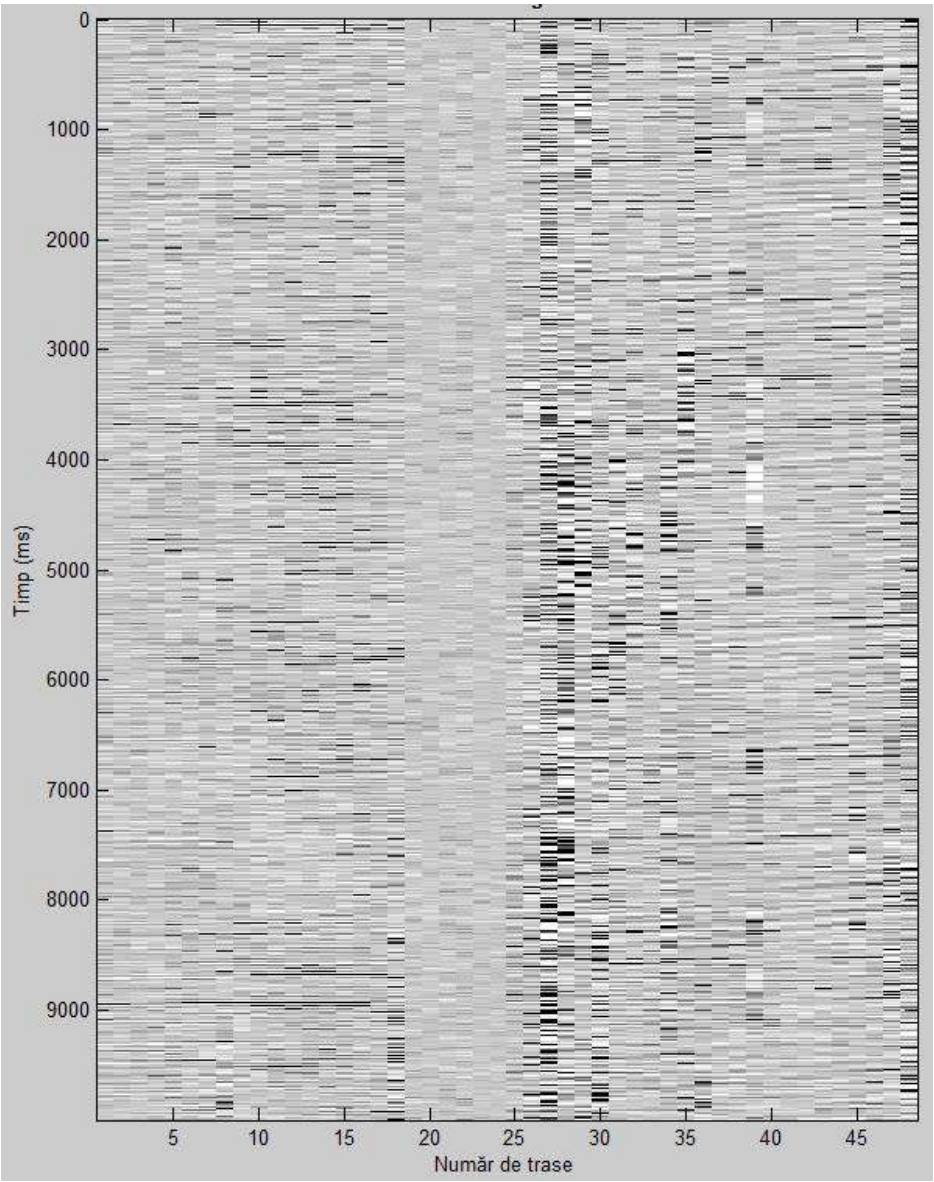

Figura 5.4: Panou de zgomot ambiental încărcat folosind funcţia `Readsu`

Editarea traselor cu numerele 19-24 în seismograma prezentată în Figura 5.4 se poate efectua folosind:

```
clear
St=ReadSu('D:\Matlab\shot101_inline.su');
nt=max(size(St))
nx=min(size(St))

for ix=1:nx
    if ix<19
        St0(:,ix)=St(:,ix);
```

```
    elseif ix > 24
        St0(:,ix)=St(:,ix);
    else
        St0(:,ix)=St(:,ix)*0.0;
    end
end
plotimage(St0)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

Rezultatul este prezentat în Figura 5.5.

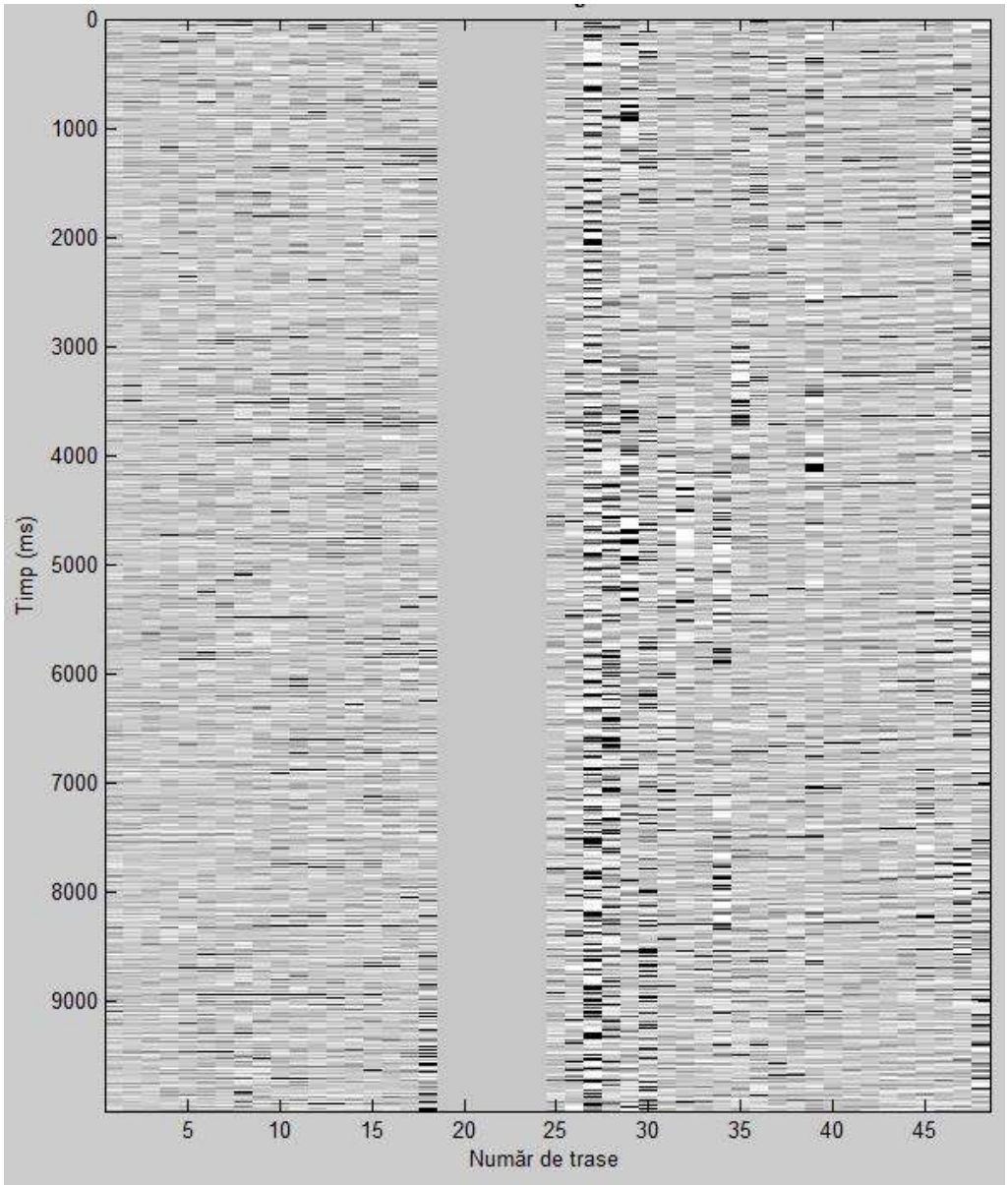

Figura 5.5: Panou de zgomot ambiental încărcat folosind `Readsu` după editarea traselor zgomotoase

5.4 Salvarea fişierelor în formatul text

Fişierele pot fi salvate în formatul MATLAB folosind:

```
>> save L_exp.prn L_exp
```

sau în formatul text (.prn) în directorul curent din Matlab folosind:

```
>> save L_exp.prn L_exp -ascii
```

Fişierul `L_exp` este definit în exerciţiul ***E8*** din Capitolul 4.
Fişierul în formatul text este salvat în directorul dorit dacă se defineşte calea la acel director:

```
>> save D:\MATLAB\L_exp.prn L_exp -ascii
```

Fie matricea `M` definită astfel:

```
M =

         100         105         110         115         120         125
         100         105         110         115         120         125
        2000        2000        2000        2500        2500        2500
        2000        2000        2000        2500        2500        2500
        1500        1500        1500        1500        1500        1500
        1500        1500        1500        1500        1500        1500
```

Din această matrice se selectează coloanele 1 şi 2 şi liniile 3 şi 6 pentru a fi salvate în formatul text. Coloanele şi liniile respective se scriu în vectori separaţi apoi se salvează fiecare vector în formatul text folosind:

```
M_c1 = M(:,1)
M_c2 = M(:,2)
M_l3 = M(3,:)
M_l6 = M(6,:)

save M_c2.prn M_c2 -ascii
```

sau

```
save M_c1.prn M_c1 -ASCII
```

în funcţie de versiunea de MATLAB folosită.

Fişierele text cu coordonate pentru receptoare şi surse, de exemplu, pot fi încărcate şi folosite pentru diferite calcule cerute de definirea geometriei într-un studiu seismic activ sau pasiv.

```
S = load('D:\Matlab\SOU_Matlab.prn');
G = load('D:\Matlab\REC_Matlab.prn');
```

Pentru verificare, citiţi şi afişaţi valorile de pe primele patru linii din variabilele S şi G:

```
S(1:4,:)
G(1:4,:)
```

Distanţele dintre o sursă şi toate receptoarele active pentru acea sursă se calculează folosind:

```
Sx = S(:,3)
Sx_1 = Sx(1)
Sy = S(:,4)
Sy_1 = Sy(1)

nxg=100
for ix=1:nxg
    dist(ix)=round(sqrt((G(ix,2)-Sx_1)^2+(G(ix,3)-Sy_1)^2));
end
```

Salvarea valorilor în formatul text se face folosind:

```
save dist.prn dist -ASCII
```

Distanţele cu semn pozitiv şi negativ în funcţie de poziţia sursei faţă de receptor se pot calcula astfel:

```
n_shot=10
Sx = S(:,3)
Sx_1 = Sx(n_shot)
Sy = S(:,4)
Sy_1 = Sy(n_shot)
nxg=100
for ix=1:nxg
    dist(ix)=round(sqrt((G(ix,2)-Sx_1)^2+(G(ix,3)-Sy_1)^2));
end

for ix=1:nxg
    if G(ix,2)<=Sx_1
        offset(ix)=-1*round(sqrt((G(ix,2)-Sx_1)^2+(G(ix,3)-Sy_1)^2));
```

```
    elseif G(ix,3)<=Sy_1
        offset(ix)=-1*round(sqrt((G(ix,2)-Sx_1)^2+(G(ix,3)-Sy_1)^2));
    else
        offset(ix)=round(sqrt((G(ix,2)-Sx_1)^2+(G(ix,3)-Sy_1)^2));
    end
end

>> offset(1:50)

ans =
Columns 1 through 12
412   397   382   366   352   337   322   308   291   277   262   247
Columns 13 through 24
232   216   202   187   173   157   142   127   112    98    82    67
Columns 25 through 36
52    37    22     7    -8   -23   -38   -53   -69   -83   -98  -113
Columns 37 through 48
-128  -143  -158  -173  -188  -204  -218  -233  -248  -264  -278  -293
Columns 49 through 50
-308  -323

>> dist(1:50)

ans =
Columns 1 through 12
412   397   382   366   352   337   322   308   291   277   262   247
Columns 13 through 24
232   216   202   187   173   157   142   127   112    98    82    67
Columns 25 through 36
52    37    22     7     8    23    38    53    69    83    98   113
Columns 37 through 48
128   143   158   173   188   204   218   233   248   264   278   293
Columns 49 through 50
308   323
```

5.5 Salvarea datelor seismice în formatul SEG-Y

Se încarcă înregistrarea `SH4_agc.SGY` folosind `segyread` şi se reprezintă grafic folosind:

```
St = segyread('D:\Matlab\SH4_agc.SGY');
plotimage(St)
```

Seismograma este prezentată în Figura 5.6.

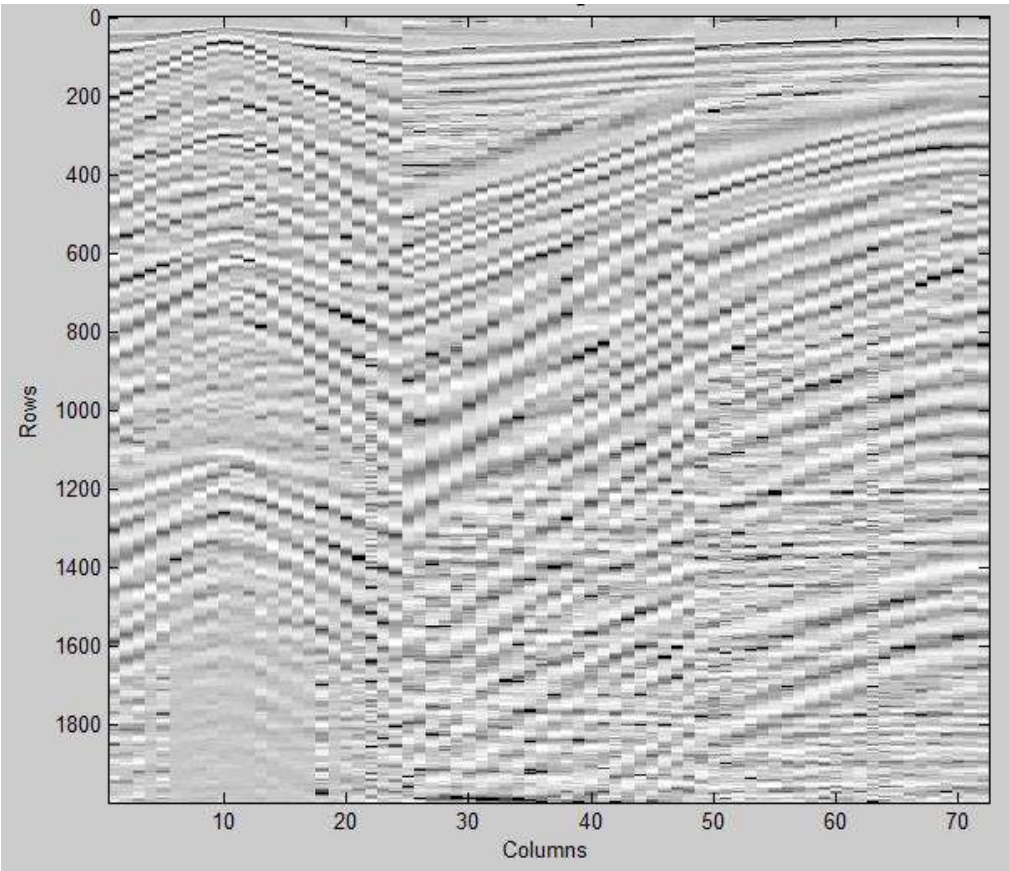

Figura 5.6: Seismograma `SH4_agc.SGY` înainte de aranjarea traselor

Se aranjează trasele în poziţia lor corectă în înregistrare apoi se salvează noua înregistrare în formatul SEG-Y ţinând cont de valoarea intervalului de eşantionare în timp; în acest caz este 0.001 s. Seismograma după aranjarea traselor, `St0`, este prezentată în Figura 5.7.

```
St0 = [St(:,1:24) fliplr(St(:,25:49))];
plotimage(St0)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

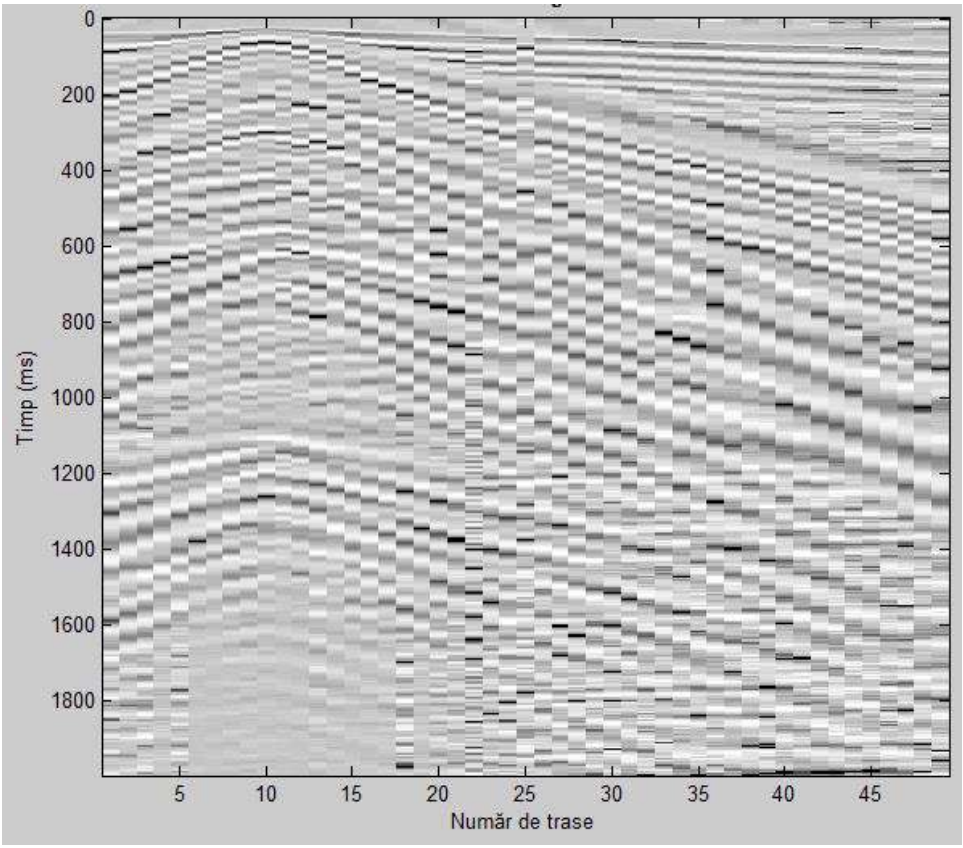

Figura 5.7: Seismograma `SH4_agc.SGY` după aranjarea traselor

Fişierul `writesegy_dt1` se obţine prin editarea valorii intervalului de eşantionare în timp `dt = 0.001` în fisierul `writesegy` din SEGYMAT, unde acesta are valoarea 0.004 s.

```
writesegy_dt1('D:\Matlab\SH4_edit.SGY',St0);
```

```
>> writesegy_dt1('D:\Matlab\SH4_edit.SGY',St0);
WriteSegy_dt1 - No dt set. Using dt=0.001
PutSegyHeader : Using datasample format : 4-byte IEEE floating-point
PutSegyHeader : Using SEG Y revision : 1
PutSegyHeader - SegyRevision 1, 4-byte IEEE floating-point
```

5.6 Salvarea datelor seismice în formatul SU

Seismograma salvată în formatul SEG-Y se poate salva şi în format SU, după editarea corectă a valorii intervalului de eşantionare în timp în fișierul de salvare `writesu_1dt` folosind:

```
writesu_1dt('D:\Matlab\SH4_edit.su',St0);

>> writesu_1dt('D:\Matlab\SH4_edit.su',St0);
WriteSu_1dt - No dt set. Using dt=0.001
>>
```

Fişierul `writesu_1dt` se obţine din editarea `writesu`, unde `dt = 0.004 s`, din SEGYMAT.

Se pot salva date seismice cu `dt = 0.002` s sau orice altă valoare cu condiţia ca aceste valori să fie scrise în fişierele `writesegy` sau `writesu`.

Exerciţii fără rezolvare afişată:

E1: Calculaţi coordonatele `X` şi `Y` ale punctelor de reflexie, `R`, folosind coordonatele din fişierele `SOU_Matlab.prn` şi `REC_Matlab.prn`; un punct de reflexie se află la jumătatea distanţei dintre sursă şi receptor, pentru medii cu stratificaţie orizontală.

E2: Editaţi şi aranjaţi trasele în poziţia lor corectă din înregistrările SH7_agc.SGY, shot1_inline.su, shot1_xline.su, apoi salvaţi rezultatele în format SEG-Y şi SU. Intervalul de esantionare în timp este de 0.001 s.

Capitolul 6
Transformata Fourier

Transformata Fourier 1D (TF 1D) este folosită pentru a converti o înregistrare din domeniul timp sau spatial în domeniul frecvenţă sau număr de undă (Frigo and Johnson, 1998). Transformata Fourier 2D (TF 2D) transformă o înregistrare din domeniul timp-spaţiu în domeniul frecvenţă-număr de undă.

6.1 Transformata Fourier 1D aplicată în domeniul timp

Transformata Fourier 1D se aplică unei singure trase seismice folosind funcţia `fft`. Pentru analiză se încarcă seismograma `SH4_agc.SGY` din care se selectează trasa cu numărul 20 folosind.

```
clear
St0=segyread('D:\Matlab\SH4_agc.SGY');

St = [St0(:,1:24) fliplr(St0(:,25:48))]; % după aranjarea traselor
plotseis(St)

T1 = St(1:500,20); % selectăm trasa cu numărul 20
nt = max(size(T1)) % determinăm numărul de eşantioane în timp
dt = 0.001 % definim interval eşantionare în timp

nf = pow2(floor(log2(nt)+1)) % calculăm număr eşantioane frecvenţă
df = 1/(nf*dt) % calculăm interval eşantionare frecvenţă
freq = df*(1:nf); % definim vector valori frecvenţă (Hz)
```

Se aplică TF 1D şi se reprezentă grafic folosind:

```
fft_T1 = fft(T1,nf); % Transformata Fourier 1D pentru T1
plot(abs(fft_T1)) % reprezentăm fără valori frecvenţe pe axa x
xlabel('Eşantioane de frecvenţă')
ylabel('Amplitudine')
```

Spectrul de amplitudine este prezentat în Figura 6.1. Spectrul de amplitudine al trasei `T1` cu afişarea valorilor de frecvenţă pe abscisă este prezentat în Figura 6.2 folosind:

```
plot(freq,abs(fft_T1)) % reprezentare cu valori frecvenţe pe axa x
xlabel('Frecvenţă (Hz)')
ylabel('Amplitudine')
```

Frecvenţa Nyquist este egală cu 500 Hz. Frecvenţele mai mari decât frecvenţa Nyquist sunt aliasate şi trebuie eliminate din datele seismice folosind:

```
plot(freq(1:nf/2),abs(fft_T1(1:nf/2))) % doar cu frecvenţe pozitive
xlabel('Frecvenţă (Hz)')
ylabel('Amplitudine')
```

Spectrul de amplitudine rezultat este prezentat în Figura 6.3.

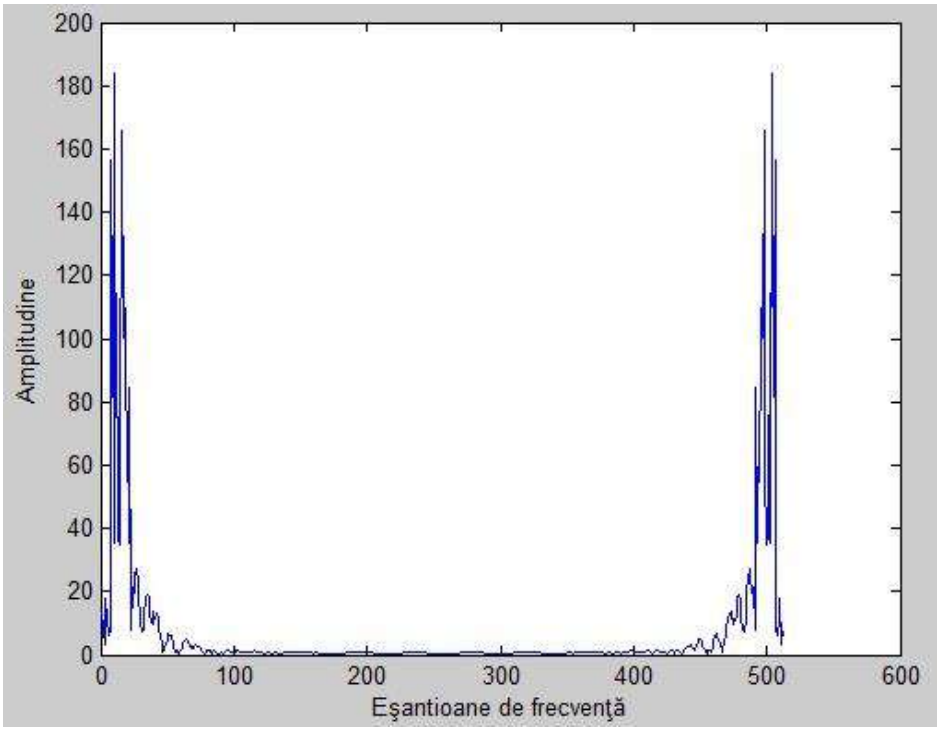

Figura 6.1: Spectrul de amplitudine al trasei `T1` fără afişarea valorilor de frecvenţă pe abscisă

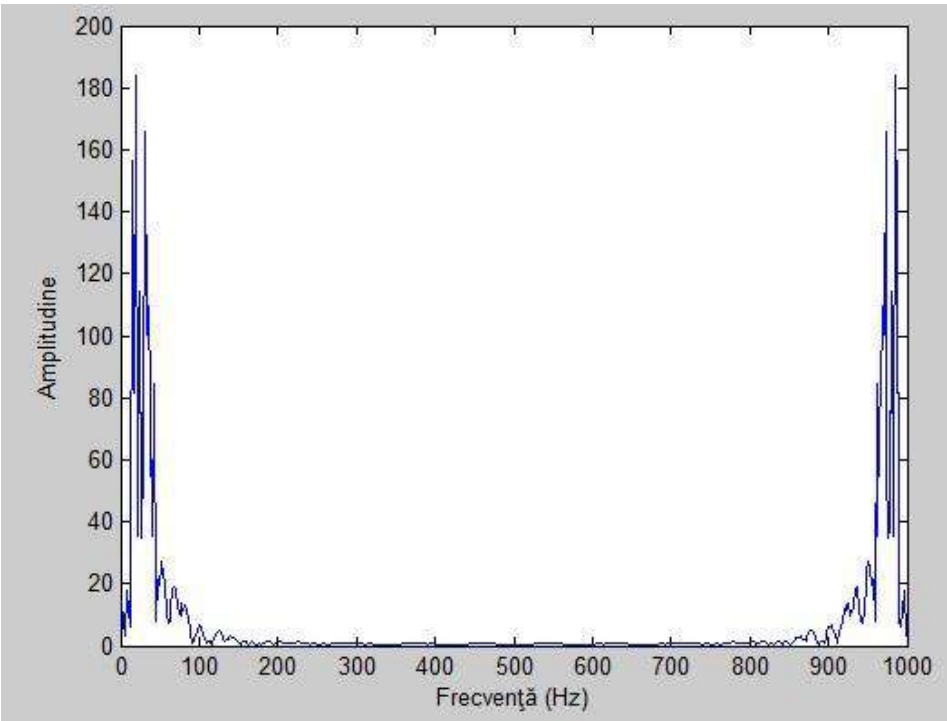

Figura 6.2: Spectrul de amplitudine al trasei `T1` cu afişarea valorilor de frecvenţă pe abscisă

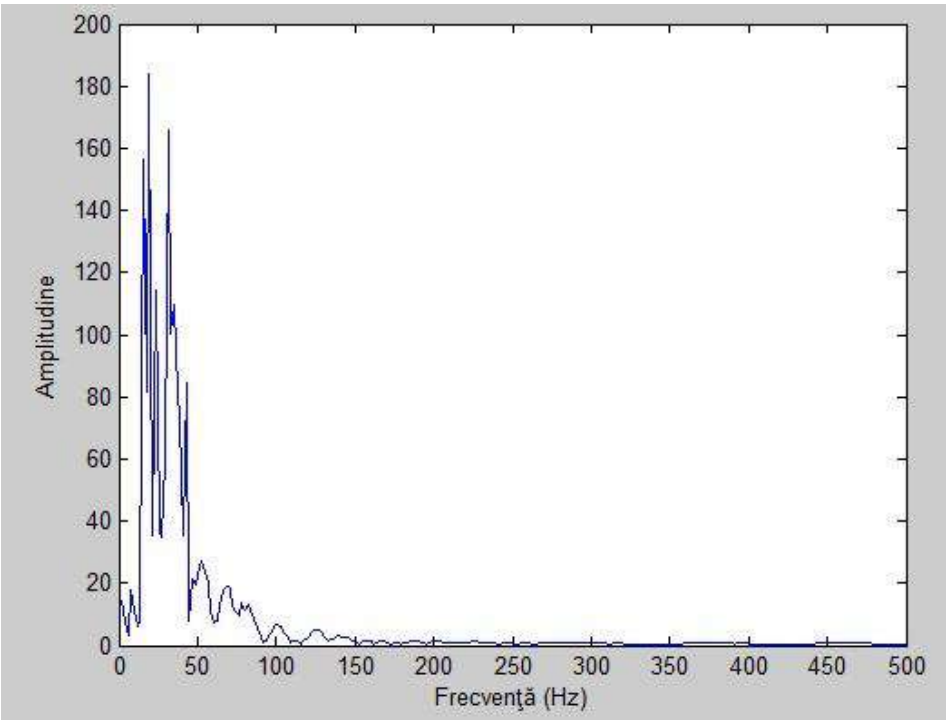

Figura 6.3: Spectrul de amplitudine al trasei `T1` pentru frecvenţe pozitive

Transformata Fourier 1D se poate aplica tuturor traselor dintr-o înregistrare folosind `for-loop`:

```
dt = 0.001 % interval eşantionare timp
dxg = 2.5 % distanţa dintre trase
nx=min(size(St)) % determin număr trase
nt = max(size(St)) % număr eşantioane timp

nf = pow2(floor(log2(nt)+1)) % număr eşantioane frecvenţă
df = 1/(nf*dt) % interval eşantionare frecvenţă
freq = df*(1:nf); % vector valori frecvenţe (Hz)

for ix=1:nx
    nr_trasa(ix)=(ix-1);
    fft_St(:,ix)=fft(St(:,ix),nf);
end
plotseis(abs(fft_St),freq,nr_trasa)
xlabel('Număr de trase')
ylabel('Frecvenţă (Hz)')
```

Spectrele de amplitudine rezultate sunt prezentate în Figura 6.4.

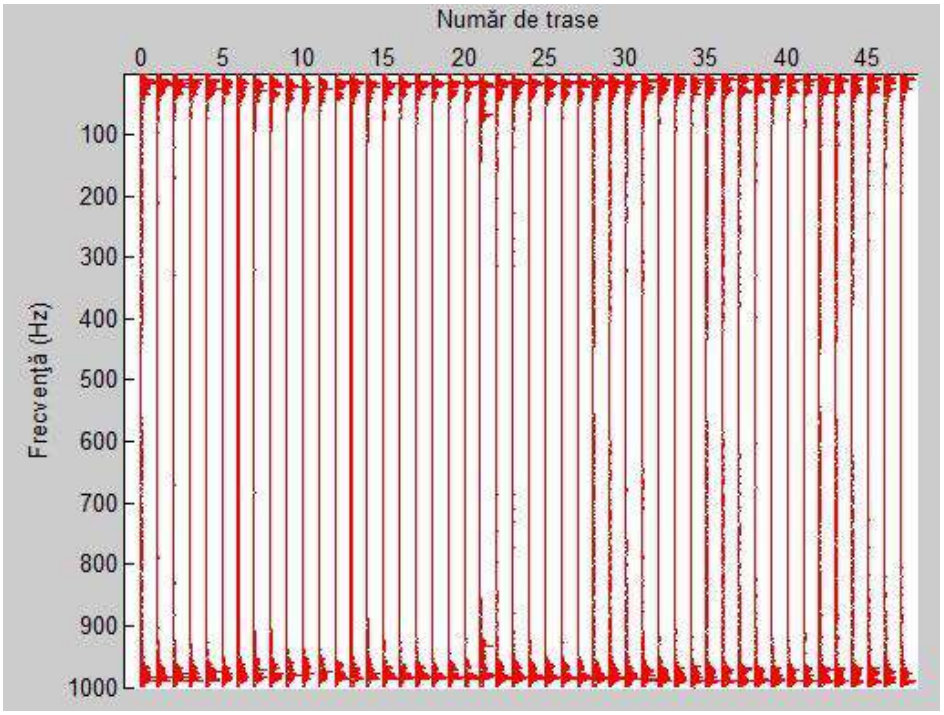

Figura 6.4: Spectrele de amplitudine pentru toate trasele din seismogramă cu afişarea valorilor de frecvenţă

6.2 Transformata Fourier 1D aplicată în domeniul spaţial

În prelucrarea datelor seismice de reflexie este utilă şi analiza în domeniul număr de undă, k. Transformata Fourier 1D se poate aplica tuturor eşantioanelor dintr-o înregistrare care au acelaşi timp de sosire.

```
clear
St0=segyread('D:\Matlab\SH4_agc.SGY'); % citim seismograma
St = [St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St) % reprezentăm în domeniul timp
xlabel('Număr de trase')
ylabel('Timp(ms)')

size_St=size(St) % determinăm mărimea lui St
nt = max(size_St) % determinăm numărul de eşantioane de timp
nxg = min(size_St) % determinăm numărul de trase
dt = 0.001 % definim interval eşantionare timp
dxg = 2.5 % definim distanţa geofoane în metri
nk = pow2(floor(log2(nxg)+1)) % calculăm număr eşantioane număr de undă
dk = 1/(nk*dxg) % calculăm interval eşantionare în număr de undă
k_nr = dk*(1:nk) % calculăm valorile vectorului număr de undă
k_max = 1/(2*dxg) % calculăm numărul de undă Nyquist
k_plot=-k_max+k_nr % vector final pentru reprezentarea în număr de undă
X = St(100,:); % selectăm toate eşantioanele cu timp de sosire 0.1 s
figure(2)
plot(X) % reprezentăm toate eşantioanele cu timp de sosire 0.1 s
xlabel('Număr de trase')
ylabel('Amplitudine')
axis([1 nxg -5 5])

fft_X = abs(fft(X,nk)); % aplicăm Transformata Fourier 1D
plot(abs(fft_X)) % reprezentăm spectrul de amplitudine
xlabel('Eşantioane în număr de undă')
ylabel('Amplitudine')
axis([1 nk 0 50])

% aranjăm valorile după numărul de undă:
fft_out = [fft_X(((nk/2)+1:nk)) fft_X(1:nk/2)];
plot(k_plot,abs(fft_out)) % reprezentăm spectrul de amplitudine
xlabel('Număr de undă (1/m)')
ylabel('Amplitudine')
axis([-k_max k_max 0 50])
```

În Figura 6.5 este reprezentată în domeniul timp seismograma din care s-au selectat toate eşantioanele cu timp de sosire 0.1 s. Trasa obţinută este prezentată în Figura 6.6. Analiza acestei reprezentări nu permite extragerea de informaţii utile despre semnal şi zgomot. După aplicarea TF 1D, analiza spectrului de amplitudine în domeniul număr

de undă (k) permite separarea semnalului la numere de undă mici, în jurul valorii de k = 0, şi zgomotului la numere de undă mari, până la numărul de undă Nyquist. Forma finală a spectrului de amplitudine este obţinută după editarea spectrului în funcţie de valorile numerelor de undă obţinute după aplicarea TF 1D (Figura 6.7).

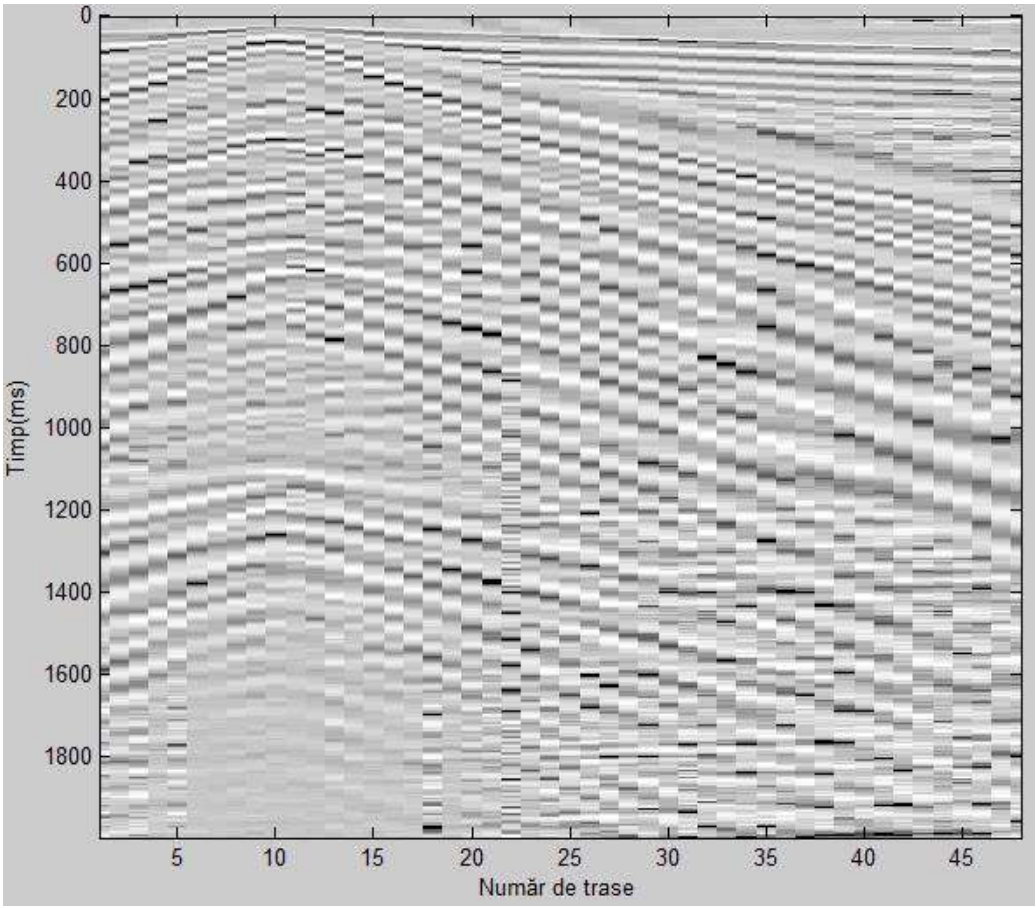

Figura 6.5: Înregistrare seismică din care selectăm toate eşantioanele cu timp de sosire 0.1 s

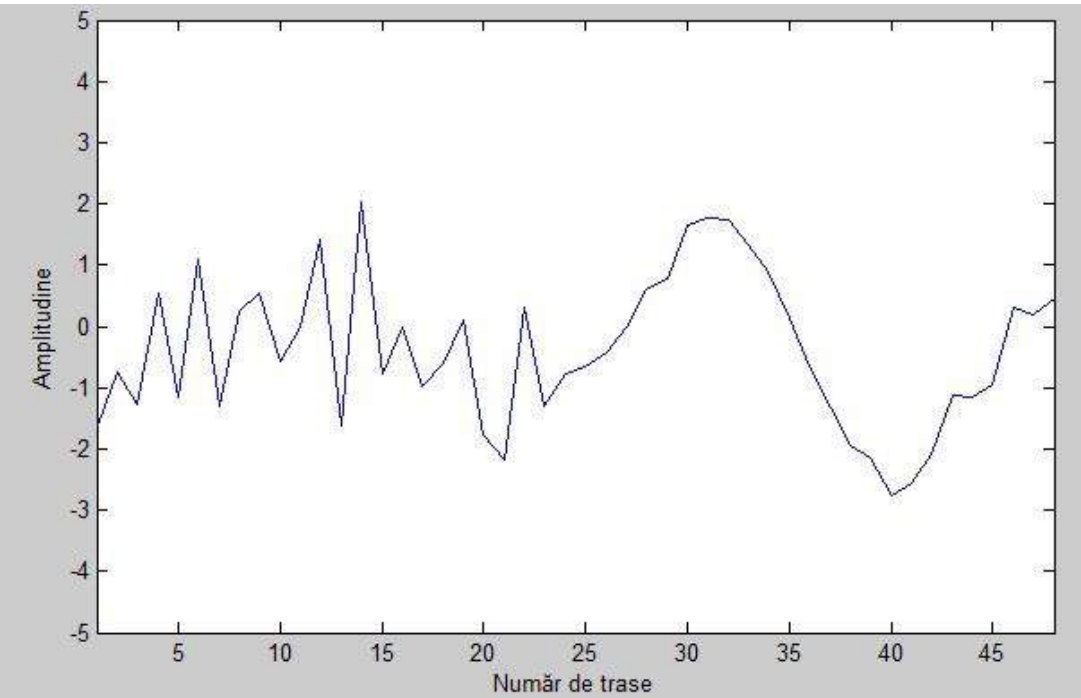

Figura 6.6: Eşantioanele cu timp de sosire de 0.1 s

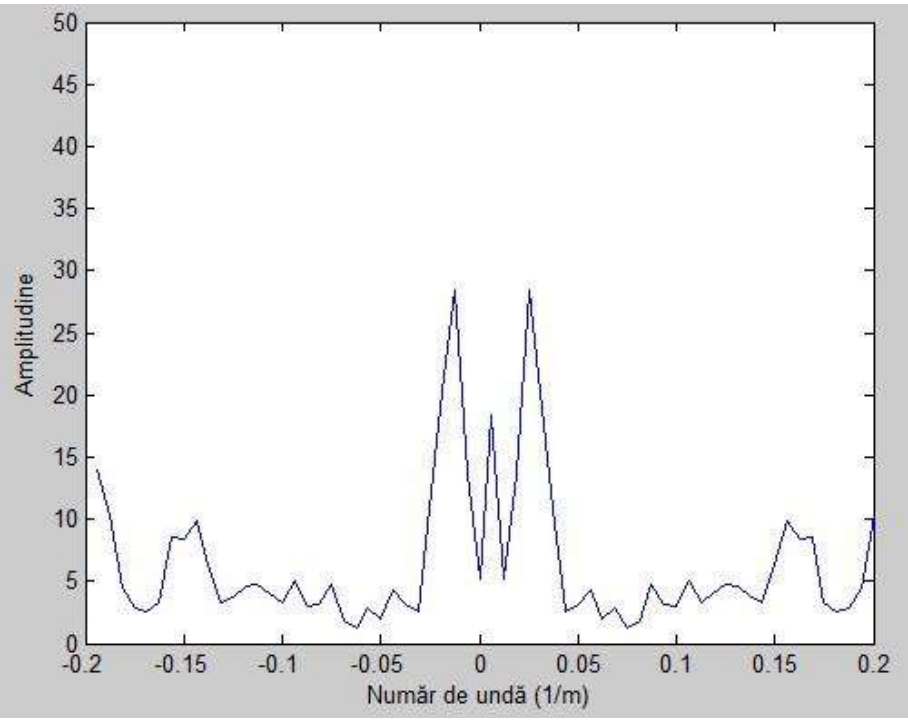

Figura 6.7: Spectrul de amplitudine al TF 1D aplicată tuturor eşantioanelor cu timp de sosire 0.1 s după editarea spectrului

6.3 Transformata Fourier 2D

Transformarea unei înregistrări seismice din domeniul timp-spaţiu în domeniul frecvenţă-număr de undă este realizată folosind Transformata Fourier 2D.

În Figura 6.8 este reprezentată în domeniul timp-spaţiu o înregistrare seismică cu 48 trase. Distanţa dintre geofoane a fost de 2.5 m iar intervalul de eşantionare în timp a fost de 0.001 s. Spectrul de amplitudine prezentat în Figura 6.9 a fost obţinut folosind funcţia `fft2`. Simpla reprezentare a spectrului de amplitudine nu este suficientă pentru analiza corectă a undelor seismice. Rearanjând spectrul după valorile numărului de undă începând cu valoarea negativă a numărului de undă Nyquist, $-k_N$, se obţine reprezentarea corectă a spectrului de amplitudine (f, k), aşa cum este prezentat în Figura 6.10.

```
clear
St0=segyread('D:\Matlab\SH3_agc.sgy');
St=[St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')

dt = 0.001
dxg = 2.5

% NU modificaţi
size_St=size(St)
nxg = min(size_St)
nt = max(size_St)

nf = pow2(floor(log2(nt)+1)) % număr eşantioane frecvenţă
df = 1/(nf*dt) % interval de eşantionare în frecvenţă
freq = df*(1:nf); % vector valori frecvenţă
nk = pow2(floor(log2(nxg)+1)) % număr eşantioane număr de undă
dk = 1/(nk*dxg) % interval de eşantionare in număr de undă
k_nr = dk*(1:nk); % vector valori număr de undă
k_max = 1/(2*dxg) % valoarea maximă in număr de undă
k_plot=k_nr-k_max % vector final pentru reprezentarea fft2

fft2_St = fft2(St,nf,nk);
plotimage(abs(fft2_St),freq,k_nr) % spectrul de amplitudine (f,k)
xlabel('Număr de undă (1/m)')
ylabel('Frecvenţă (Hz)')

abs_fft2_St=fliplr([abs(fft2_St(:,(nk/2)+1:nk))
abs(fft2_St(:,1:nk/2))]); % spectrul final de amplitudine (f, k)

plotimage(abs_fft2_St(1:nf/2,:),freq(1:nf/2),k_plot)
xlabel('Număr de undă (1/m)')
ylabel('Frecvenţă (Hz)')
```

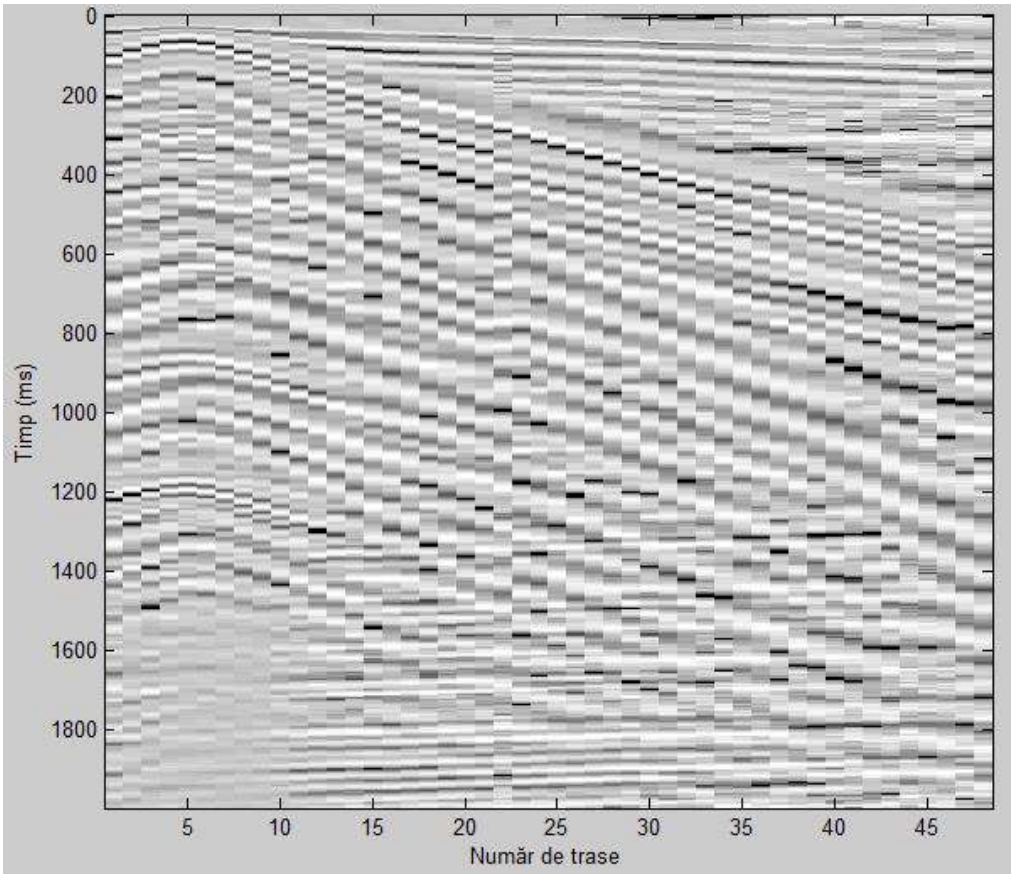

Figura 6.8: Înregistrare seismică reprezentată în domeniul (t, x)

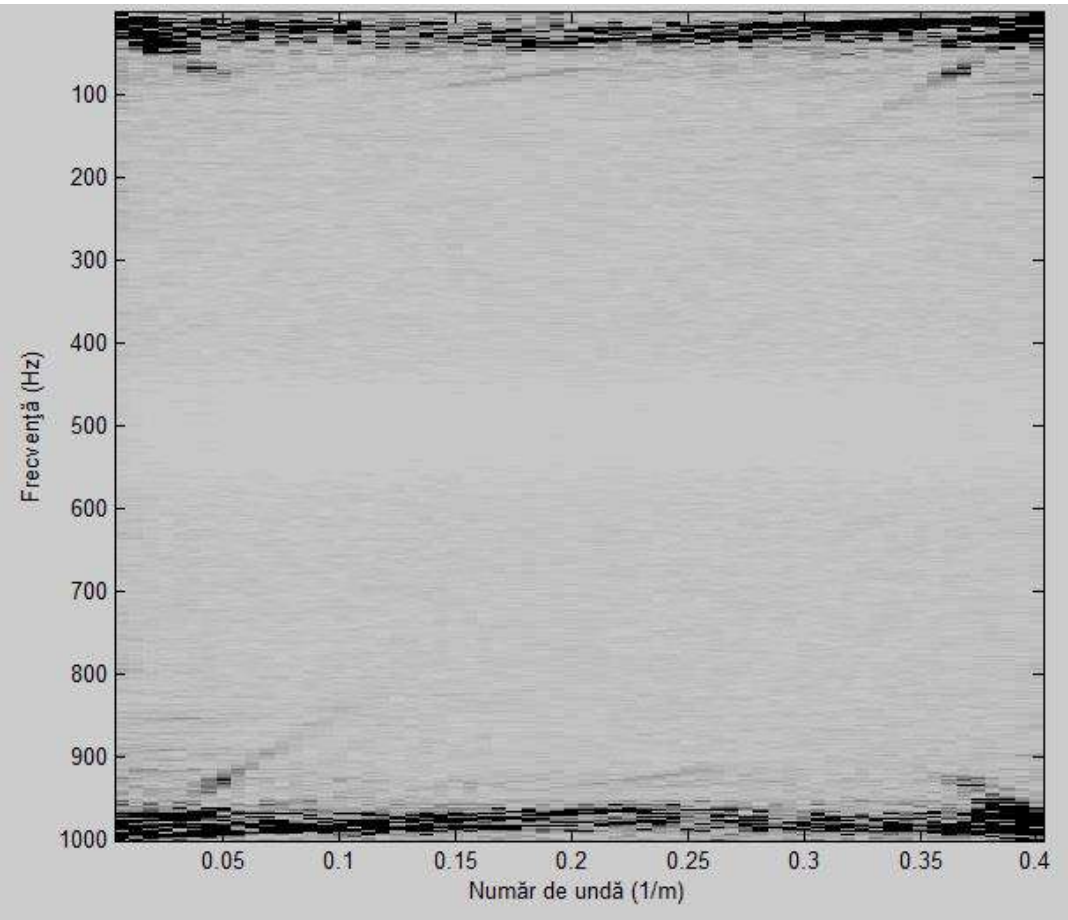

Figura 6.9: Spectrul de amplitudine (f, k) al seismogramei prezentate în Figura 6.8 înainte de editarea după numărul de undă

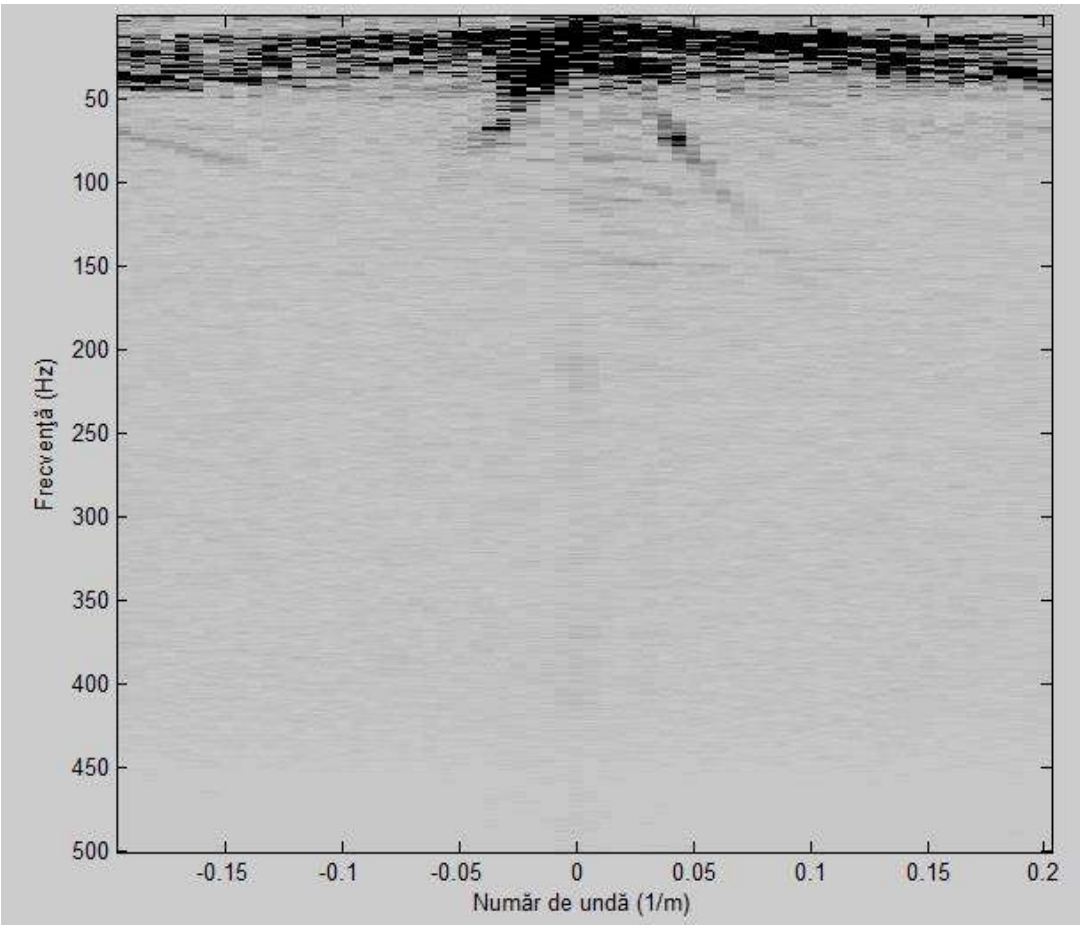

Figura 6.10: Spectrul de amplitudine (f, k) al seismogramei prezentate în Figura 6.8 după editarea după numărul de undă

6.4 Transformata Fourier inversă

Transformata Fourier inversă este folosită pentru a converti semnalul prelucrat în domeniul frecvenţă, număr de undă sau frecvenţă-număr de undă în domeniul timp, spaţiu sau timp-spaţiu. Prelucrarea poate însemna aplicarea unui filtru de frecvenţă trece-bandă sau frecvență-număr de undă (f-k).

Parametrii necesari aplicării TF 1D inverse în domeniul frecvenţă sunt spectrul trasei analizate şi numărul de eşantioane de frecvenţă, `nf`. Pentru TF 1D inversă folosită în domeniul număr de undă, datele de intrare sunt reprezentate de spectrul trasei analizate şi numărul de eşantioane de număr de undă, `nk`. În cazul TF 2D inverse, aceasta se aplică folosind spectrul 2D al înregistrării seismice, numărul de eşantioane de număr de undă, `nk`, şi numărul de eşantioane de frecvenţă, `nf`.

In Figura 6.11 este reprezentată o înregistrare seismică în domeniul timp-spaţiu cu 48 de trase. Distanţa dintre geofoane a fost de 2.5 m iar intervalul de eşantionare în timp

a fost de 0.001 s. Se calculează spectrul de amplitudine (f, k) folosind `fft2` şi se reprezintă după editarea lui în funcţie de numărul de undă (Figura 6.12). După analiza spectrului de amplitudine, selectăm frecvenţa minimă şi cea maximă folosite în filtrarea trece-bandă (Figura 6.13). TF 2D inversă se aplică pe spectrul filtrat pentru a se obţine seismograma filtrată (Figura 6.14). Comparând seismogramele dinainte şi de după filtrarea de frecvenţă se observă că o mare parte din undele de suprafaţă au fost eliminate.

```
clear

% date de intrare:
St0=segyread('D:\Matlab\SH3_agc.sgy');
St=[St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')

dt = 0.001
dxg = 2.5
nf1 = 49 % primul eșantion de frecvență pentru filtrare; F = 24 Hz
nf2 = 140 % ultimul eșantion de frecvență pentru filtrare; F = 68 Hz

% NU modificaţi
% date de intrare în FK:
size_St=size(St)
nxg = min(size_St)
nt = max(size_St)

nf = pow2(floor(log2(nt)+1)) % număr eșantioane de frecvență
df = 1/(nf*dt) % interval de eșantionare în frecvență
freq = df*(1:nf); % vector valori frecvență
nk = pow2(floor(log2(nxg)+1)) % număr de eșantioane în număr de undă
dk = 1/(nk*dxg) % interval de eșantionare in număr de undă
k_nr = dk*(1:nk); % vector valori număr de undă
k_max = 1/(2*dxg) % valoarea maximă în număr de undă
k_plot=k_nr-k_max % vector final pentru reprezentare fft2

nf0 = nf2-nf1
nf_max = (nf/2)-nf2

% calculez FK:
fft2_St = fft2(St,nf,nk);

abs_fft2_St=fliplr([abs(fft2_St(:,(nk/2)+1:nk))
abs(fft2_St(:,1:nk/2))]);
plotimage(abs_fft2_St(1:nf/2,:),freq(1:nf/2),k_plot)
xlabel('Număr de undă (1/m)')
ylabel('Frecvență (Hz)')

% definesc şi aplic vectorul filtru frecvenţă:
zero_freq = [zeros(nf1,1); ones(nf0,1); zeros(nf_max,1)];
for ifreq = 1:nf/2
    for ik=1:nk
```

```
        St_fk_filt(ifreq,ik)=fft2_St(ifreq,ik)*zero_freq(ifreq);
    end
end
plotimage(abs(St_fk_filt),freq(1:nf/2),k_plot)
xlabel('Număr de undă (1/m)')
ylabel('Frecvenţă (Hz)')

% aplic TF 2D inversă:

St_filt = real(ifft2(St_fk_filt,nf,nk));
plotimage(St_filt(1:nt,1:nxg))
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

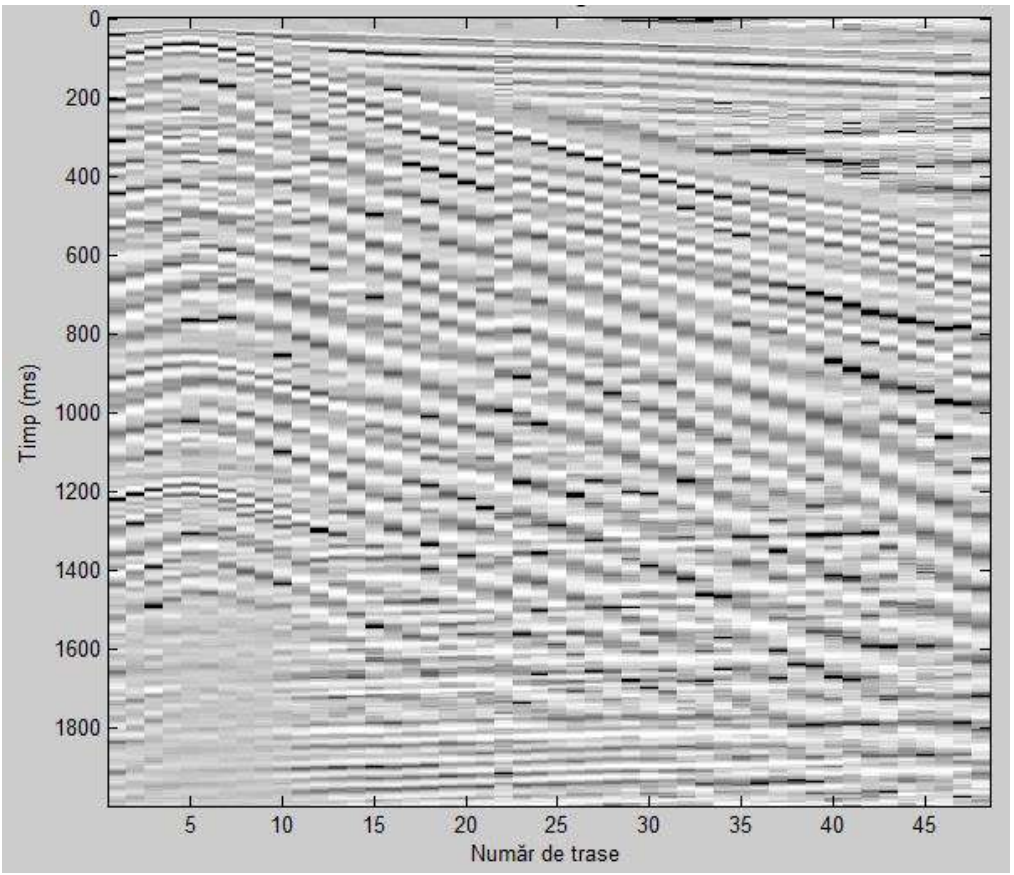

Figura 6.11: Înregistrare seismică reprezentată în domeniul (t, x) înainte de filtrarea de frecvenţă

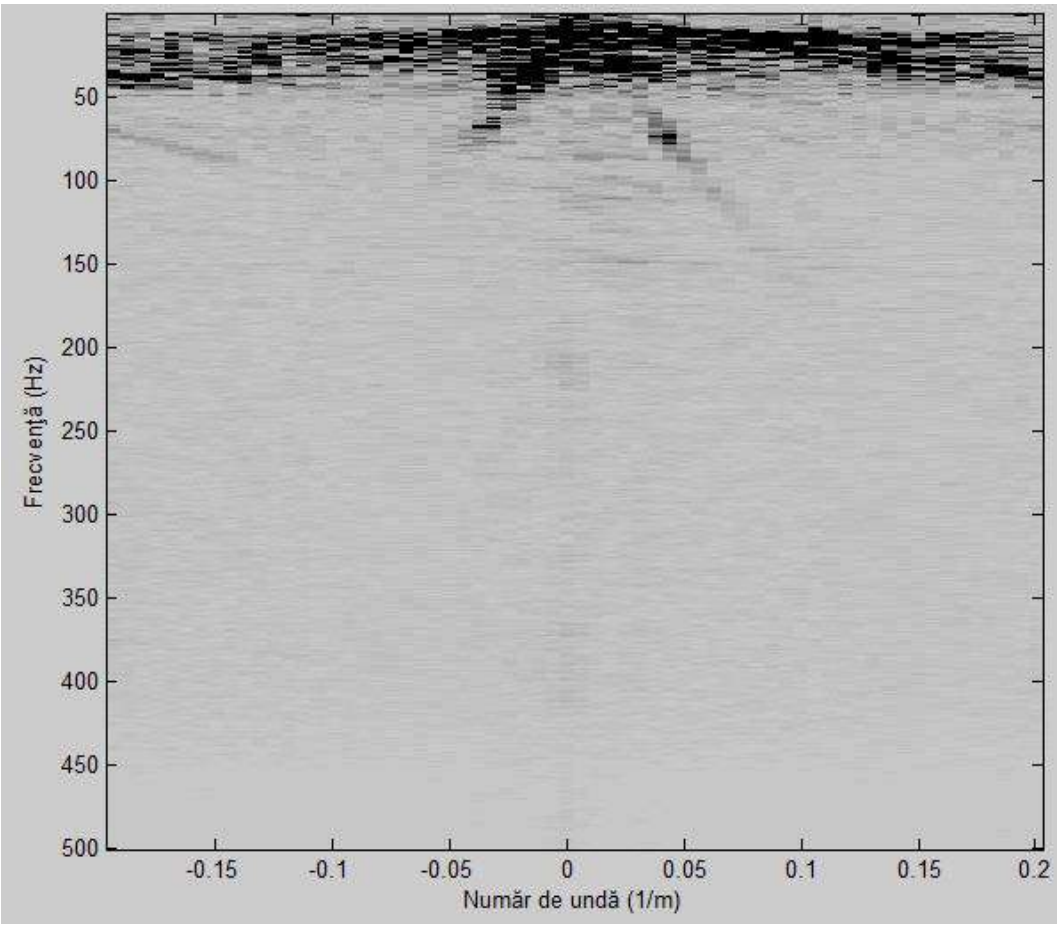

Figura 6.12: Spectrul de amplitudine (f, k) al seismogramei prezentate în Figura 6.11 după editarea după numărul de undă

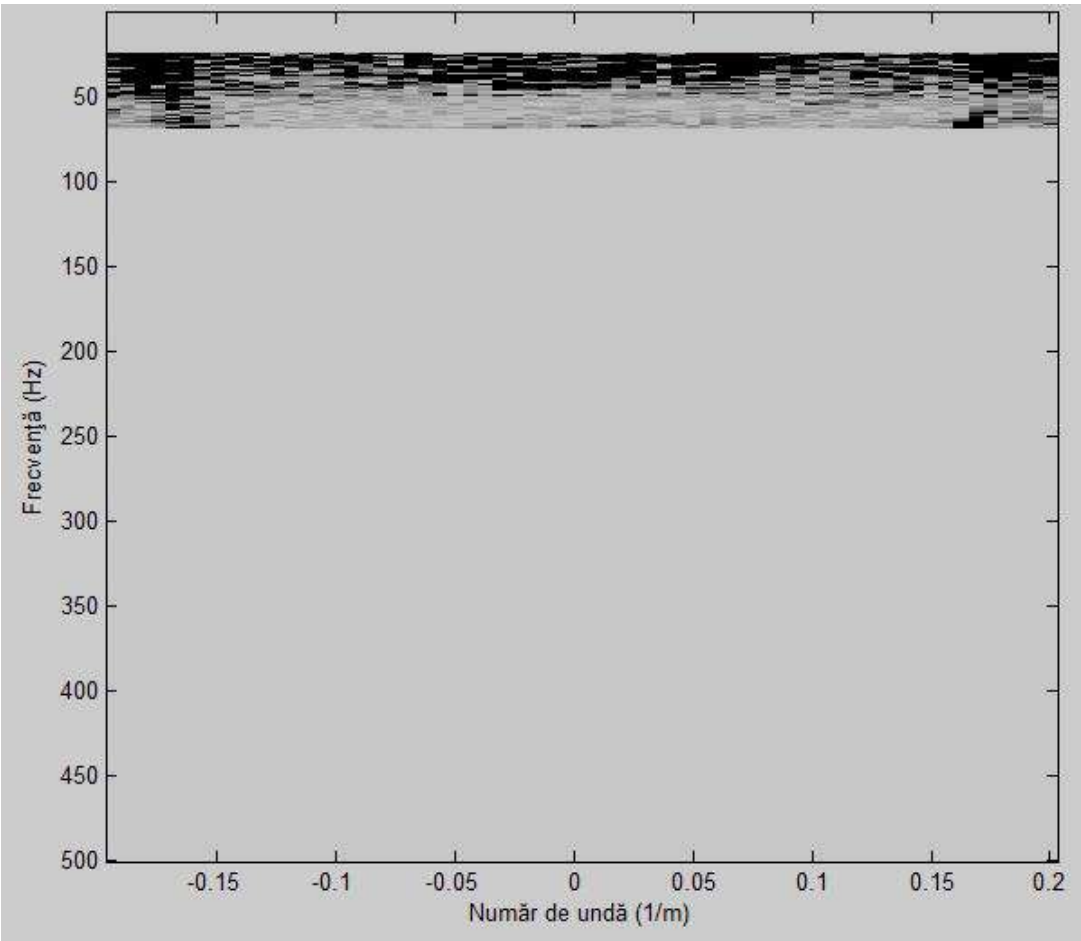

Figura 6.13: Spectrul de amplitudine (f, k) prezentat în Figura 6.12 după filtrarea trece-bandă de frecvenţă

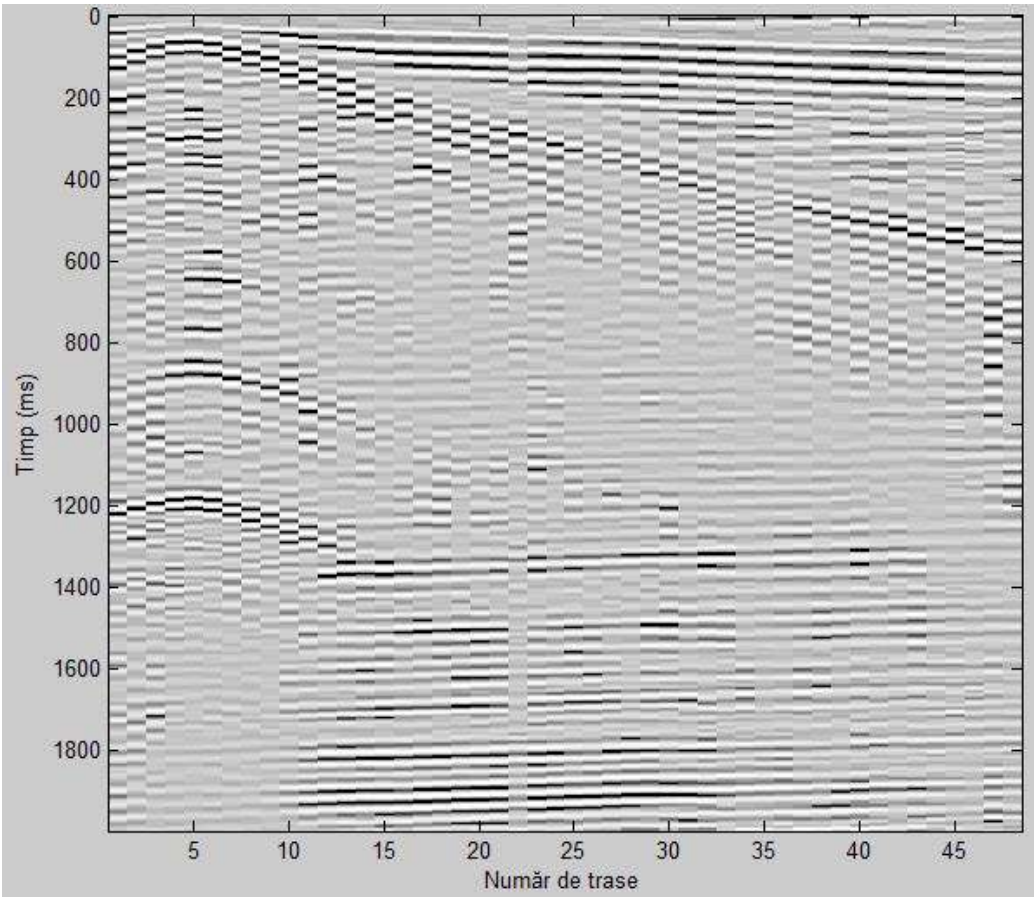

Figura 6.14: Înregistrarea seismică prezentată în Figura 6.11 după filtrarea de frecvenţă

Filtrul trece-bandă se poate aplica folosind ponderi care variază de la 0 la 1 la intrarea în banda de frecvenţe şi ponderi cu valori de la 1 la 0 la ieşirea din bandă. Înregistrarea seismică analizată este prezentată în Figura 6.15. Vectorul pondere este prezentat în Figura 6.16. Spectrul de amplitudine (f, k) după aplicarea filtrului trece-bandă este prezentat în Figura 6.17, iar înregistrarea seismică după aplicarea filtrului este prezentată în Figura 6.18.

```
clear

% date de intrare
St0=segyread('D:\Matlab\SH4_agc.sgy');
St = [St0(:,1:24) fliplr(St0(:,25:48))];
plotseis(St)
xlabel('Număr de trase')
ylabel('Time (ms)')

dt = 0.001 % interval de eşantionare în timp
dxg = 2.5 %2.5 % distanţă receptoare
```

```
nf1 = 204 % 20 Hz
nf2 = 100 % 50 Hz
nf_zoom = 600 % număr eşantion frecvenţă pentru zoom

%%%%%%% NU MODIFICAŢI %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nt = max(size(St)) % număr eşantioane timp
nxg = min(size(St)) % număr geofoane
nf = pow2(floor(log2(nt))+1) % număr eşantioane frecvenţă
df = 1/(nf*dt) % interval eşantionare frecvenţă
freq = df*(1:nf); % vector frecvenţă
nk = pow2(floor(log2(nxg))+1) % număr eşantioane număr de undă
dk = 1/(nk*dxg) % interval eşantionare număr de undă
k_no = dk*(1:nk); % vector număr de undă
kN = 1/(2*dxg) % număr de undă Nyquist
k_plot = k_no-kN % vector număr de undă pentru reprezentare

% calculăm ponderile pentru filtrul trece-bandă
cos_0_1=((cos(0:pi/10:pi)-1)/-2)'
cos_1_0=((1+cos(0:pi/10:pi))/2)'
pondere=[zeros(29,1);cos_0_1;ones(60,1);cos_1_0;zeros(1937,1)];
plot(pondere)
axis([0 1000 0 1.5])
xlabel('Eşantion frecvenţă')
ylabel('Pondere')

% aplicăm transformata Fourier 2D
fft2_St = fft2(St,nf,nk);

% aplicăm filtrul trece-banda pe spectrul de amplitudine (f,k)
for ifreq=1:nf
    for ik=1:nk
        fft2_St_filt(ifreq,ik)=fft2_St(ifreq,ik)*pondere(ifreq);
    end
end
plotimage(freq(1:nf_zoom),k_plot,abs(fft2_St_filt(1:nf_zoom,:)))
xlabel('Număr de undă (1/m)')
ylabel('Frecvenţă (Hz)')

% aplicăm transformata Fourier 2D inversă
St_filt=real(ifft2(fft2_St_filt,nf,nk));
St_filt_fin = St_filt(1:nt,1:nxg);
figure(3)
plotseis(St_filt_fin)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

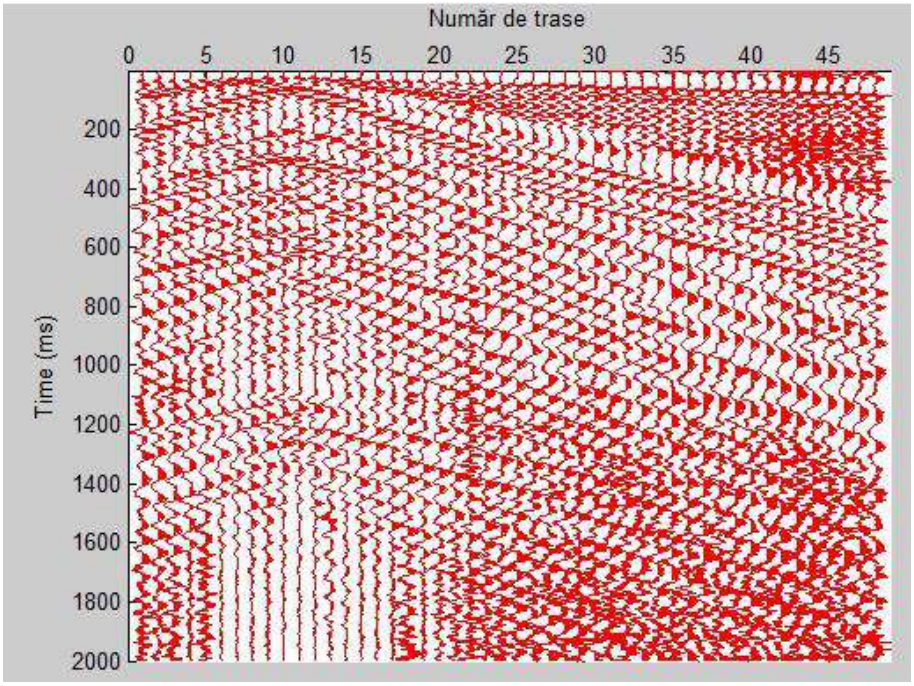

Figura 6.15: Înregistrarea seismică înainte de filtrarea de frecvenţă trece-bandă

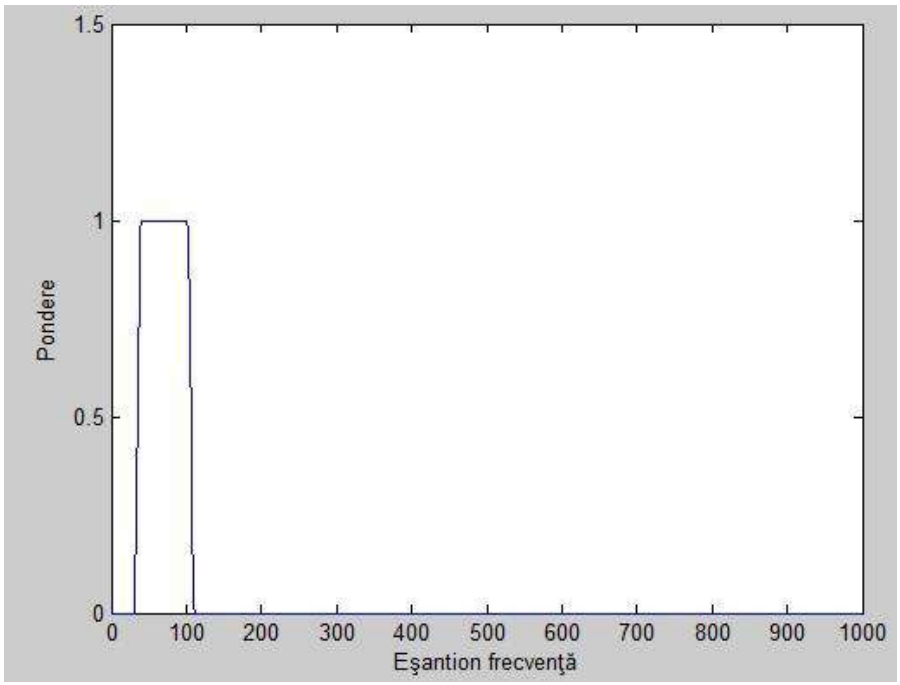

Figura 6.16: Ponderile folosite pentru definirea filtrului trece-bandă

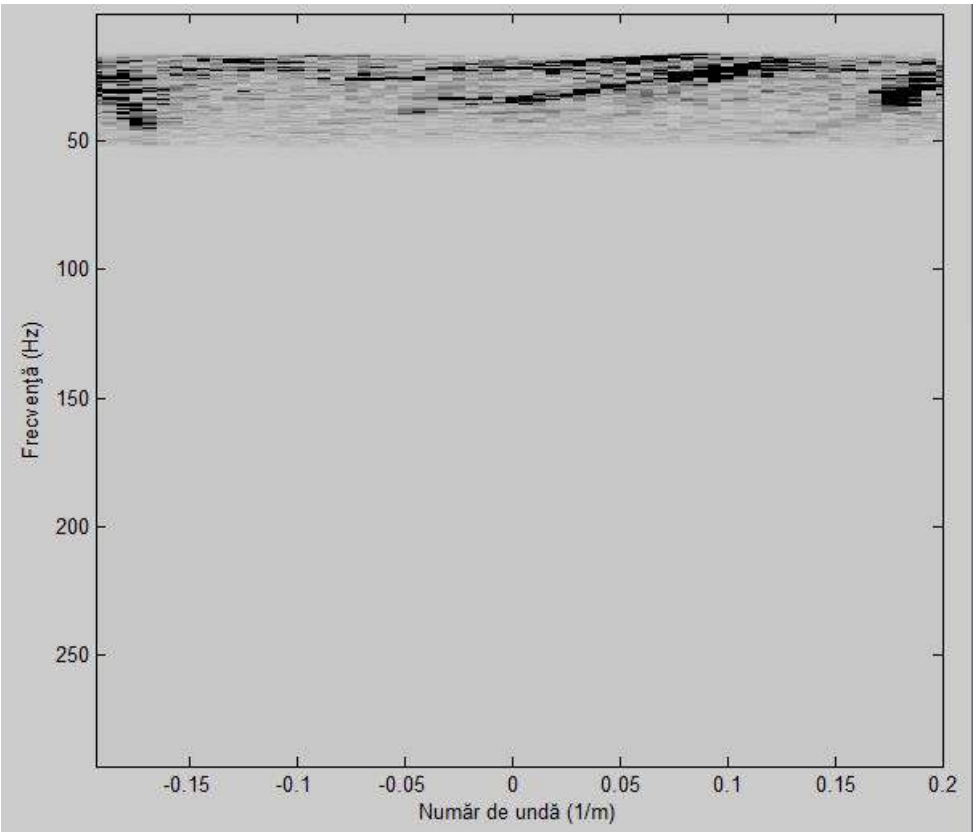

Figura 6.17: Spectrul de amplitudine (f, k) după aplicarea filtrului trece-bandă

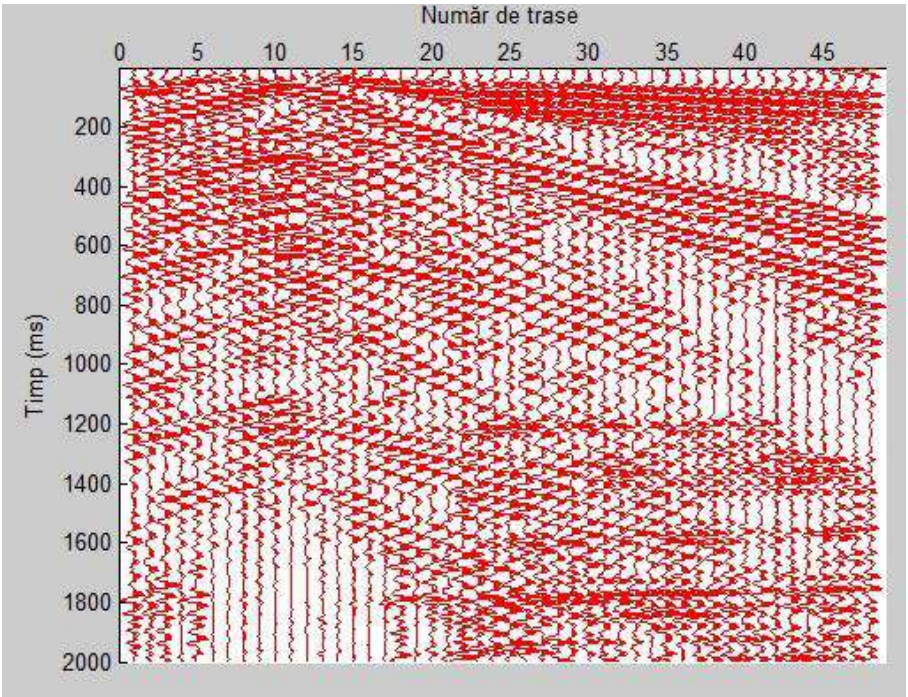

Figura 6.18: Înregistrarea seismică după filtrarea de frecvență trece-bandă

Filtrul f-k se poate aplica folosind un vector ce conține valori de frecvențe variabile cu numerele de undă (Figura 6.19). Rezultatul filtrării f-k este prezentat în domeniile (f, k) și (t, x) în Figurile 6.20 și 6.21. Definirea filtrului f-k și aplicarea acestuia s-a efectuat folosind programul de jos:

```
clear
% date de intrare
St0=segyread('D:\Matlab\SH4_agc.sgy');
St = [St0(:,1:24) fliplr(St0(:,25:48))];
dt = 0.001 % interval de eşantionare în timp
dxg = 2.5  % distanţă receptoare
ng = 50 % număr trase pentru reprezentare folosind plotseis

% definim filtru f-k
nf1 = 144 % număr eşantion frecvenţă pentru frecvenţa 50 Hz
nf2 = 20 % număr eşantion frecvenţă pentru frecvenţa 15 Hz
nf3 = 144 % număr eşantion frecvenţă pentru frecvenţa 50 Hz

%%%%%%%%%%%%%%%% NU MODIFICAŢI %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nt = max(size(St)) % număr eşantioane timp
nxg = min(size(St)) % număr geofoane
nf = pow2(floor(log2(nt))+1) % număr eşantioane frecvenţă
df = 1/(nf*dt) % interval eşantionare frecvenţă
freq = df*(1:nf); % vector frecvenţă
nk = pow2(floor(log2(nxg))+1) % număr eşantioane număr de undă
dk = 1/(nk*dxg) % interval eşantionare număr de undă
k_no = dk*(1:nk); % vector număr de undă
kN = 1/(2*dxg) % număr de undă Nyquist
k_plot = k_no-kN % vector număr de undă pentru reprezentare

if min(size(St)) < ng
    plotseis(St)
else
    plotimage(St)
end
% aplicăm FFT 2D:
fft2_St = fft2(St,nf,nk);
fft2_St_fin = fliplr([fft2_St(:,(nk/2)+1:nk) fft2_St(:,1:nk/2)]);

% definim poligonul pentru filtrul fk
for ik = 1:nk
    if ik <= nk/2
        fk_line(ik)= nf2+(ik-1)*((nf1-nf2)/(nk/2));
    else
        fk_line(ik)= nf3-(ik-(nk/2))*((nf3-nf2)/(nk/2));
    end
end
figure(2)
plot(fk_line)
xlabel('Eşantioane de număr de undă')
ylabel('Număr eşantioane de frecvenţă')

% aplic filtrul fk
```

```
for ifreq=1:nf/2
    for ik=1:nk
        if ifreq<fk_line(ik)
            fft2_St_fk(ifreq,ik) = 0.0;
        else
            fft2_St_fk(ifreq,ik) = fft2_St(ifreq,ik);
        end
    end
end
plotimage(abs(fft2_St_fk))
xlabel('Eşantioane de număr de undă')
ylabel('Număr eşantioane de frecvenţă')

% aplicăm transformata Fourier 2D inversă
St_fk = real(ifft2(fft2_St_fk,nf,nk));
St_fk_fin = St_fk(1:nt,1:nxg);

if min(size(St_fk_fin)) < ng
    plotseis(St_fk_fin)
else
    plotimage(St_fk_fin)
end
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

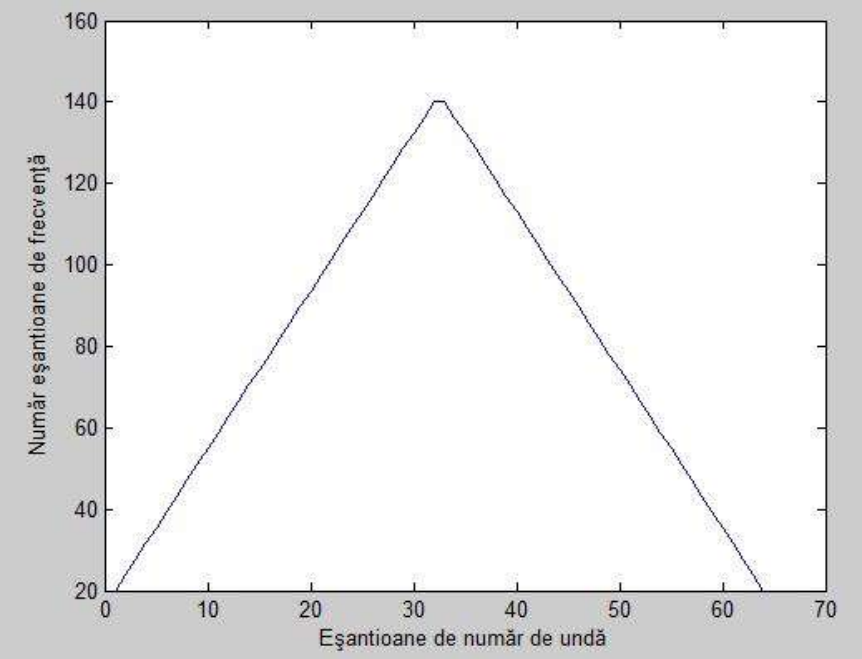

Figura 6.19: Valorile filtrului f-k

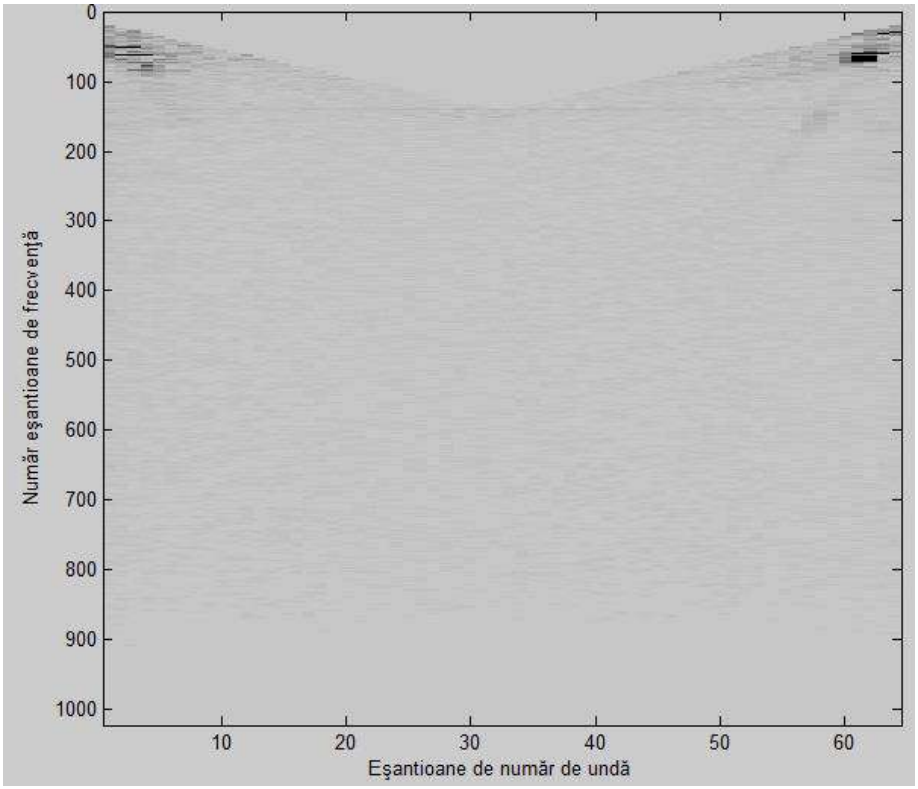

Figura 6.20: Spectrul de amplitudine (f, k) după aplicarea filtrului f-k

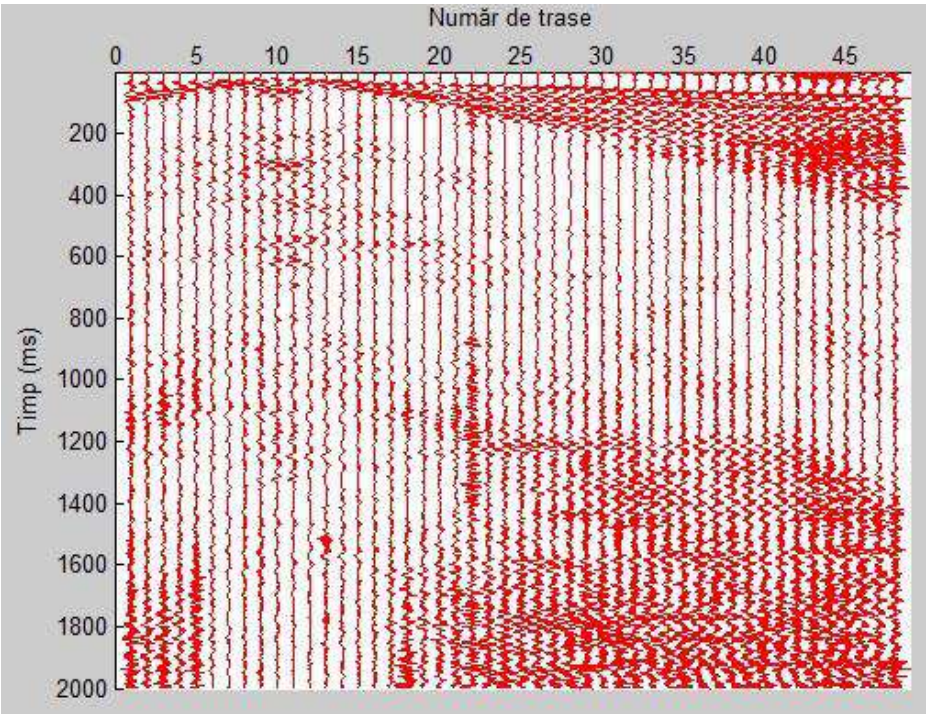

Figura 6.21: Înregistrarea seismică prezentată în Figura 6.15 după filtrarea f-k

Exerciţii fără rezolvare afişată:

E1: Calculaţi şi reprezentaţi grafic spectrele de amplitudine pentru toate trasele din SH7_agc.sgy.

E2: Calculaţi şi reprezentaţi grafic spectrele de amplitudine pentru toate trasele din shot1_inline.su şi shot1_xline.su, înainte şi după editarea traselor zgomotoase.

Capitolul 7
Prelucrarea înregistrărilor seismice în domeniul timp

Zgomotul prezent pe înregistrările seismice se poate elimina folosind filtre ale căror aplicare nu necesită convertirea înregistrării în domeniul frecvenţă şi/sau număr de undă. Top-mute, bottom-mute, surgical-mute şi gruparea de geofoane sunt exemple de astfel de filtre.

7.1 Definirea şi aplicarea top-mute-ului

Acest tip de filtru se defineşte şi se aplică pe o înregistrare seismică reprezentată în domeniul timp (Figura 7.1). Se defineşte un vector ce conţine valorile de timp de pe fiecare trasă seismică deasupra cărora se doreşte ştergerea informaţiei seismice. Un exemplu de program folosit pentru definirea şi aplicarea top-mute-ului este scris mai jos:

```
clear

% date de intrare
St0=segyread('D:\MATLAB\SH3_agc.sgy');
St=[St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St(1:200,:))
xlabel('Număr trase')
ylabel('Timp (ms)')

% date de intrare
dxg = 2.5 % metri, distanţă geofoane
dt = 0.001 % interval eşantionare în timp
nt_zoom = 200

% definesc top-mute
tm1_stg = 20
tm2_stg = 10
ntr_stg = 5

tm1_dr = 10
tm2_dr = 80
ntr_dr = 43

% NU modificaţi
size_St=size(St)
nt = max(size_St)
nxg = min(size_St)
nr_esant_stg = (tm1_stg-tm2_stg)/ntr_stg
tm_stg = round(tm1_stg:-nr_esant_stg:tm2_stg)
nr_esant_dr = (tm2_dr-tm1_dr)/ntr_dr
tm_dr = round(tm1_dr:nr_esant_dr:tm2_dr)
top_mute=[tm_stg(1:ntr_stg) tm_dr(1:ntr_dr)];
figure(2)
```

```
plot(top_mute)
xlabel('Număr trase')
ylabel('Timp (ms)')

% aplic top mute:
for it=1:nt
    for ix=1:nxg
        if it <= top_mute(ix)
            St_tm(it,ix)=0.0;
        else
            St_tm(it,ix)=St(it,ix);
        end
    end
end
figure(3)
plotimage(St_tm(1:nt_zoom,:))
xlabel('Număr trase')
ylabel('Timp (ms)')
```

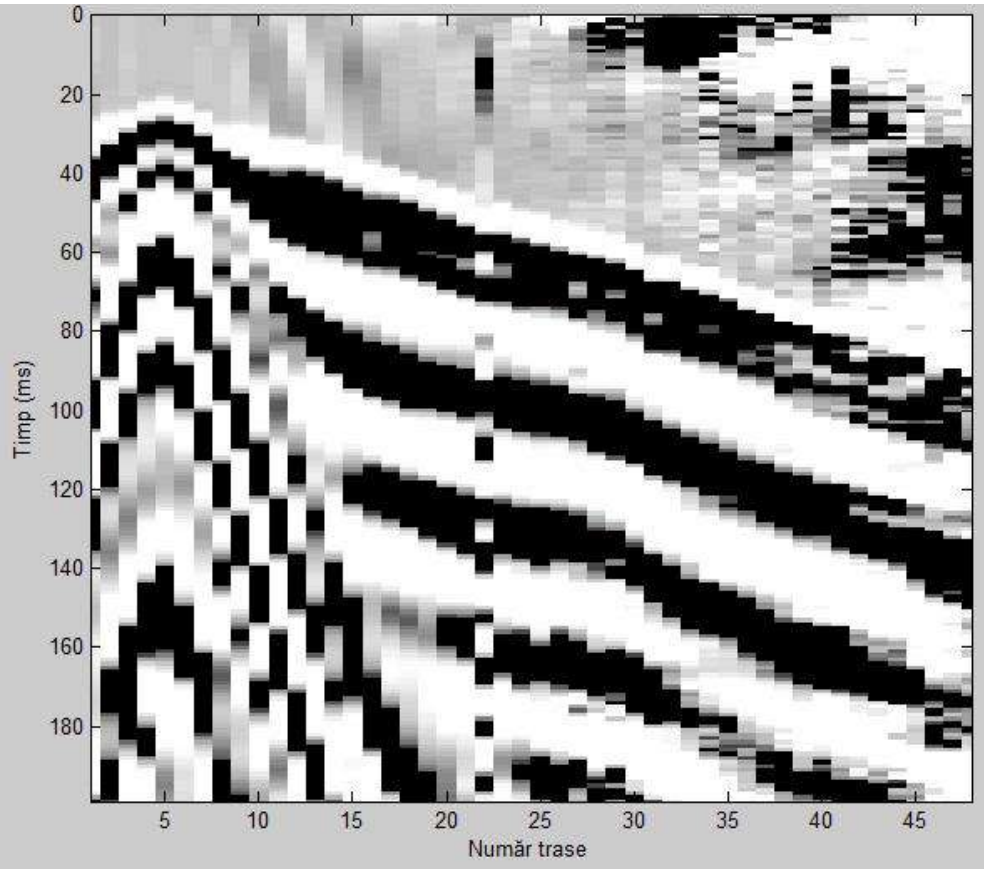

Figura 7.1: Înregistrare seismică înainte de aplicarea unui top-mute

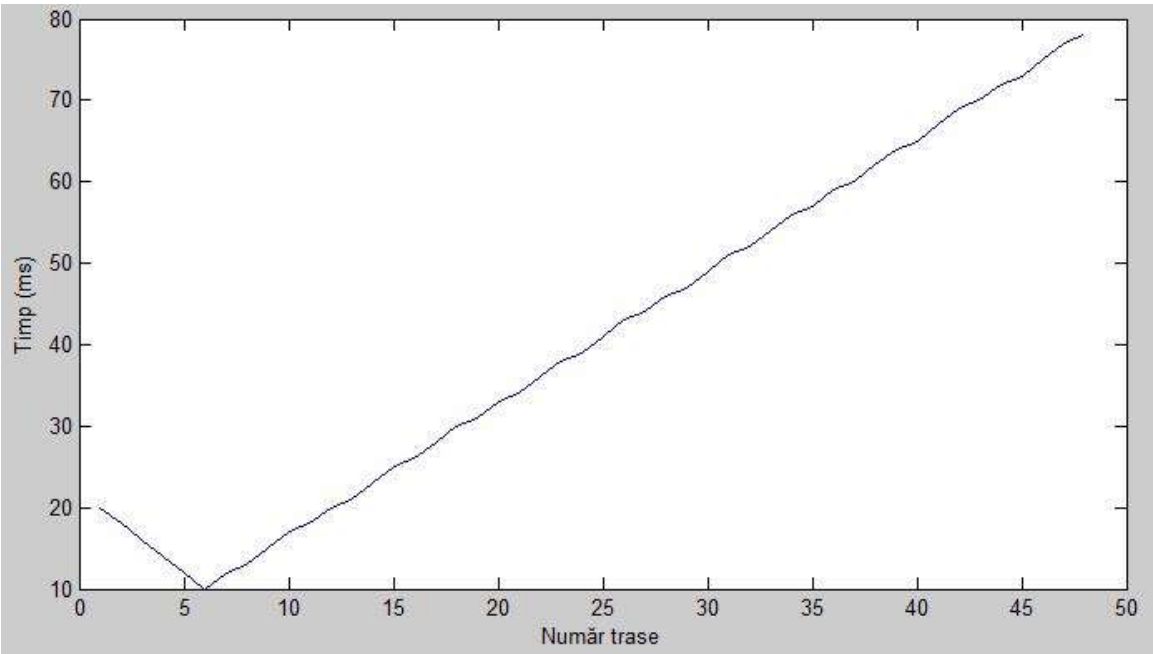

Figura 7.2: Vectorul folosit pentru definirea top-mute-ului

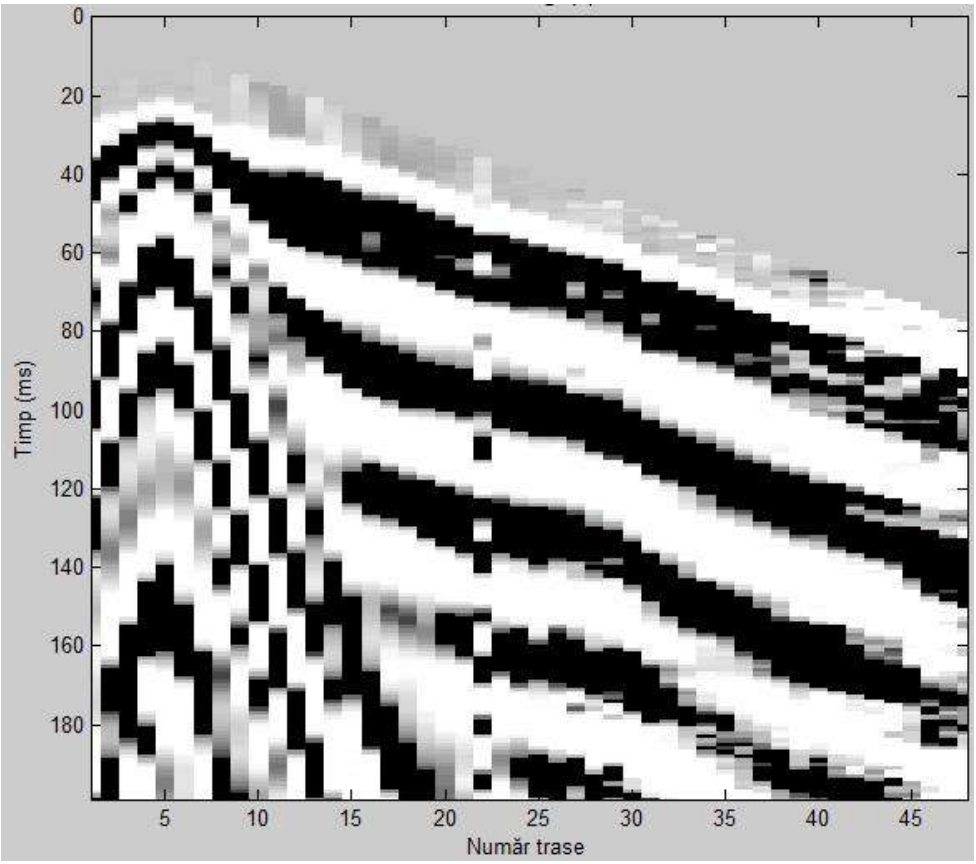

Figura 7.3: Înregistrare seismică după aplicarea unui top-mute

7.2 Definirea şi aplicarea bottom-mute-ului

Acest tip de filtru se defineşte şi aplică pentru a elimina undele seismice cu timpi de sosire mai mari decât cei doriţi. Programul folosit pentru definirea şi aplicarea acestui tip de filtru este scris mai jos. Undele de suprafaţă observate pe înregistrarea prezentată în Figura 7.4 sunt eliminate prin aplicarea unui bottom-mute cu valorile de timp prezentate în Figura 7.5. Seismograma finală este prezentată în Figura 7.6.

```
clear

% date de intrare
St0=segyread('D:\MATLAB\SH3_agc.sgy');
St=[St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St(1:1000,:))
xlabel('Număr trase')
ylabel('Timp (ms)')

% date de intrare
dxg = 2.5 % metri, distanţă geofoane
dt = 0.001 % interval esantionare timp
nt_zoom = 200

% definesc bottom-mute
bm1_stg = 100
bm2_stg = 60
ntr_stg = 5

bm1_dr = 60
bm2_dr = 450
ntr_dr = 43

% NU modificaţi
size_St=size(St)
nt = size_St(1)
nxg = size_St(2)
nr_esant_stg = (bm1_stg-bm2_stg)/ntr_stg
bm_stg = round(bm1_stg:-nr_esant_stg:bm2_stg) % minus un eşantion
nr_esant_dr = (bm2_dr-bm1_dr)/ntr_dr
bm_dr = round(bm1_dr:nr_esant_dr:bm2_dr)% minus un eşantion
bot_mute=[bm_stg(1:ntr_stg) bm_dr(1:ntr_dr)];
figure(2)
plot(bot_mute)
xlabel('Număr trase')
ylabel('Timp (ms)')

% aplic bottom-mute:
for it=1:nt
    for ix=1:nxg
        if it <= bot_mute(ix)
            St_bm(it,ix)=St(it,ix);
        else
            St_bm(it,ix)=0.0;
        end
```

```
    end
end
plotimage(St_bm(1:nt_zoom,:))
xlabel('Număr trase')
ylabel('Timp (ms)')
```

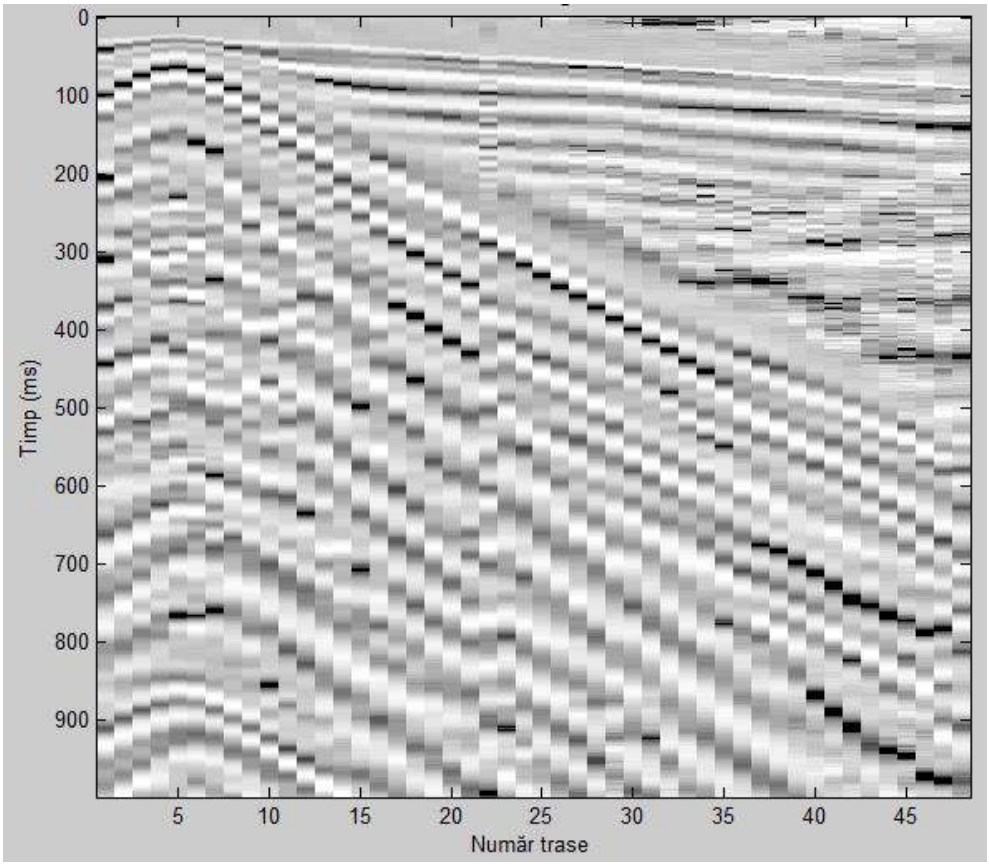

Figura 7.4: Înregistrare seismică înainte de aplicarea unui bottom-mute

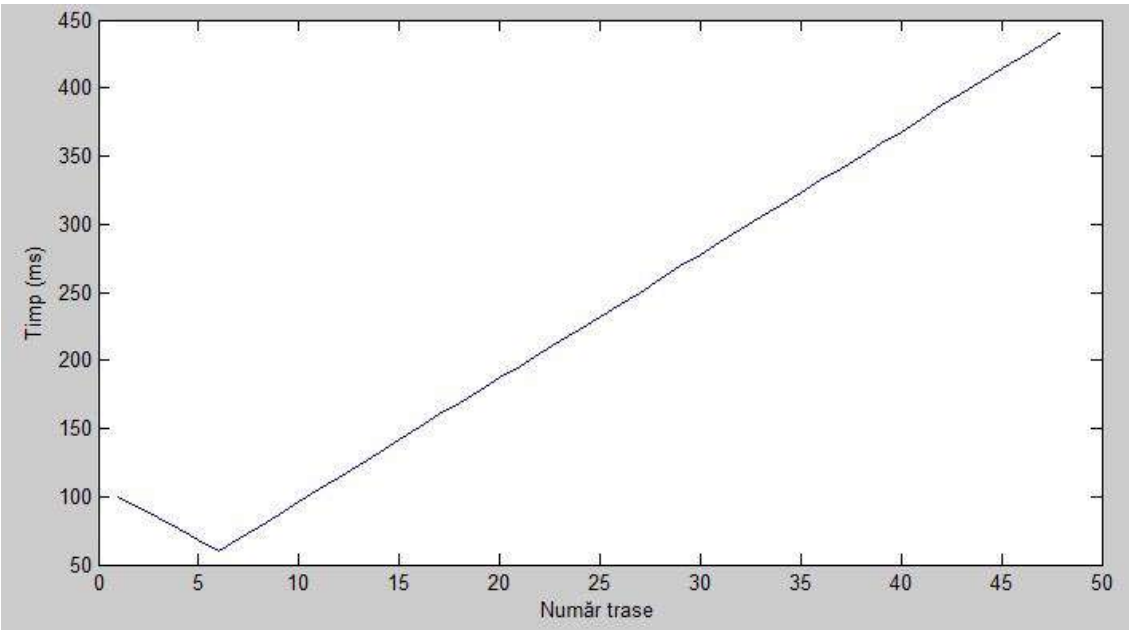

Figura 7.5: Vectorul folosit pentru definirea bottom-mute-ului

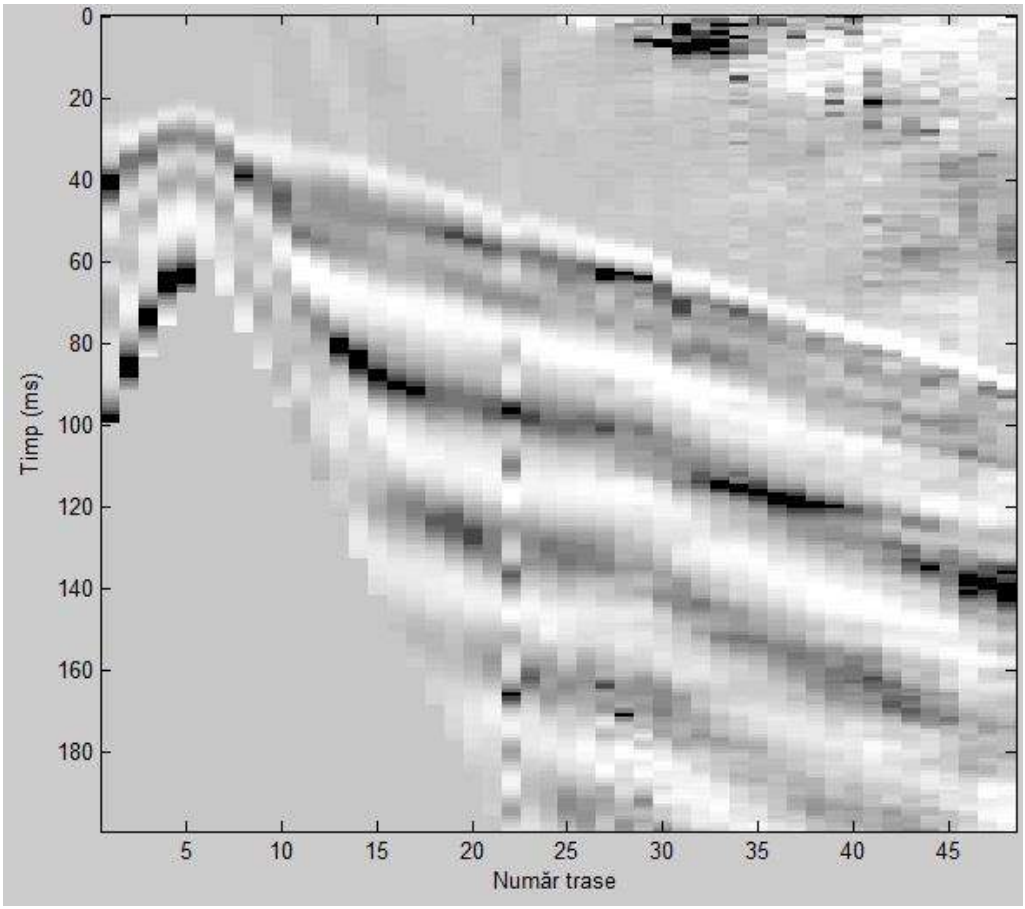

Figura 7.6: Înregistrare seismică după aplicarea unui bottom-mute

7.3 Definirea şi aplicarea unui surgical-mute

Surgical-mute este o combinaţie de top-mute şi bottom-mute. Aplicarea acestuia permite păstrarea semnalului pe un interval variabil în timp şi/sau spaţiu. Un exemplu de program folosit pentru definirea şi aplicarea unui surgical-mute este scris mai jos:

```
clear

% date de intrare
St0=segyread('D:\MATLAB\SH3_agc.sgy');
St=[St0(:,1:24) fliplr(St0(:,25:48))];
plotimage(St(1:1000,:))
xlabel('Număr trase')
ylabel('Timp (ms)')

% date de intrare
dxg = 2.5 % metri, distanţă geofoane
dt = 0.001 % interval eşantionare timp
nt_zoom = 1000

% definesc top-mute pentru dispozitiv cu generare în interiorul        %
întinderii de geofoane
% ramura din stânga
tm1_stg = 20
tm2_stg = 10
ntr_stg = 5

% ramura din dreapta
tm1_dr = 10
tm2_dr = 80
ntr_dr = 43

% definesc bottom-mute pentru dispozitiv cu generare în interiorul     %
întinderii de geofoane
% ramura din stânga
bm1_stg = 100
bm2_stg = 60
b_ntr_stg = 5

% ramura din dreapta
bm1_dr = 60
bm2_dr = 450
b_ntr_dr = 43

% NU modificaţi
size_St=size(St)
nt = max(size_St)
nxg = min(size_St)

nr_esant_stg = (tm1_stg-tm2_stg)/ntr_stg
tm_stg = round(tm1_stg:-nr_esant_stg:tm2_stg) % minus un eşantion
nr_esant_dr = (tm2_dr-tm1_dr)/ntr_dr
tm_dr = round(tm1_dr:nr_esant_dr:tm2_dr)% minus un eşantion
top_mute=[tm_stg(1:ntr_stg) tm_dr(1:ntr_dr)];
```

```
b_nr_esant_stg = (bm1_stg-bm2_stg)/b_ntr_stg
bm_stg = round(bm1_stg:-b_nr_esant_stg:bm2_stg) % minus un eşantion
b_nr_esant_dr = (bm2_dr-bm1_dr)/b_ntr_dr
bm_dr = round(bm1_dr:b_nr_esant_dr:bm2_dr)% minus un eşantion
bot_mute=[bm_stg(1:b_ntr_stg) bm_dr(1:b_ntr_dr)];

figure(2)
plot(top_mute)
hold on
plot(bot_mute,'r')
xlabel('Număr trase')
ylabel('Timp (ms)')

% aplic surgical-mute:
for it=1:nt
    for ix=1:nxg
        if it <= top_mute(ix)
            St_sm(it,ix)=0.0;
        elseif it >=bot_mute(ix)
            St_sm(it,ix)=0.0;
        else
            St_sm(it,ix)=St(it,ix);
        end
    end
end
plotimage(St_sm(1:nt_zoom,:))
xlabel('Număr trase')
ylabel('Timp (ms)')
```

Undele de suprafaţă şi zgomotul observat deasupra undelor frontale pe înregistrarea reprezentată în domeniul timp din Figura 7.7 sunt eliminate folosind un surgical-mute definit de timpii prezentaţi în Figura 7.8. Înregistrarea după aplicarea acestui filtru este prezentată în Figura 7.9.

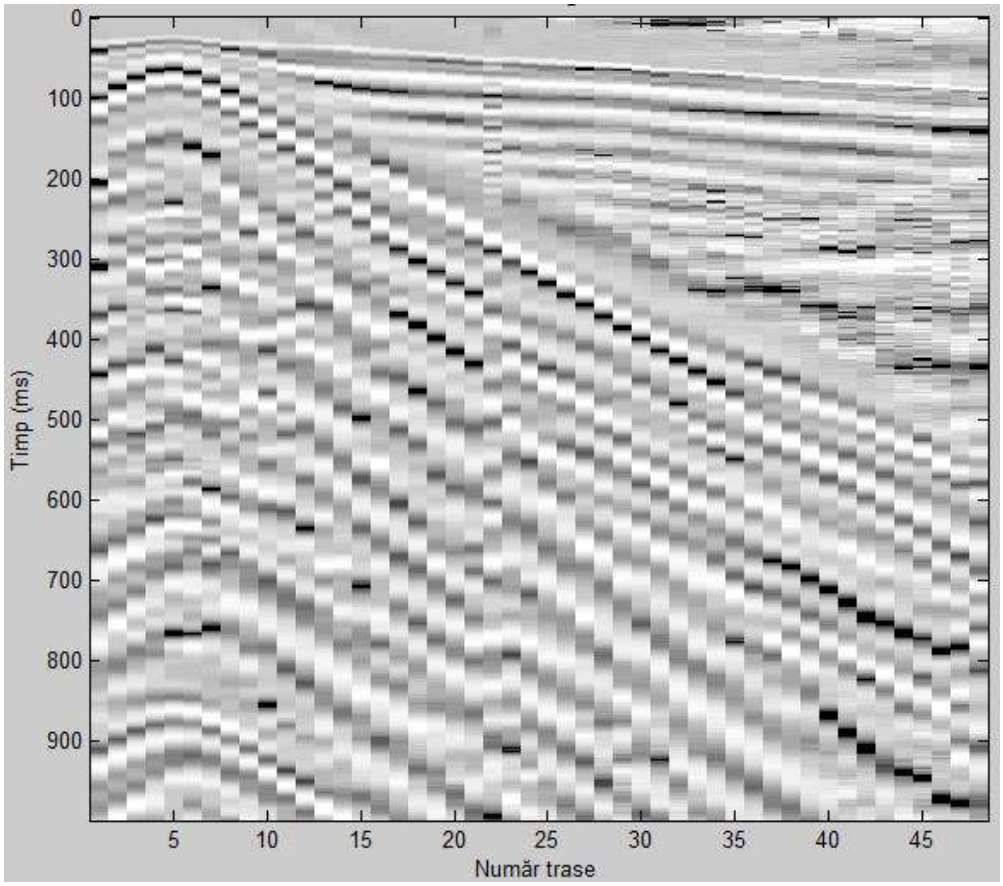

Figura 7.7: Înregistrare seismică înainte de aplicarea unui surgical-mute

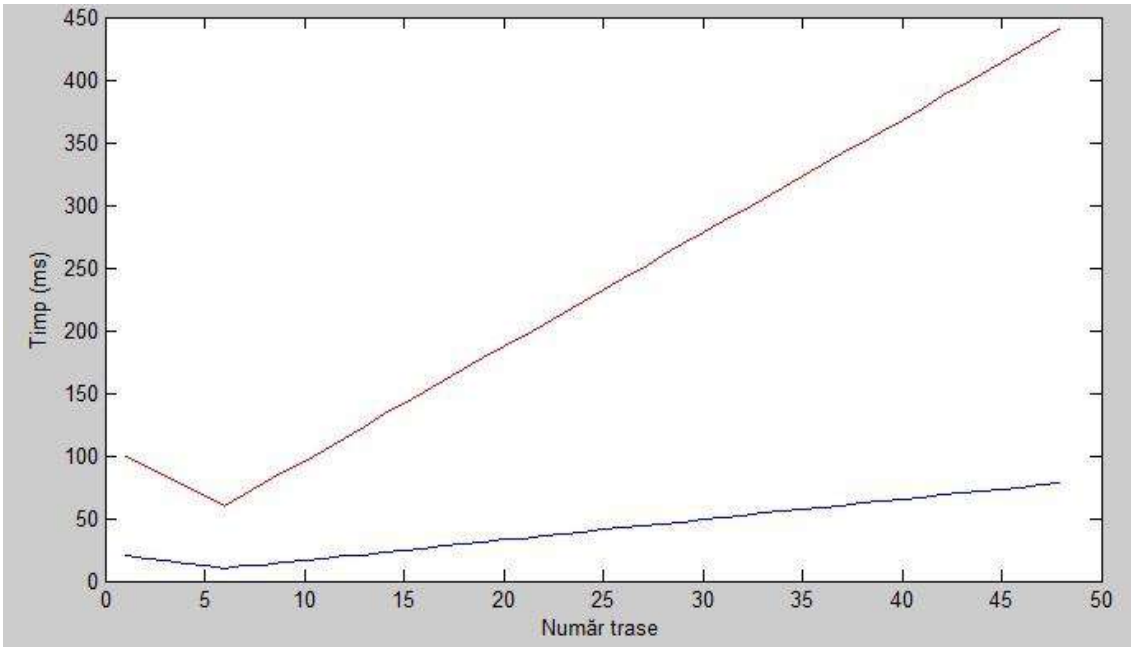

Figura 7.8: Vectorii folosiţi pentru definirea surgical-mute-ului

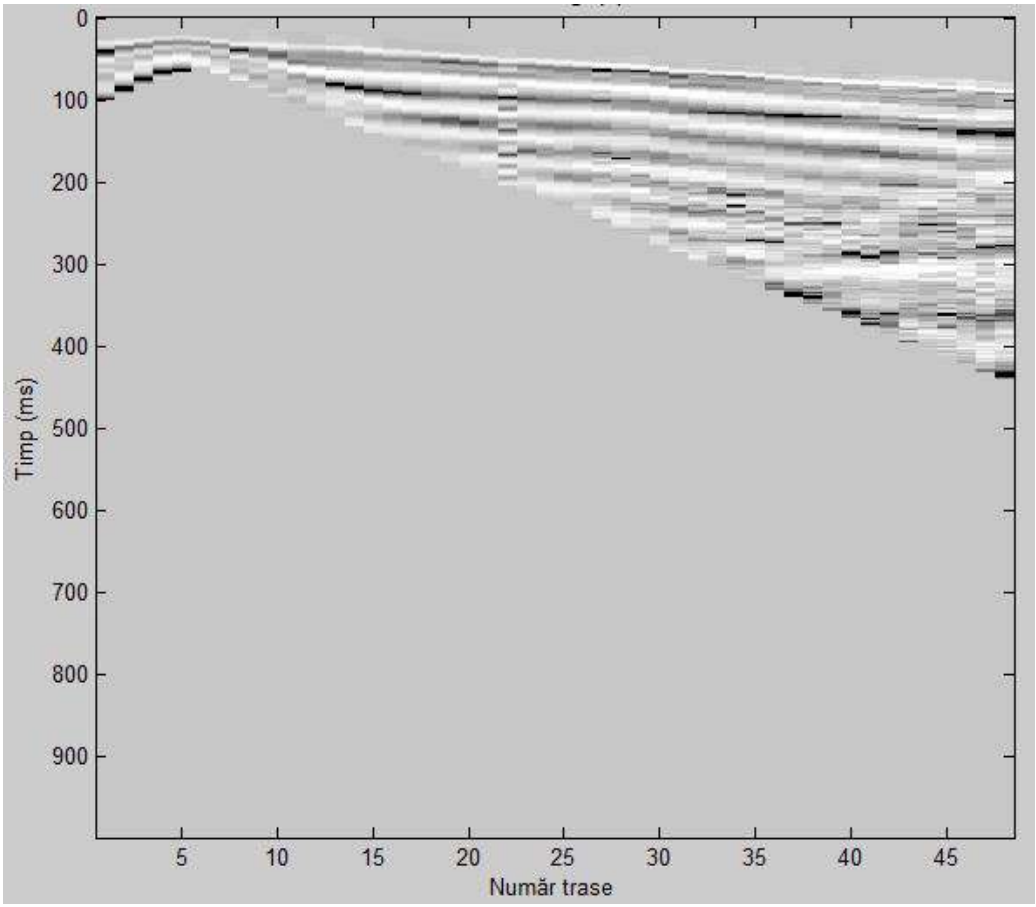

Figura 7.9: Înregistrare seismică după aplicarea unui surgical-mute

7.4 Definirea şi aplicarea grupărilor liniare de geofoane

Grupările de geofoane sunt folosite pentru atenuarea zgomotului coerent observat pe înregistrările seismice de reflexie (Levin, 1989). Răspunsul unei grupări de geofoane calculat din înregistrări cu geofoane individuale se obţine prin însumarea unui număr de trase egal cu numărul dorit de geofoane din grupare şi, apoi, reeşantionarea spaţială a rezultatului însumării la intervalul de grup (Panea and Drijkoningen, 2008). Intervalul de grup este distanţa dintre centrele a două grupări de geofoane alăturate.

Un exemplu de program folosit pentru a calcula răspunsul unei grupări liniare de geofoane este prezentat mai jos:

```
clear

St=segyread('D:\MATLAB\2025_5_agc.sgy');

% date de intrare
dxg = 5 % distanţă geofoane
dt = 0.001 % interval esantionare timp
nx1 = 5 % număr de geofoane din grupare
```

```
ng = 2 % interval de grup dxG = 2*dxg

plotimage(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')

% NU modificaţi

size_St=size(St)
nt = max(size_St)
nxg = min(size_St)

ax = 1:1:nx1
nxout = nxg - nx1
for ixout = 1:nxout
    out = (ixout - 1) + ax;
    Stw = St(:,out);
    Saw = zeros(nt,1);
    for ix=1:nx1
        Saw(:) = Saw(:) + Stw(:,ix);
    end
    Sawn = Saw/nx1;
    San(:,ixout) = Sawn;
end % ixout-loop

plotimage(San)
xlabel('Număr de trase')
ylabel('Timp (ms)')

% Răspuns grupare după reeşantionarea spaţială
St_grup = San(:,1:ng:nxout);
plotimage(St_grup)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

Salvarea rezultatelor în formatul SEG-Y se face folosind liniile de mai jos:

```
WriteSegy_dt1('D:\MATLAB\2025_5_San.sgy',San);
WriteSegy_dt1('D:\MATLAB\2025_5_add_2D.sgy',St_grup);
```

În Figura 7.10 este prezentată o înregistrare seismică obţinută folosind 100 geofoane individuale plasate la distanţe de 5 m. Intervalul de eşantionare în timp a fost de 0.001 s. Undele de suprafaţă care trebuie atenuate folosind grupările de geofoane sunt reprezentate de evenimentele liniare foarte înclinate observate pe seismograma din Figura 7.10. Răspunsul unei grupări de geofoane cu 5 elemente se obţine prin însumarea a câte 5 trase (1-5, 2-6, 3-7 etc) spaţiate la distanţe de 5 m şi, apoi, reeşantionarea răspunsului acestor însumări la distanţe de 10 m; această reeşantionare se obţine selectând fiecare a doua trasă din rezultatul însumării a câte 5 trase. Răspunul însumării de trase este prezentat în Figura 7.11 iar cel al reeşantionării de trase la 10 m în Figura 7.12.

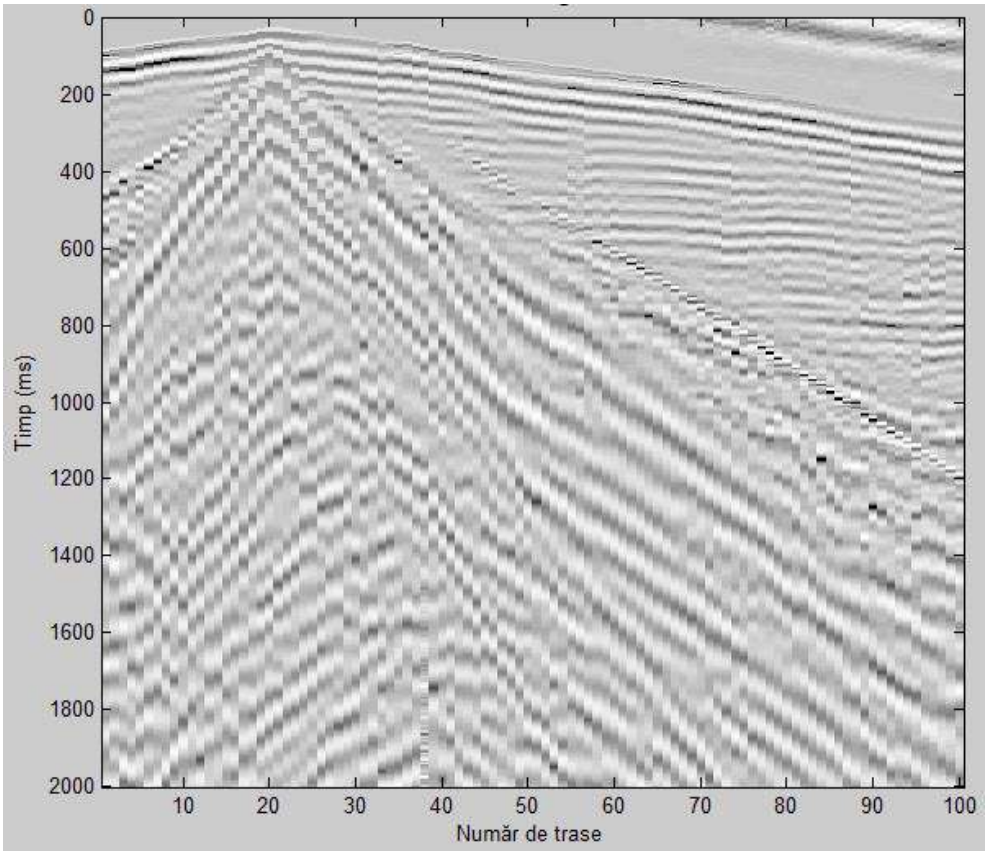

Figura 7.10: Înregistrare seismică obţinută din geofoane individuale

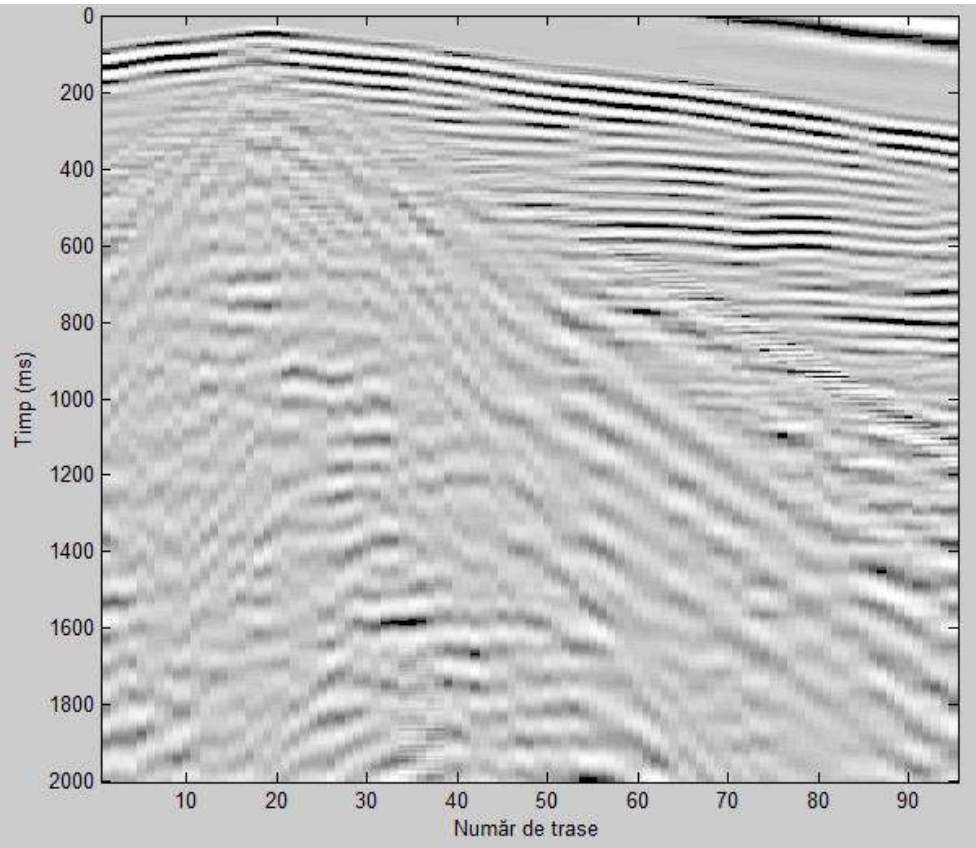

Figura 7.11: Răspunsul însumării a câte 5 trase

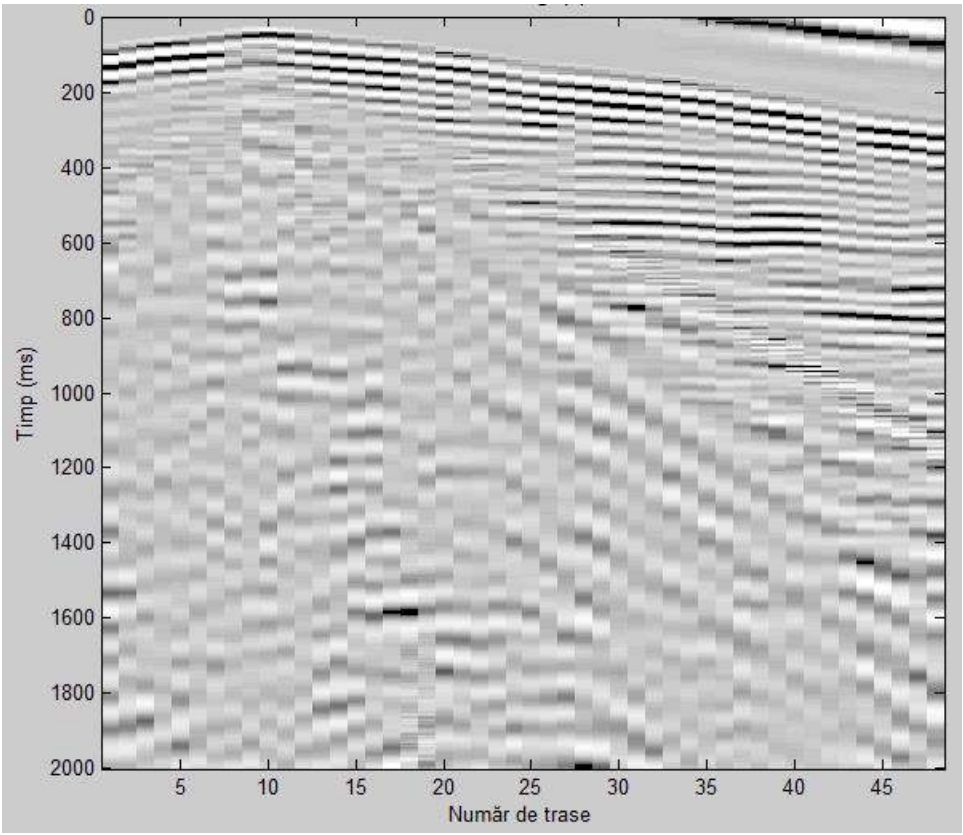

Figura 7.12: Răspunsul reeşantionării spaţiale la 10 m

Analiza atenuării undelor de suprafaţă se face comparând înregistrarea seismică înainte şi după aplicarea unei grupări de geofoane reprezentate în domeniul timp-spaţiu şi în domeniul frecvenţă-număr de undă. Spectrul de amplitudine (f, k) al unei înregistrări seismice se obţine aplicând TF 2D. Spectrul de amplitudine (f, k) al înregistrării din geofoane individuale este prezentat în Figura 7.13, cel al răspunsului însumării de trase este prezentat în Figura 7.14, iar spectrul de amplitudine (f, k) al răspunsului final al grupării de geofoane este prezentat în Figura 7.15.

Un exemplu de program folosit pentru calculul si reprezentarea acestor spectre de amplitudine este prezentat mai jos:

```
clear
% date de intrare
St = segyread('D:\MATLAB\2025_5_add_2D.sgy');

dxg = 10 % distanţe trase în St
dt = 0.001
f_plot = 155 % corespunde frecvenţei de 151 Hz
```

```
%%%%%%%%%%%%%%%% NU MODIFICAŢI %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nt = max(size(St)) % număr eşantioane timp
nxg = min(size(St)) % număr geofoane
nf = pow2(floor(log2(nt))+1) % număr eşantioane frecvenţă
df = 1/(nf*dt) % interval eşantionare frecvenţă
freq = df*(1:nf); % vector frecvenţă
nk = pow2(floor(log2(nxg))+1) % număr eşantioane număr de undă
dk = 1/(nk*dxg) % interval eşantionare număr de undă
k_no = dk*(1:nk); % vector număr de undă
kN = 1/(2*dxg) % număr de undă Nyquist
k_plot = k_no-kN % vector număr de undă pentru reprezentare

fft2_St = fft2(St,nf,nk);

abs_fft2_St=fliplr([abs(fft2_St(:,(nk/2)+1:nk))
abs(fft2_St(:,1:nk/2))]);
plotimage(abs_fft2_St(1:f_plot,:),freq(1:f_plot),k_plot)
xlabel('Număr de undă (1/m)')
ylabel('Frecvenţă (Hz)')
```

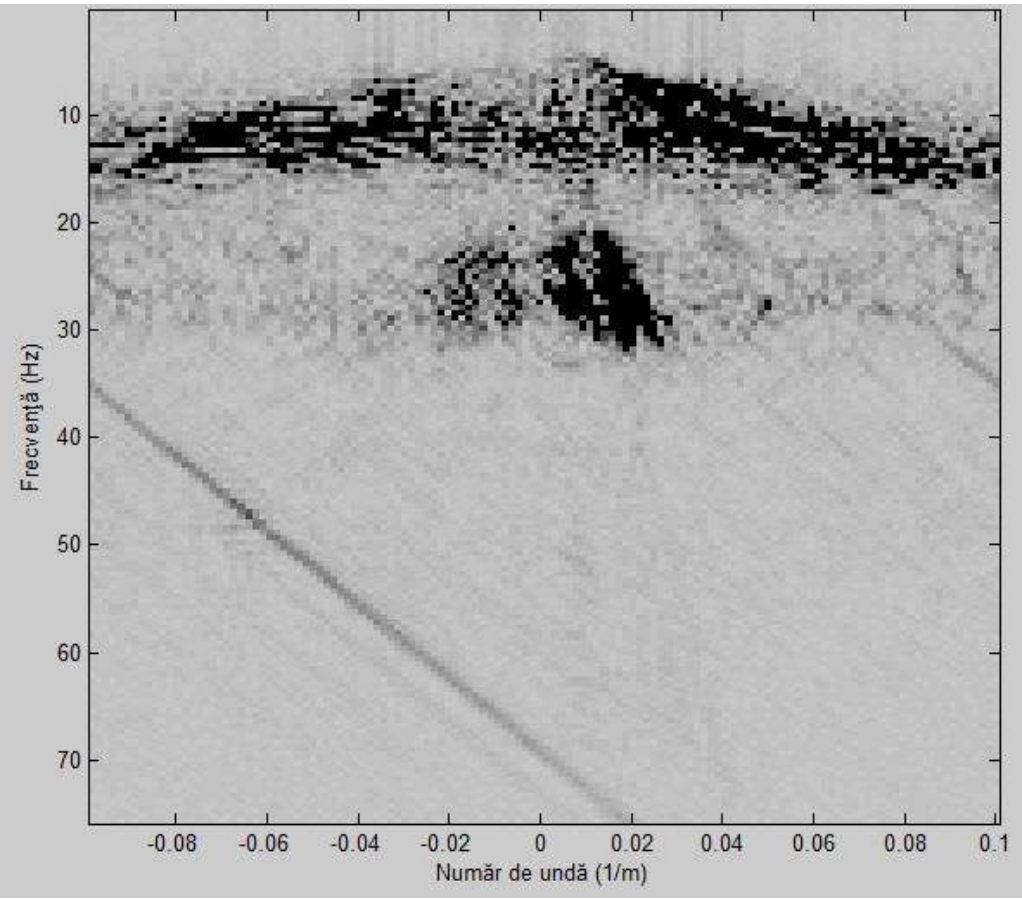

Figura 7.13: Spectrul de amplitudine (f, k) al înregistrării din Figura 7.10

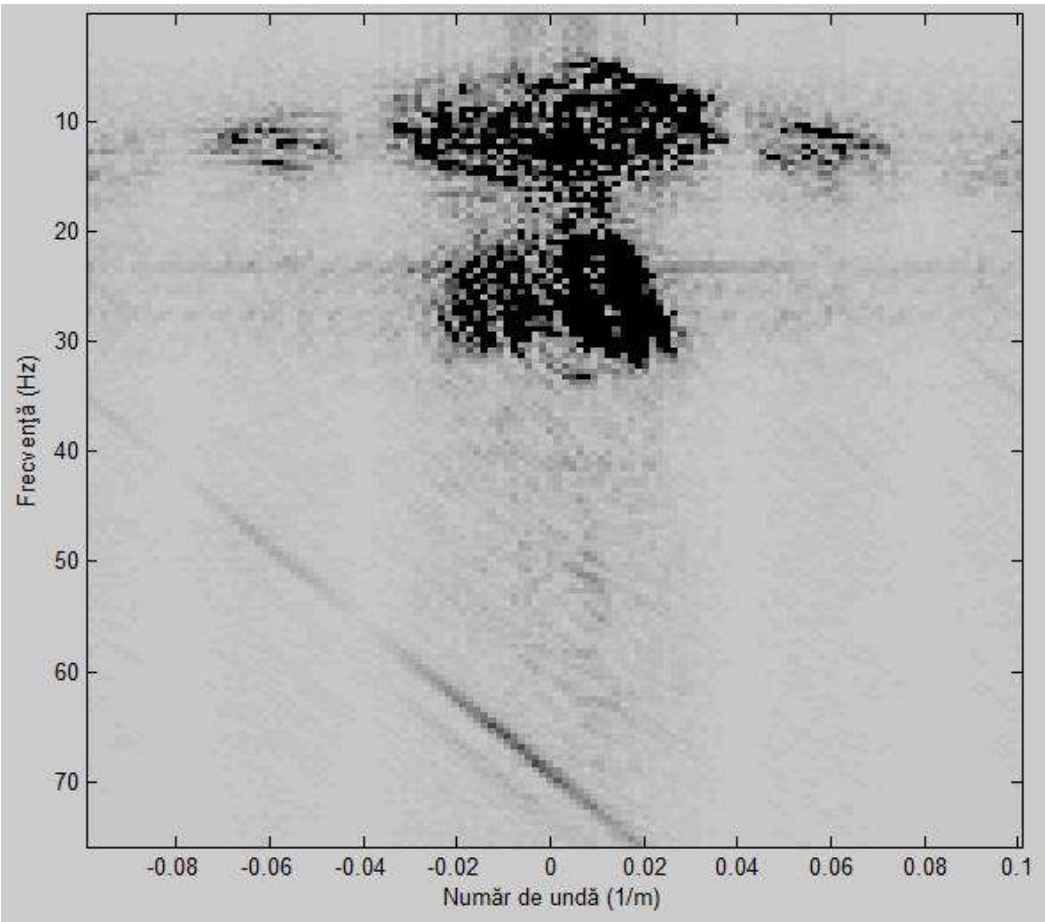

Figura 7.14: Spectrul de amplitudine (f, k) al înregistrării din Figura 7.11

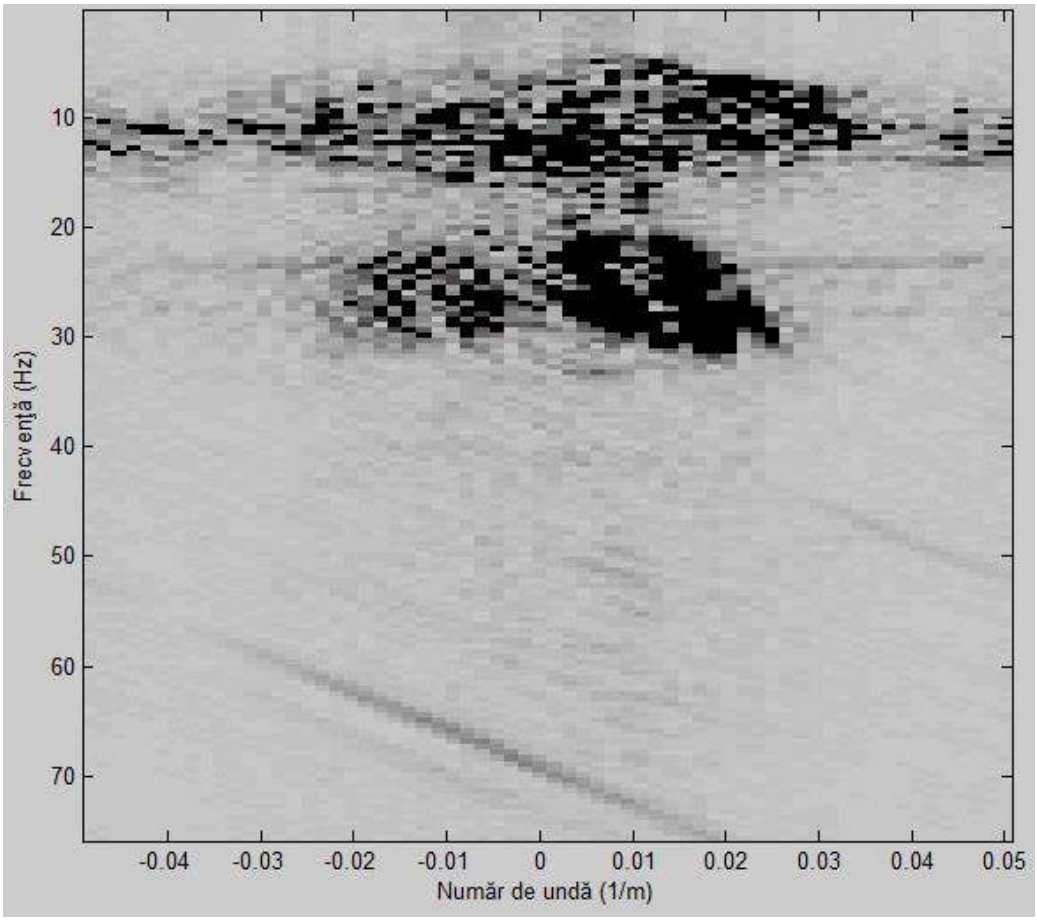

Figura 7.15: Spectrul de amplitudine (f, k) al înregistrării din Figura 7.12

7.5 Definirea şi aplicarea grupărilor de geofoane 3D

Grupările de geofoane 3D au diferite forme date de modul în care sunt aşezate geofoanele pe profilul seismic (Alsadi, 2016). În Figura 7.16 sunt prezentate patru tipuri de grupări de geofoane 3D, şi anume grupările rectangulare, circulare, cruce şi coadă de rândunică.

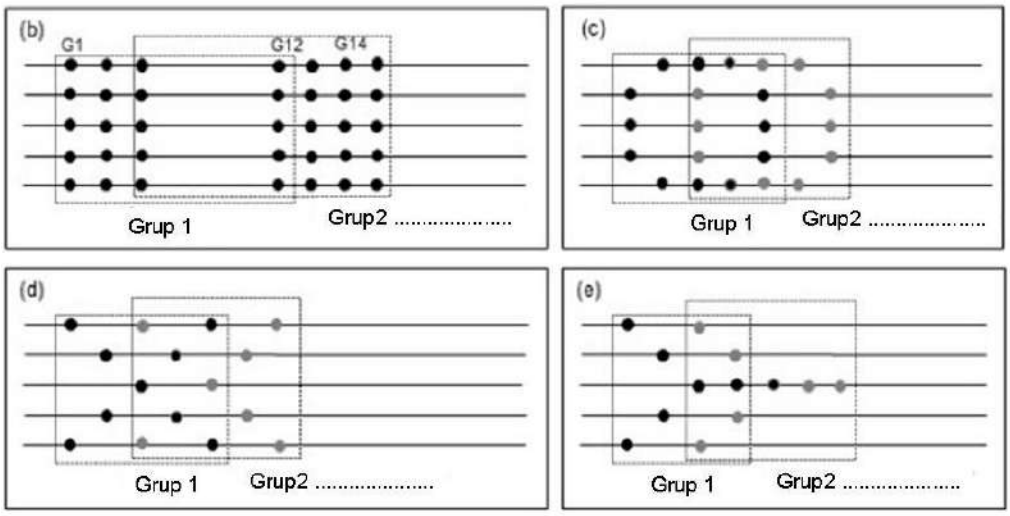

Figura 7.16: Tipuri de grupări de geofoane 3D; (a) rectangulară, (b) circulară, (c) cruce, (d) coadă de rândunică

Un exemplu de înregistrare seismică folosită în calculul răspunsului grupării de geofoane este prezentat în Figura 7.17 (seismograma notată cu variabila `St1` în programul de mai jos). Răspunsul grupării rectangulare se calculează folosind programul de mai jos. Răspunsul însumării de trase este prezentat în Figura 7.18, iar cel al reeşantionării la intervalul de grup este prezentat în Figura 7.19.

```
clear
% citim înregistrările cu corecţii de amplitudine aplicate:
St_1=segyread('D:\Matlab\2025_1_agc.sgy');
St_2=segyread('D:\Matlab\2025_2_agc.sgy');
St_3=segyread('D:\Matlab\2025_3_agc.sgy');
St_4=segyread('D:\Matlab\2025_4_agc.sgy');
St_5=segyread('D:\Matlab\2025_5_agc.sgy');

% selectăm geofoanele din partea dreaptă a punctului de împuşcare
St1=St_1(:,21:100);
St2=St_2(:,21:100);
St3=St_3(:,81:160);
St4=St_4(:,21:100);
St5=St_5(:,21:100);
plotimage(St1)
ylabel('Timp (ms)')
xlabel('Număr trase')

% grupare liniară de geofoane 3D, aplicată pe fiecare linie de geofoane:
nx1 = 12 % număr geofoane în gruparea liniară
nxg = min(size(St1))
nt = max(size(St1))
nxout = nxg - nx1
ax=1:nx1
```

```
dxG = 2 % interval de grup = 2*dxg
nx_cross = 5

% însumez crossline:
St = (St1+St2+St3+St4+St5)/nx_cross;

% calculez grupare inline după însumare crossline:
for ixout = 1:nxout
    out = (ixout - 1) + ax;
    St_w = St(:,out);
    St_aw = zeros(nt,1);
    for ix=1:nx1
        St_aw(:) = St_aw(:) + St_w(:,ix);
    end
    St_awn = St_aw/nx1;
    St_an(:,ixout) = St_awn;
end % ixout-loop
plotimage(St_an)
ylabel('Timp (ms)')
xlabel('Număr trase')

St_grup_3D = St_an(:,1:dxG:nxout);
figure(2)
plotimage(St_grup_3D)
ylabel('Timp (ms)')
xlabel('Număr trase')
```

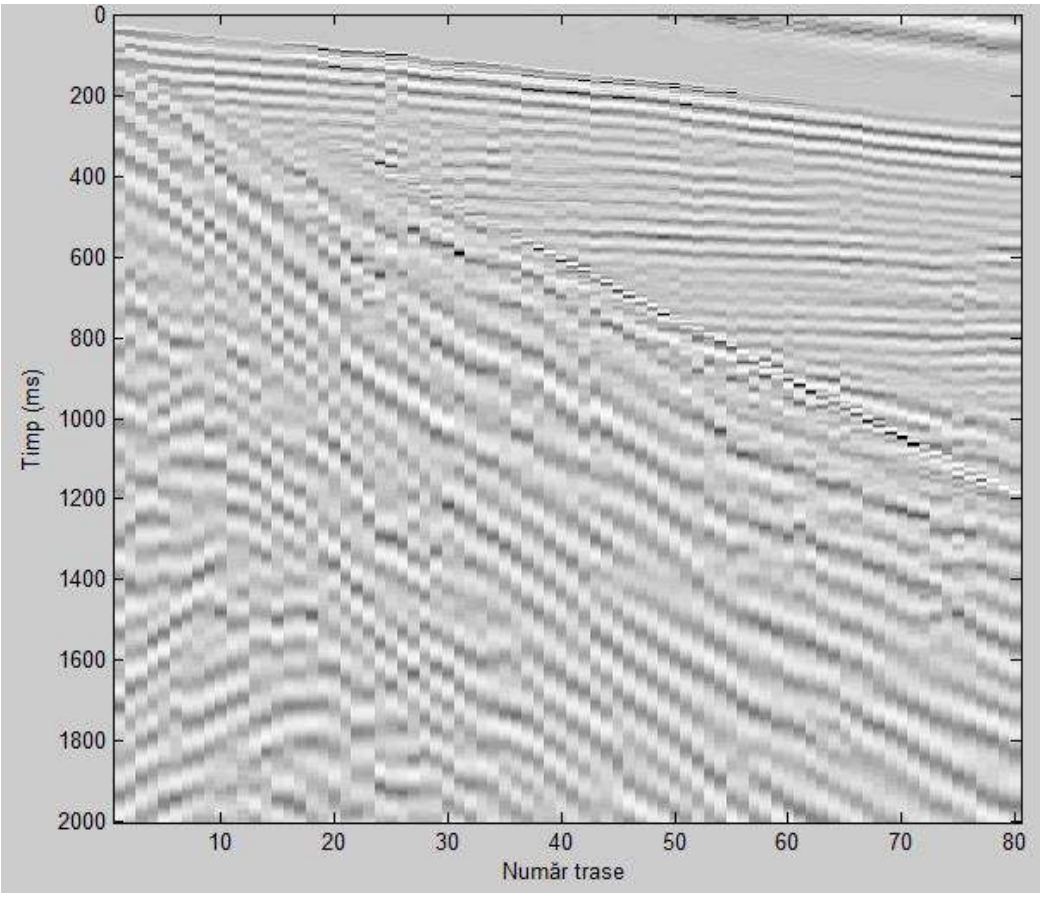

Figura 7.17: Înregistrare seismică obţinută din geofoane individuale

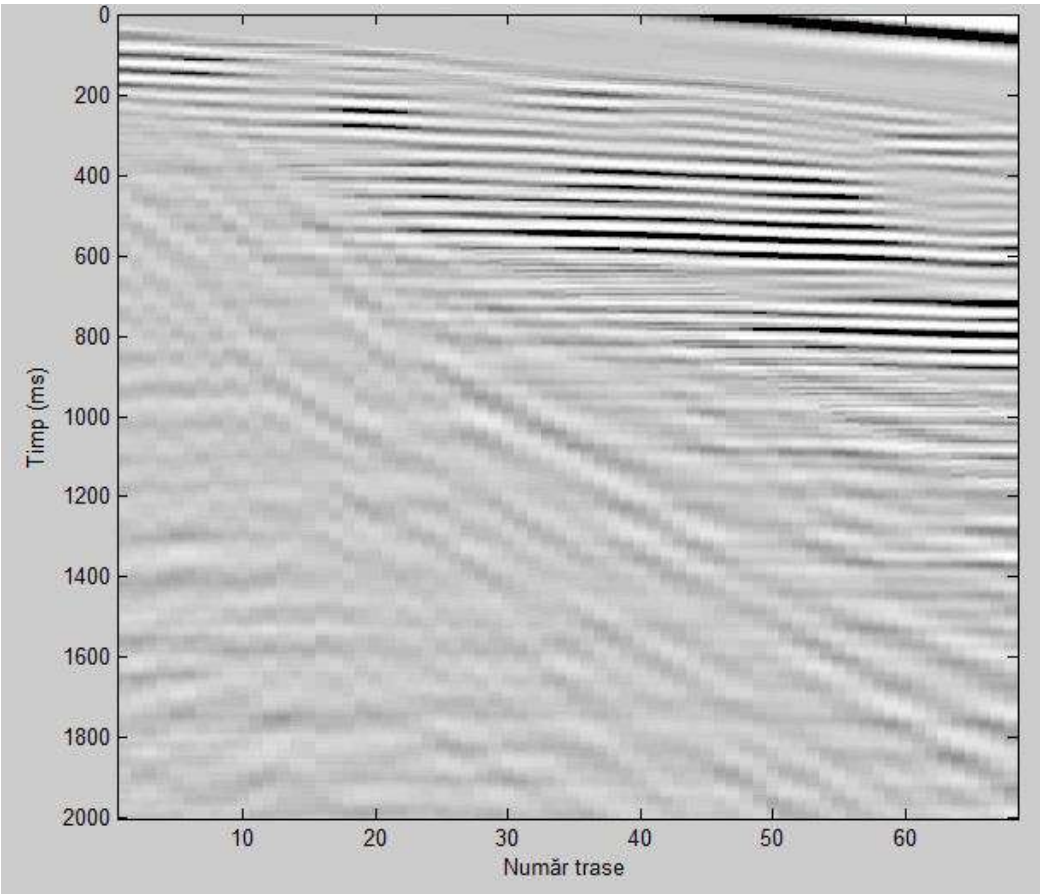

Figura 7.18: Răspunsul însumării a câte 12 × 5 trase

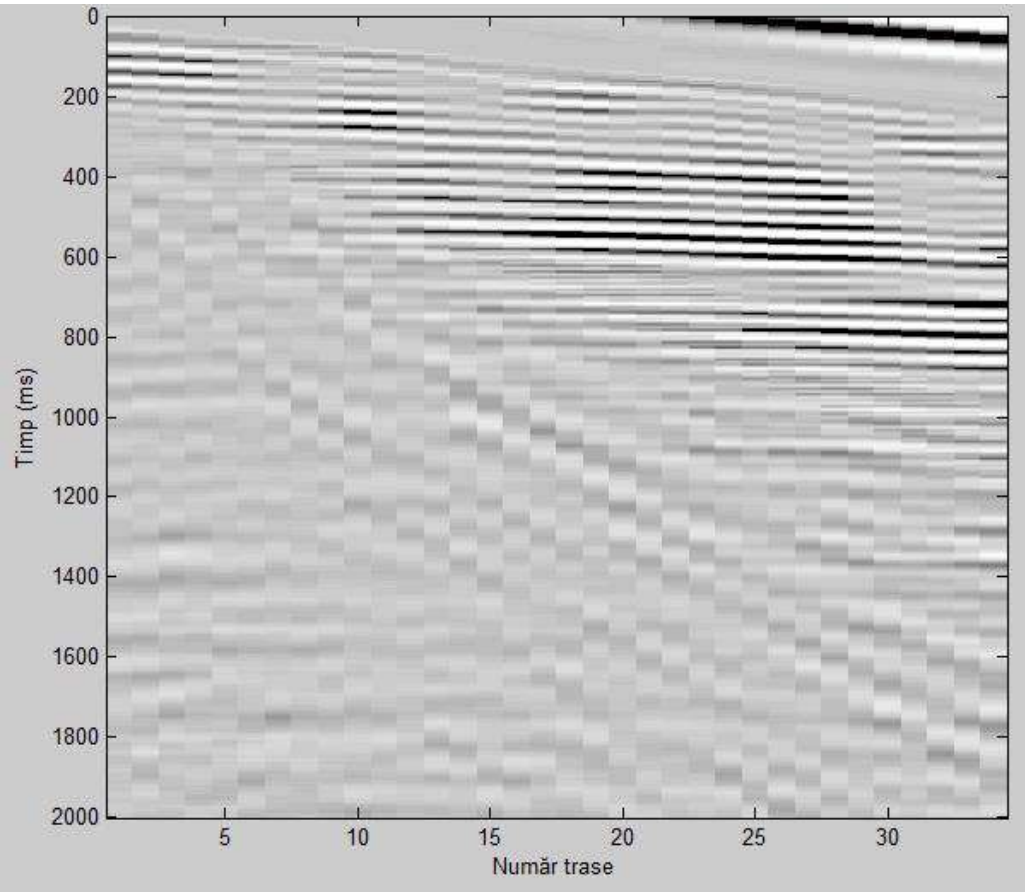

Figura 7.19: Răspunsul reeşantionării spaţiale la 10 m

Răspunsul grupării circulare se calculează folosind programul de mai jos:

```
clear
% citim înregistrările cu corecţii de amplitudine aplicate:
St_1=segyread('D:\Matlab\2025_1_agc.sgy');
St_2=segyread('D:\Matlab\2025_2_agc.sgy');
St_3=segyread('D:\Matlab\2025_3_agc.sgy');
St_4=segyread('D:\Matlab\2025_4_agc.sgy');
St_5=segyread('D:\Matlab\2025_5_agc.sgy');

% selectăm geofoanele din partea dreaptă a punctului de împuşcare
St1=St_1(:,21:100);
St2=St_2(:,21:100);
St3=St_3(:,81:160);
St4=St_4(:,21:100);
St5=St_5(:,21:100);
plotimage(St1)
ylabel('Timp (ms)')
xlabel('Număr trase')
```

```
% grupare de forma circulară:
nxg = min(size(St1))
nt = max(size(St1))
nx1 = 5 % maxim linii crossline
nxout = nxg-nx1
ax1 = [2 3 4]
ax2 = [1 5]
nx_norm=2*nx1+2
dxG = 2 % interval de grup = 2*dxg

T = zeros(nt,nx1-3);
T3 = zeros(nt,nx1-2);
for ixout = 1:nxout
    out1 = (ixout - 1) + ax1;
    out2 = (ixout - 1) + ax2;

    St1_w = [St1(:,out1) T];
    St5_w = [St5(:,out1) T];
    St2_w = [St2(:,out2) T3];
    St3_w = [St3(:,out2) T3];
    St4_w = [St4(:,out2) T3];

    St_add_w = St1_w+St2_w+St3_w+St4_w+St5_w;
    St_aw = zeros(nt,1);
    for ix=1:nx1
        St_aw(:)=St_aw(:)+St_add_w(:,ix);
    end
    St_awn = St_aw/nx_norm;
    St_an(:,ixout) = St_awn;
end % ixout-loop
plotimage(St_an)
ylabel('Timp (ms)')
xlabel('Număr trase')

St_grup_cerc = St_an(:,1:dxG:nxout);
plotimage(St_grup_cerc)
ylabel('Timp (ms)')
xlabel('Număr trase')
```

Seismograma folosită în această analiză este prezentată în Figura 7.20 (variabila `St1`). Răspunsul însumării a câte 12 trase corespunzătoare grupării circulare după reeşantionarea la 10 m este prezentat în Figura 7.21.

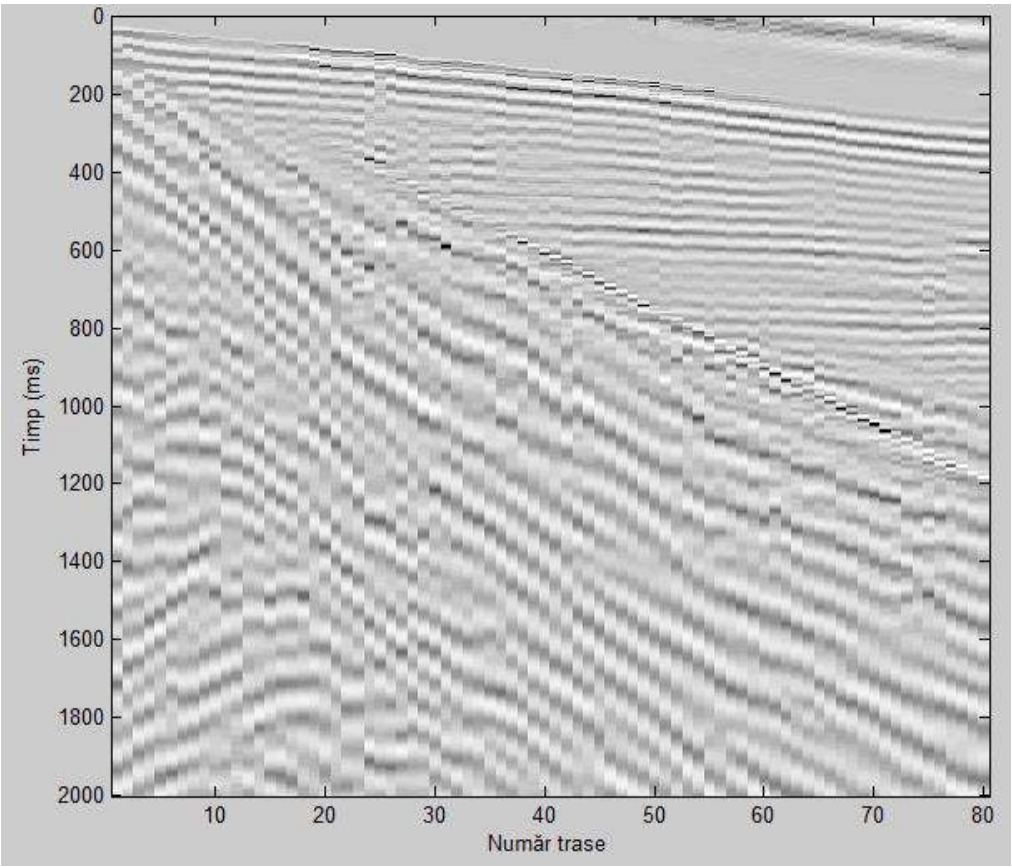

Figura 7.20: Răspunsul însumării a câte 12 trase care definesc o grupare circulară

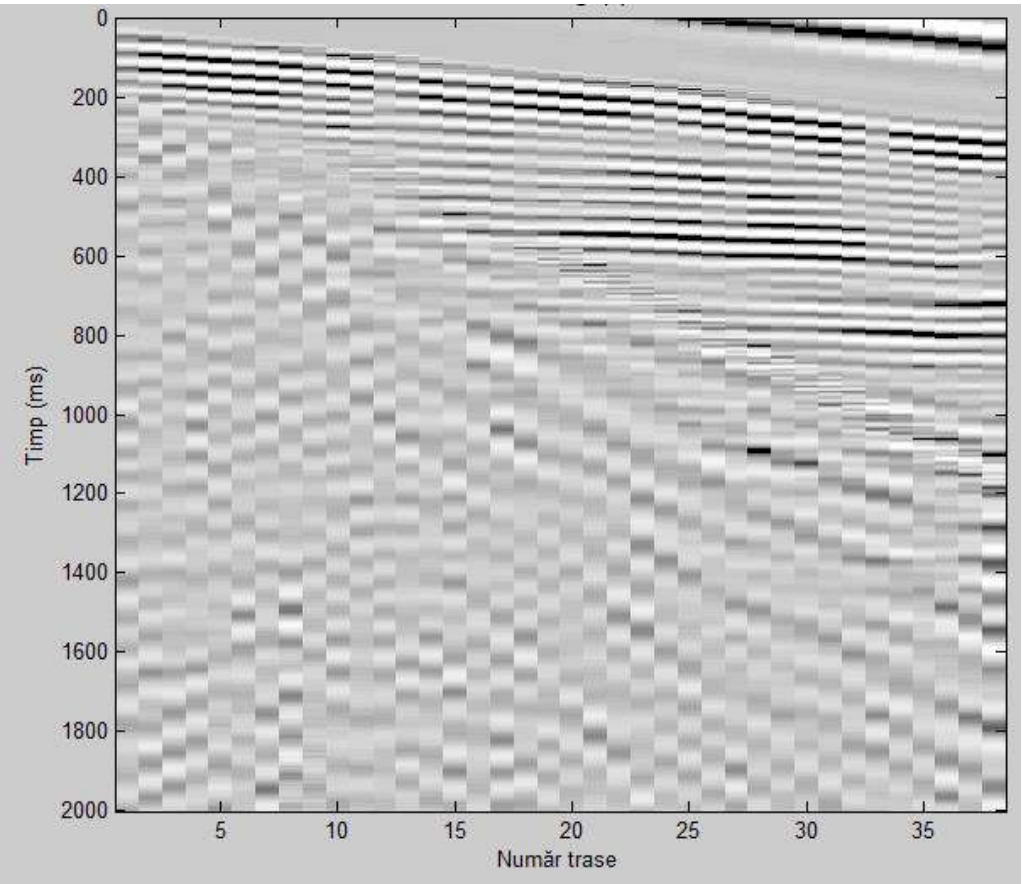

Figura 7.21: Răspunsul însumării de trase şi reeşantionării spaţiale la 10 m

7.6 Folosirea metodei "one-bit normalization"

Această metodă de normalizare este folosită în prelucrarea datelor seismice pasive, atunci când nu contează amplitudinile undelor seismice (Bensen et al., 2007). Amplitudinile cu valori negative sunt salvate ca fiind egale cu -1 iar cele cu valori pozitive sunt salvate egale cu 1. Programul descris mai jos poate fi folosit pentru aplicarea acestei metode.

```
clear

% date intrare
St=segyread('D:\Matlab\SH4_edit.sgy');

size_St = size(St)
nxg = min(size_St)
nt = max(size_St)

% normalizare un bit:
for it=1:nt
    for ix=1:nxg
```

```
        if St(it,ix)>0
            St_bit(it,ix)=1;
        else
            St_bit(it,ix)=-1;
        end
    end
end

% pentru verificare
St(100:105,20:25)
St_bit(100:105,20:25)

ans =
   -1.7552   -2.1927    0.3128   -1.3052   -0.7756   -1.2925
   -1.3367   -2.1937    0.6622   -1.6228   -1.1201   -0.6266
   -0.8044   -2.0947    0.7886   -1.8027   -1.4134   -0.1956
   -0.2198   -1.8918    0.6531   -1.8910   -1.7030   -0.1173
    0.3487   -1.5717    0.3389   -1.9655   -1.8114    0.1173
    0.8796   -1.1481   -0.1551   -1.9654   -1.7424    0.6252


ans =
    -1    -1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1
    -1    -1     1    -1    -1    -1
     1    -1     1    -1    -1     1
     1    -1    -1    -1    -1     1
```

7.7 Analiza de corelare

Analiza de corelare este folosită pentru a studia asemănarea dintre două semnale (Box and Jenkins, 1976). Funcţia de autocorelare analizează asemănarea dintre un semnal şi versiunea lui întârziată în timp. Funcţia de corelare încrucişată analizează asemănările dintre două semnale diferite. Ambele funcţii sunt folosite în prelucrarea datelor seismice înregistrate folosind metodele activă şi pasivă.

7.7.1 Funcţia de autocorelare

Ştiind o formă de undă, `A`, răspunsul autocorelării se poate calcula folosind programul de mai jos. Răspunsul este simetric faţă de întârzierea zero (Figura 7.22). Funcţia predefinită din Matlab `xcorr` aplicată pentru forma de undă `A` conduce la acelaşi răspuns. Funcţia de autocorelare se foloseşte în prelucrarea datelor seismice înregistrate în studiile seismice efectuate pentru explorarea pentru hidrocarburi şi în prelucrarea zgomotului ambiental înregistrat în studiile seismice pasive (Panea et al., 2014; Panea and Bugheanu, 2017).

```
clear

A = [0; 0; 3; 3; 3; 0; 0];
nt=max(size(A))

% întârzieri pozitive (ilag=1 înseamnă întirziere zero):
for ilag=1:nt
    out1=ilag:nt;
    out2=1:nt-ilag+1;
    fac_pos(ilag,1) = (A(out1,1))'*A(out2,1);
end

% întârzieri negative:
for ilag=1:nt
    out1=1:nt-ilag+1;
    out2=ilag:nt;
    fac_neg0(ilag,1)=(A(out1,1))'*A(out2,1);
end
fac_neg0;
fac_neg=flipud(fac_neg0(2:nt,1));

% Autocorelare:
AC = flipud([fac_neg; fac_pos])
plot(AC)
axis([0 15 -2 35])
xlabel('Eşantioane de timp')
ylabel('Amplitudine')
```

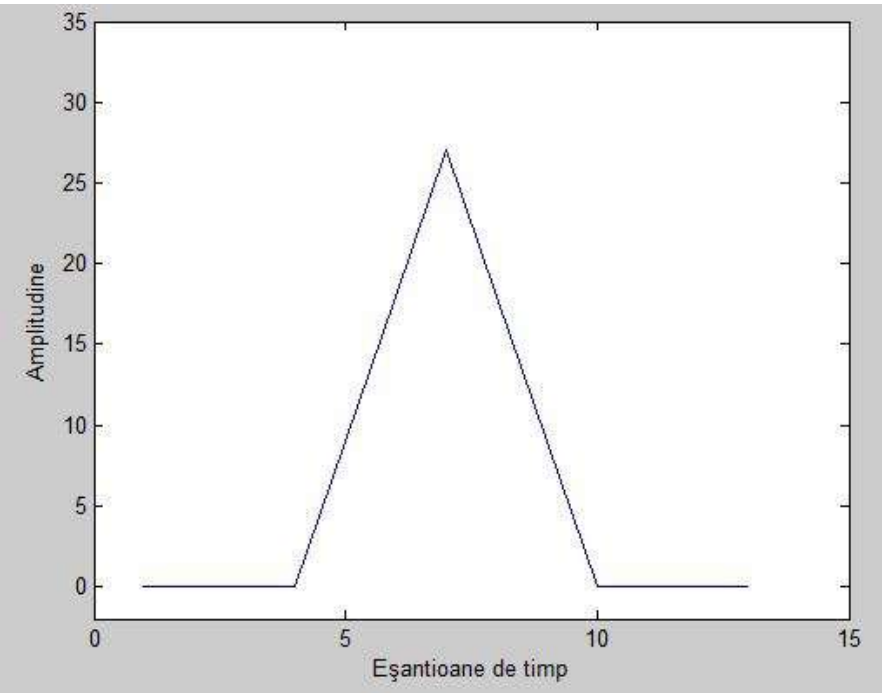

Figura 7.22: Răspunsul autocorelării pentru forma de undă `A`

În Figura 7.23 sunt prezentate patru trase seismice selectate dintr-un panou de zgomot ambiental. Răspunsul autocorelării este prezentat în Figura 7.24. După autocorelare, lungimea în timp a trasei se dublează.

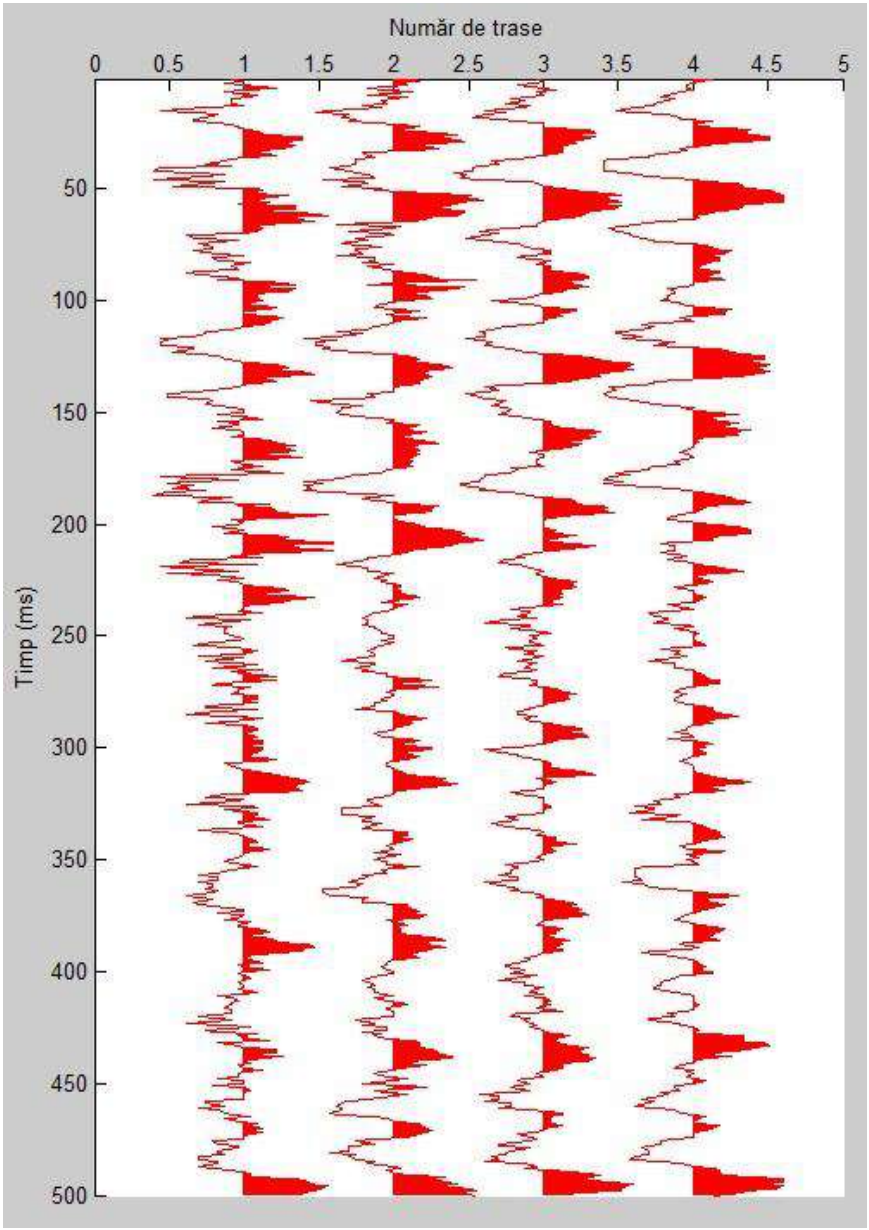

Figura 7.23: Trase seismice înainte de autocorelare

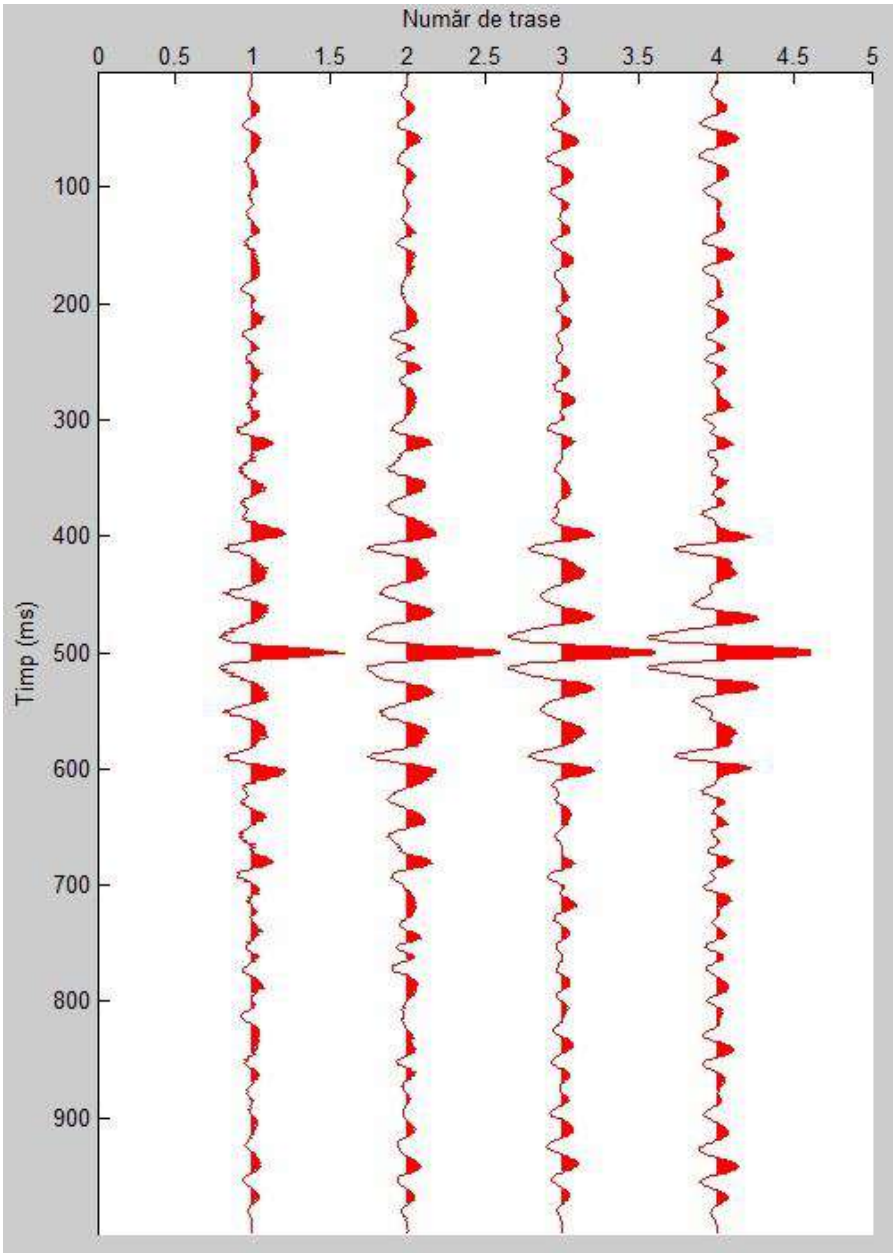

Figura 7.24: Trase seismice după autocorelare

7.7.2 Funcţia de corelare încrucişată

Ştiind formele de undă `A` şi `B`, răspunsul corelării încrucişate se poate calcula folosind programul de mai jos (Figura 7.25):

```
clear

A = [0; 0; 3; 3; 3; 0; 0];
```

```
B = [0; 0; 3; 2; 1; 0; 0];
nt=max(size(A))

% întârzieri pozitive (ilag=1 înseamnă întârziere zero):
for ilag=1:nt
    out1=ilag:nt;
    out2=1:nt-ilag+1;
    fac_pos(ilag,1) = (A(out1,1))'*B(out2,1);
end

% întârzieri negative:
for ilag=1:nt
    out1=1:nt-ilag+1;
    out2=ilag:nt;
    fac_neg0(ilag,1)=(A(out1,1))'*B(out2,1);
end
fac_neg0;
fac_neg=flipud(fac_neg0(2:nt,1));

% Cross_correlation:
CC = flipud([fac_neg; fac_pos])
plot(CC)
axis([0 20 -2 20])
xlabel('Eşantioane de timp')
ylabel('Amplitudine')
```

În Figura 7.26 sunt prezentate patru trase seismice selectate dintr-un panou de zgomot ambiental. Răspunsul autocorelării pentru trasa cu numărul 1 din panoul de zgomot este reprezentat de prima trasă din Figura 7.27. Restul traselor reprezintă răspunsurile corelărilor încrucişate între prima trasă din panoul de zgomot şi restul de trei trase din panou. După corelare, lungimea în timp a trasei se dublează.

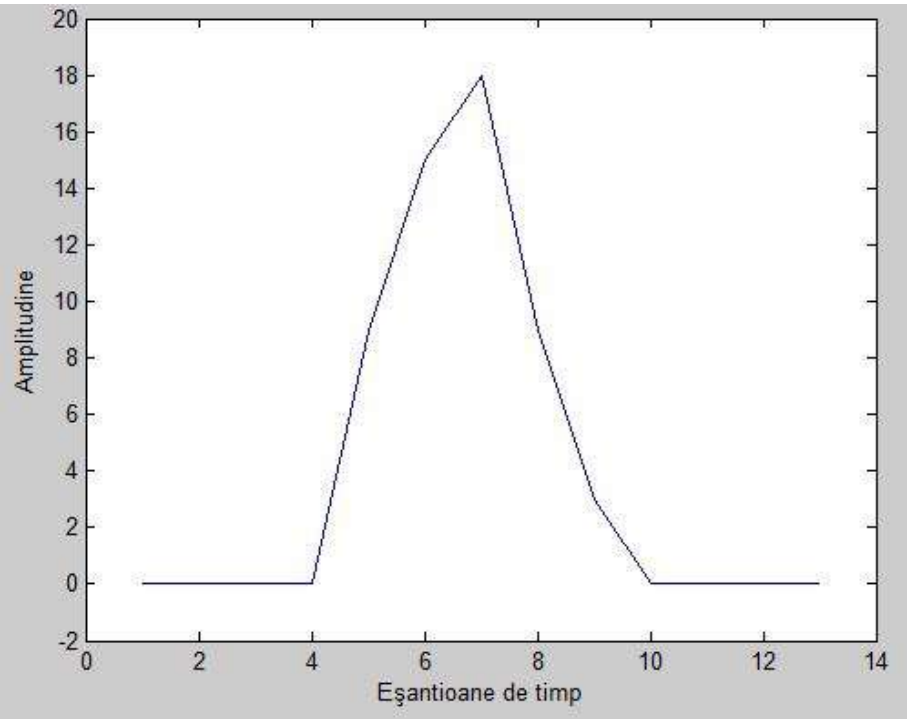

Figura 7.25: Răspunsul corelării încrucişate dintre formele de undă `A` şi `B`

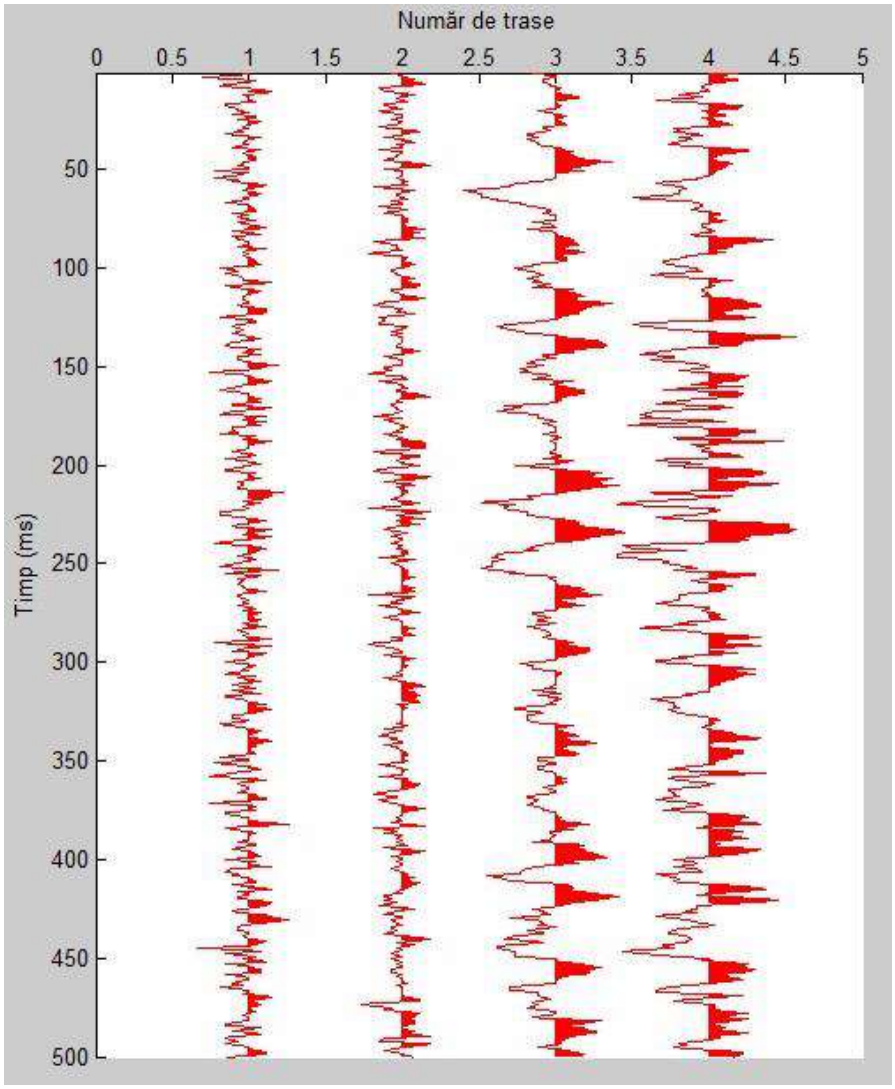

Figura 7.26: Trase seismice înainte de corelarea încrucişată

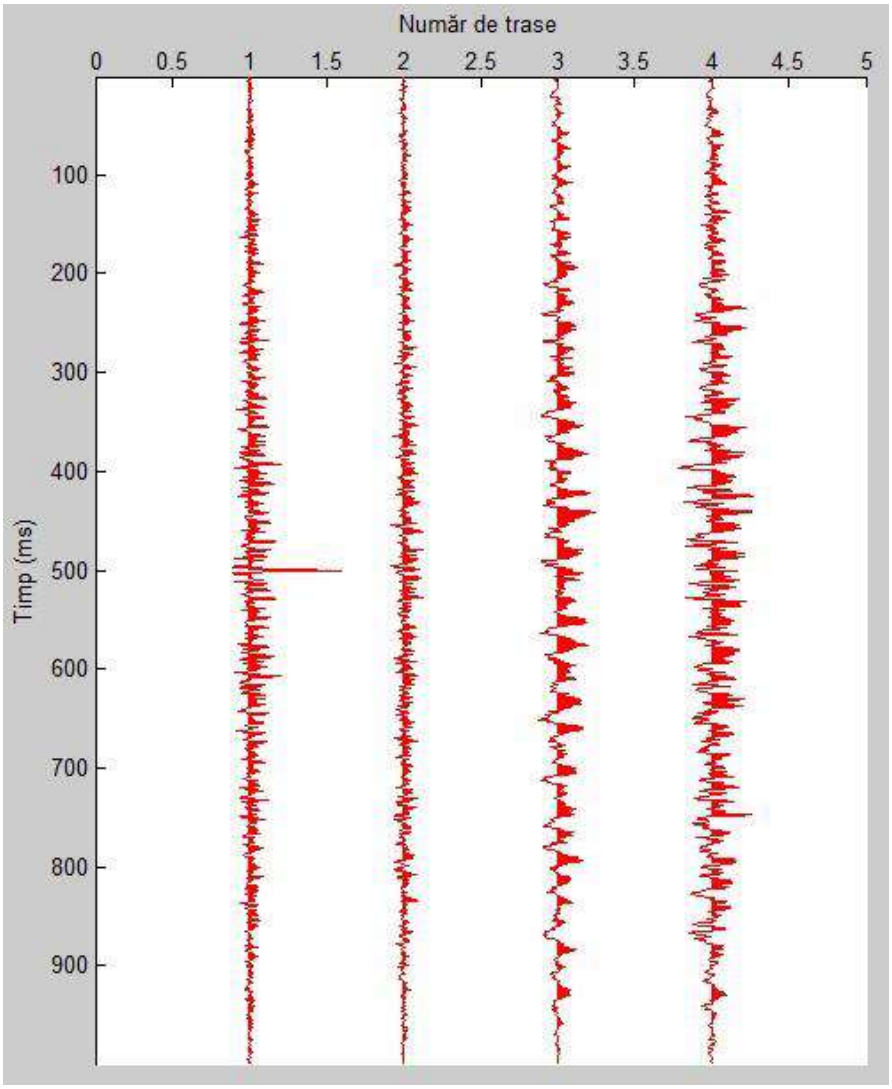

Figura 7.27: Trase seismice după corelarea încrucişată

Exerciţii fără rezolvare afişată:

E1: Reprezentaţi în domeniul (f, k) înregistrările SH6_agc.sgy şi SH7_agc.sgy înainte şi după eliminarea undelor de suprafaţă prin bottom- şi surgical-mute.

E2: Calculaţi şi reprezentaţi spectrele de amplitudine (f, k) ale răspunsurilor însumării de trase într-o grupare circulară şi reeşantionării spaţiale la 10 m.

E3: Definiţi şi reprezentaţi în domeniile (t, x) şi (f, k) răspunsurile însumării de trase din grupările de geofoane în cruce şi coadă de rândunică.

E4: Definiţi şi reprezentaţi în domeniile (t, x) şi (f, k) răspunsurile reşantionării spaţiale la 10 m din grupările de geofoane în cruce şi coadă de rândunică.

E5: Calculaţi şi reprezentaţi în domeniul timp răspunsurile autocorelării şi corelării încrucişate pentru trasele conţinute de panoul de zgomot shot_1.su. Analizaţi răspunsurile.

Capitolul 8
Modelarea undelor seismice

8.1 Modelare simplă (fără formă de undă Ricker)

O seismogramă sintetică se poate construi ştiind vitezele de propagare ale undelor seismice modelate, grosimile stratelor conţinute de modelul geologic şi paranetrii dispozitivului seismic (număr de geofoane, distanţa dintre aceştia, număr eşantioane de timp, valoarea offset-ului pentru un dispozitiv de împuşcare cu offset).

Programul de mai jos poate fi folosit pentru obţinerea seismogramei sintetice prezentate în Figura 8.1.

```
clear

% date model geologic
nxg = 200 % număr geofoane
dxg = 2 % metri, distanţă geofoane
h1 = 20 % metri, grosime strat 1
h2 = 35 % metri, grosime strat 2
V1 = 950 % m/s, viteză strat 1
V2 = 1200 % m/s, viteză strat 2
V3 = 1800 % m/s, viteză strat 3
V = 250 % m/s, viteză undă de suprafaţă
offset = 50 % metri

% formule
Vrms = round(sqrt((h1*V1+h2*V2)/((h1/V1)+(h2/V2))))
sin_icr1=V1/V2
cos_icr1=sqrt(1-sin_icr1^2)
tg_icr1=sin_icr1/cos_icr1

sin_icr2=V2/V3
cos_icr2=sqrt(1-sin_icr2^2)
tg_icr2=sin_icr2/cos_icr2
sin_i1=V1/V3
cos_i1=sqrt(1-sin_i1^2)
tg_i1=sin_i1/cos_i1
x_UF1=round((2*h1*tg_icr1)/dxg)
x_UF2=round((2*h1*tg_i1+2*h2*tg_icr2)/dxg)

for ix=1:nxg
    %x(ix)=(ix-1)*dxg; % poziţie geofoane împuşcare capăt
    %x(ix)=offset+(ix-1)*dxg; % pozitie geofoane împuşcare offset
    x(ix)=(ix-1)*dxg-((nxg-1)*dxg)/2; % poziţie geofoane split-spread
    t(ix)=abs(x(ix))/V;
    t_USR1(ix)=(1/V1)*sqrt(x(ix)^2+4*h1^2);
    t_UF1(ix)=(2*h1)/(cos_icr1*V1)+(abs(x(ix))-2*h1*tg_icr1)/V2;
    t_USR2(ix)=(1/Vrms)*sqrt(x(ix)^2+4*(h1+h2)^2);
```

```
    t_UF2(ix)=(2*h1)/(cos_i1*V1)+(2*h2)/(cos_icr2*V2)+(abs(x(ix))-
(2*h1*tg_i1+2*h2*tg_icr2))/V3;
end

plot(x,t)
hold on
plot(x,t_USR1,'r')
plot(x(1:round(nxg/2)-x_UF1),t_UF1(1:round(nxg/2)-x_UF1),'g')
plot(x(round(nxg/2)+x_UF1:nxg),t_UF1(round(nxg/2)+x_UF1:nxg),'g')
plot(x,t_USR2,'k')
plot(x(1:round(nxg/2)-x_UF2),t_UF2(1:round(nxg/2)-x_UF2),'c')
plot(x(round(nxg/2)+x_UF2:nxg),t_UF2(round(nxg/2)+x_UF2:nxg),'c')
axis([-200 200 0 0.5])
xlabel('pozitie geofoane (m)')
ylabel('timp (s)')
```

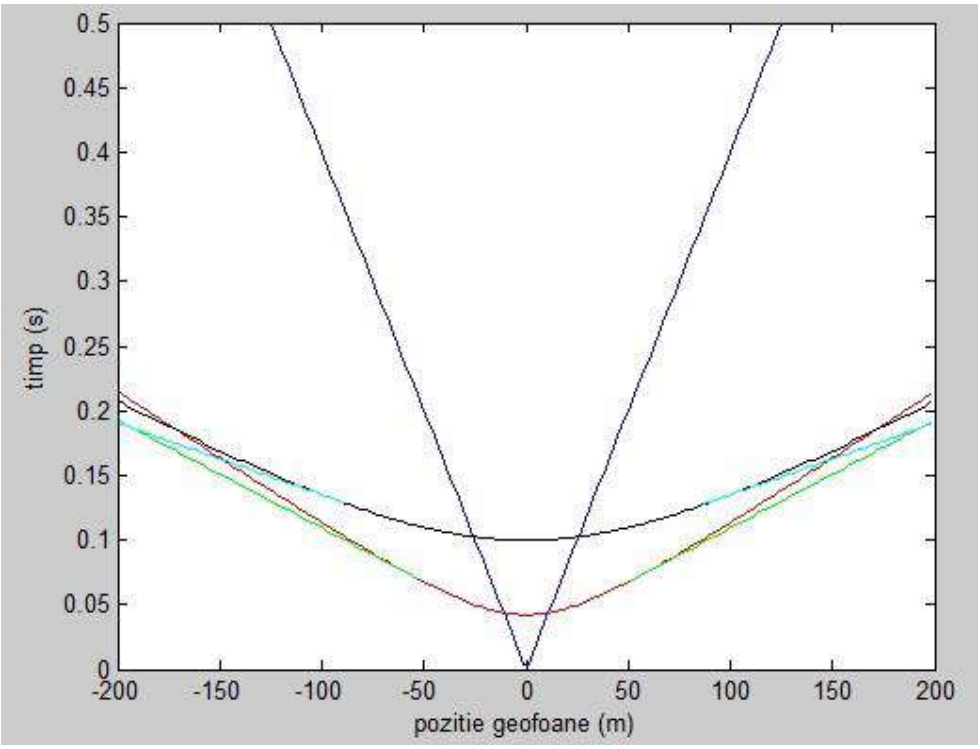

Figura 8.1: Reprezentarea timpilor de propagare calculați pentru o undă de suprafață, două unde reflectate şi două unde frontale. Dispozitiv seismic – împuşcare la centru

Programul de mai jos poate fi folosit pentru obţinerea seismogramei sintetice prezentate în Figura 8.2.

```
clear

% date model geologic
nxg = 200 % număr geofoane
dxg = 2 % metri, distanţă geofoane
```

```
h1 = 20 % metri, grosime strat 1
h2 = 35 % metri, grosime strat 2
V1 = 950 % m/s, viteză strat 1
V2 = 1200 % m/s, viteză strat 2
V3 = 1800 % m/s, viteză strat 3
V = 250 % m/s, viteză undă de suprafață
offset = 50 % metri

% formule
Vrms = round(sqrt((h1*V1+h2*V2)/((h1/V1)+(h2/V2))))
sin_icr1=V1/V2
cos_icr1=sqrt(1-sin_icr1^2)
tg_icr1=sin_icr1/cos_icr1

sin_icr2=V2/V3
cos_icr2=sqrt(1-sin_icr2^2)
tg_icr2=sin_icr2/cos_icr2
sin_i1=V1/V3
cos_i1=sqrt(1-sin_i1^2)
tg_i1=sin_i1/cos_i1
x_UF1=round((2*h1*tg_icr1)/dxg)
x_UF2=round((2*h1*tg_i1+2*h2*tg_icr2)/dxg)

for ix=1:nxg
    %x(ix)=(ix-1)*dxg; % poziție geofoane împușcare capăt
    x(ix)=offset+(ix-1)*dxg; % poziție geofoane împușcare offset
    %x(ix)=(ix-1)*dxg-((nxg-1)*dxg)/2; % poziție geofoane split-spread
    t(ix)=abs(x(ix))/V;
    t_USR1(ix)=(1/V1)*sqrt(x(ix)^2+4*h1^2);
    t_UF1(ix)=(2*h1)/(cos_icr1*V1)+(abs(x(ix))-2*h1*tg_icr1)/V2;
    t_USR2(ix)=(1/Vrms)*sqrt(x(ix)^2+4*(h1+h2)^2);
    t_UF2(ix)=(2*h1)/(cos_i1*V1)+(2*h2)/(cos_icr2*V2)+(abs(x(ix))-
(2*h1*tg_i1+2*h2*tg_icr2))/V3;
end

% offset
plot(x,t)
hold on
plot(x,t_USR1,'r')
plot(x,t_USR2,'k')
plot(x(x_UF1:nxg),t_UF1(x_UF1:nxg),'g')
plot(x(x_UF2:nxg),t_UF2(x_UF2:nxg),'c')
xlabel('pozitie geofoane (m)')
ylabel('timp (s)')
axis([50 450 0 1])
```

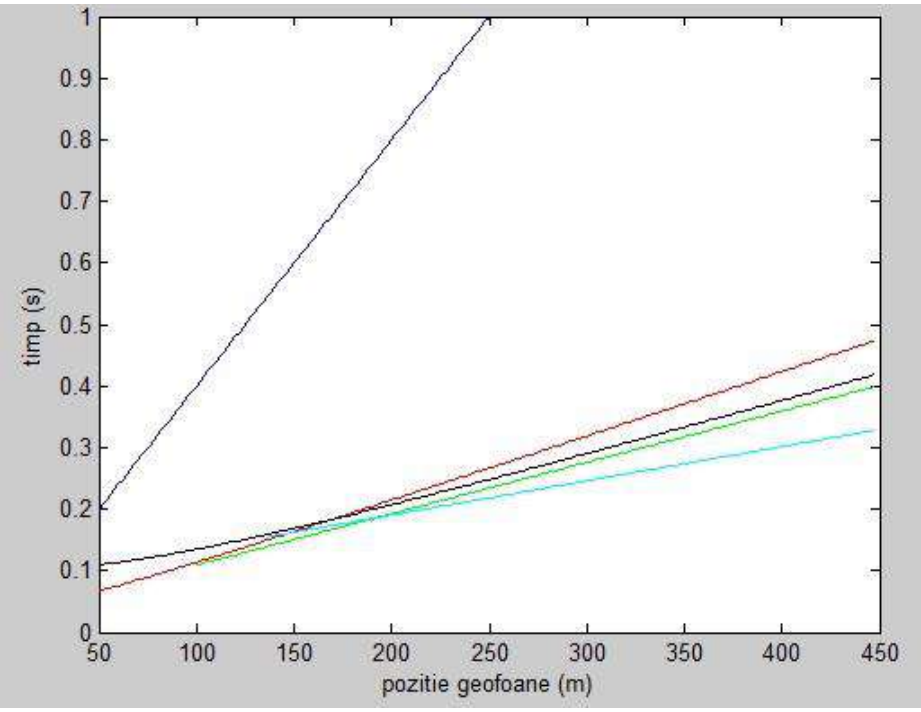

Figura 8.2: Reprezentarea timpilor de propagare calculați pentru o undă de suprafață, două unde reflectate şi două unde frontale. Dispozitiv seismic – împuşcare cu offset de 50 m

Programul de mai jos poate fi folosit pentru obţinerea seismogramei sintetice prezentate în Figura 8.3.

```
clear
% date model geologic
nxg = 200 % numar geofoane
dxg = 2 % metri, distanţă geofoane
h1 = 20 % metri, grosime strat 1
h2 = 35 % metri, grosime strat 2
V1 = 950 % m/s, viteză strat 1
V2 = 1200 % m/s, viteză strat 2
V3 = 1800 % m/s, viteză strat 3
V = 250 % m/s, viteză undă de suprafaţă
offset = 50 % metri

% formule
Vrms = round(sqrt((h1*V1+h2*V2)/((h1/V1)+(h2/V2))))
sin_icr1=V1/V2
cos_icr1=sqrt(1-sin_icr1^2)
tg_icr1=sin_icr1/cos_icr1

sin_icr2=V2/V3
cos_icr2=sqrt(1-sin_icr2^2)
tg_icr2=sin_icr2/cos_icr2
sin_i1=V1/V3
cos_i1=sqrt(1-sin_i1^2)
```

```
tg_i1=sin_i1/cos_i1
x_UF1=round((2*h1*tg_icr1)/dxg)
x_UF2=round((2*h1*tg_i1+2*h2*tg_icr2)/dxg)

for ix=1:nxg
    x(ix)=(ix-1)*dxg; % poziţie geofoane împuşcare capăt
    %x(ix)=offset+(ix-1)*dxg; % poziţie geofoane împuşcare offset
    %x(ix)=(ix-1)*dxg-((nxg-1)*dxg)/2; % poziţie geofoane split-spread
    t(ix)=abs(x(ix))/V;
    t_USR1(ix)=(1/V1)*sqrt(x(ix)^2+4*h1^2);
    t_UF1(ix)=(2*h1)/(cos_icr1*V1)+(abs(x(ix))-2*h1*tg_icr1)/V2;
    t_USR2(ix)=(1/Vrms)*sqrt(x(ix)^2+4*(h1+h2)^2);
    t_UF2(ix)=(2*h1)/(cos_i1*V1)+(2*h2)/(cos_icr2*V2)+(abs(x(ix))-
(2*h1*tg_i1+2*h2*tg_icr2))/V3;
end

% offset
plot(x,t)
hold on
plot(x,t_USR1,'r')
plot(x,t_USR2,'k')
plot(x(x_UF1:nxg),t_UF1(x_UF1:nxg),'g')
plot(x(x_UF2:nxg),t_UF2(x_UF2:nxg),'c')
xlabel('pozitie geofoane (m)')
ylabel('timp (s)')
axis([0 400 0 1])
```

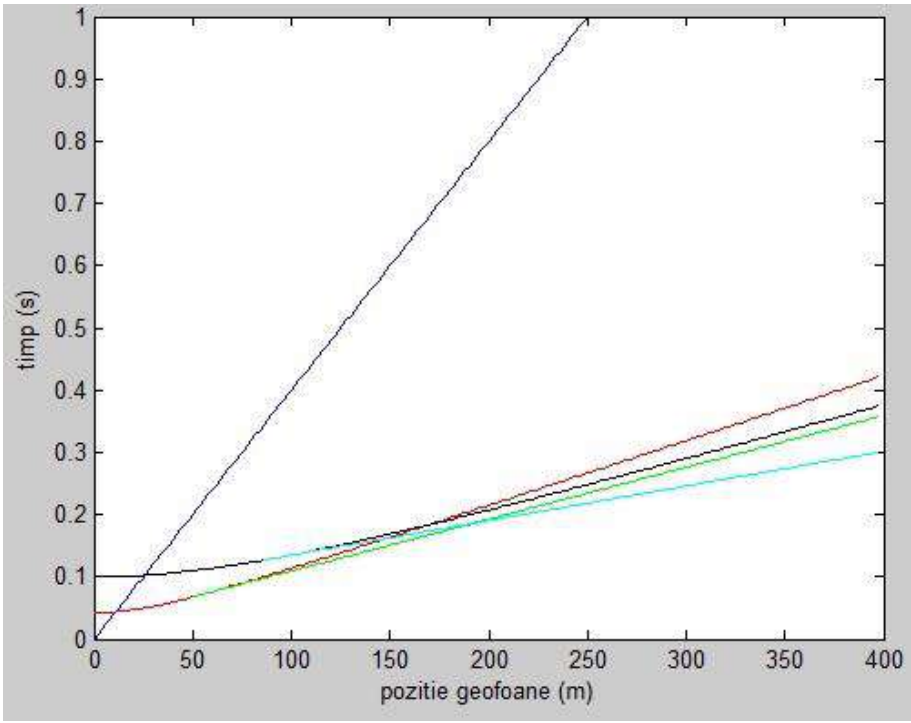

Figura 8.3: Reprezentarea timpilor de propagare calculaţi pentru o undă de suprafaţă, două unde reflectate şi două unde frontale. Dispozitiv seismic – împuşcare la capăt

8.2 Modelare folosind forma de undă Ricker

Seismogramele sintetice pot fi modelate folosind forma de undă Ricker; în acest fel, undele seismice modelate pot fi considerate asemănătoare cu cele de pe înregistrările seismice de teren.

Un exemplu de program folosit pentru modelarea formei de undă Ricker este descris mai jos:

```
function [wavelet] = ricker_lili(nt,dt,t0,f0)
%
%  această funcţie calculează o formă de undă Ricker la timpul
%  rickt=t0 si freq=f0:
%
%
disp(' within function ricker');

pi2 = pi*pi;
f02 = f0*f0;
for it=1:nt
    tt    = (it-1).*dt;
    tt02  = (tt-t0).*(tt-t0);
    ee    = pi2.*f02.*tt02;
    if ee < 60.0
      wavelet(it)=(1.-2.0*pi2*f02*tt02) .* exp(-pi2*f02*tt02);
    else
      wavelet(it)=0.0;
    end
    clear tt tt02 ee;
end
plot(wavelet);
```

Modelarea unei seismograme ce conţine o undă de suprafaţă şi o undă reflectată se poate obţine folosind programul de mai jos (Figura 8.4):

```
clear
% dispozitiv seismic și forma de undă Ricker
nxg = 100 % număr de geofoane
dxg = 12.5 % metri, distanța dintre geofoane
dt = 0.002 % interval de eșantionare în timp
nt = 1001 % număr total de eșantioane în timp
rickt0 = 0.048 % timp central formă de undă Ricker
rickf0 = 16    % frecvența dominantă pentru zgomot
rickf  = 36    % frecvența dominantă pentru semnal
%offset = 10 % distanța sursă - geofon 1

% date model geologic
thick(1) = 350   % grosime strat 1
vel(1)   = 2250  % viteza undei reflectate de la baza stratului 1
velgr(1) = 450   % viteza undei de suprafață

% calcul poziție geofoane
```

```
for ix=1:nxg
    %x(ix) = dxg*(ix-1); % împuscare la capat
    %x(ix) = offset + dxg*(ix-1); % împuscare cu offset
    x(ix) = (nxg*dxg)/2 - dxg*(ix-1); % împuscare la centru
end
x_hyperb=x;

% calcul timpi sosire unde seismice
% unda de suprafață
for ix=1:nxg
    tdirect(ix) = abs((x(ix)/velgr(1)));
end
% unda reflectată
for ix=1:nxg
    t0 = (2*thick(1))/vel(1);
    trefl(ix)=sqrt((x_hyperb(ix)/vel(1))^2+t0^2);
    trefl(ix)=sqrt((x(ix)/vel(1))^2+t0^2);
end

% modelare seismograme
ntpow2 = 2^(floor(log2(nt))+1)
nf = ntpow2;
df   = 1./(nf*dt);
dom  = 2.*pi*df;

% unda de suprafață
[wavelet] = ricker_lili(ntpow2,dt,rickt0,rickf0);
fftwav = fft(wavelet);
cf = complex(zeros(1,nf),zeros(1,nf));
for ix=1:nxg
   for ifreq = 1:nf/2+1
      om(ifreq) = (ifreq-1).*dom;
      cc(ifreq) = complex(0.0, om(ifreq).*(tdirect(ix)-rickt0));
      cf(ifreq) = fftwav(ifreq).*exp(-cc(ifreq));
   end
   for ifreq=nf/2+2:nf
      if1 = nf-ifreq+2;
      rr  = real(cf(if1));
      ri  = imag(cf(if1));
      cf(ifreq) = complex(rr,-ri);
   end
   data = ifft(cf);
   for it=1:nt
      St(it,ix) = real(data(it));
   end
end

% unda reflectată
[wavelet] = ricker_lili(ntpow2,dt,rickt0,rickf);
fftwav = fft(wavelet);
cf = complex(zeros(1,nf),zeros(1,nf));
for ix=1:nxg
    tr(ix) = trefl(ix)-rickt0;
    for ifreq = 1:nf/2+1
        om(ifreq) = (ifreq-1).*dom;
        cc(ifreq) = complex(0.0, om(ifreq).*tr(ix));
```

```
        cf(ifreq) = fftwav(ifreq).*exp(-cc(ifreq));
    end
    for ifreq=nf/2+2:nf
        if1 = nf-ifreq+2;
        rr  = real(cf(if1));
        ri  = imag(cf(if1));
        cf(ifreq) = complex(rr,-ri);
    end
    data = ifft(cf);
    for it=1:nt
        St(it,ix) = St(it,ix) + real(data(it));
    end
end % end of ix-loop

plotseis(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

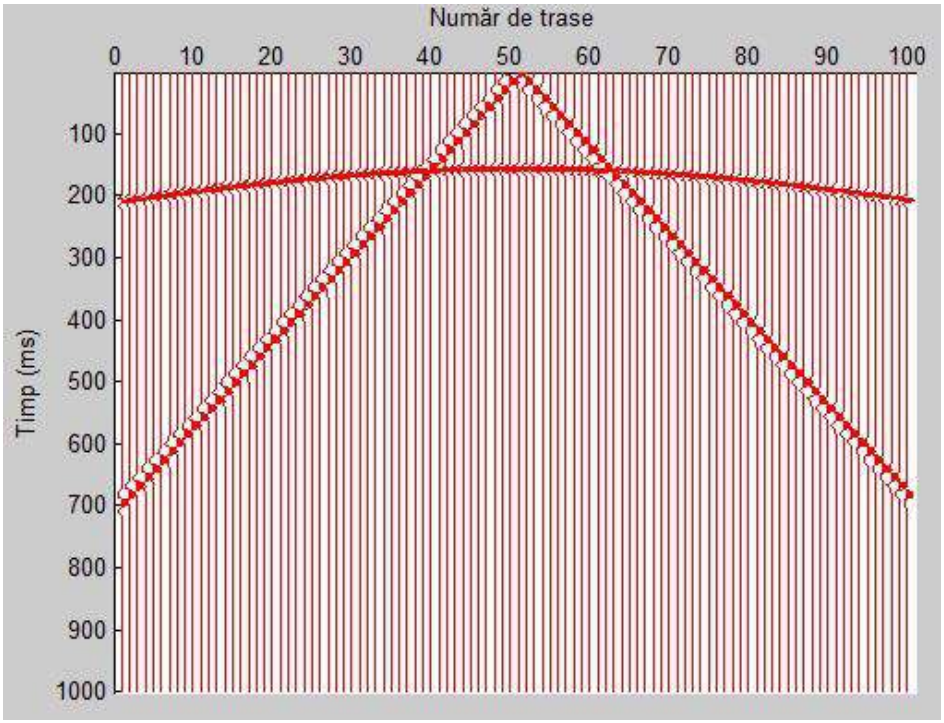

Figura 8.4: Seismogramă ce conţine o undă de suprafaţă şi o undă reflectată de la o limită orizontală

8.3 Interferenţa undelor seismice

Undele reflectate interferă destructiv sau constructiv în drumul lor spre receptoarele plasate la suprafaţa solului. Programul de mai jos calculează şi reprezintă răspunsul interferenţei dintre o undă reflectată de acoperişul unui strat orizontal şi cea

reflectată de la culcuşul acestui strat. Datele de intrare sunt reprezentate de o formă de undă şi funcţia de reflectivitate din punctul în care se fac măsurătorile. Grosimea stratului este exprimată în secunde, valoare determinată folosind timpul ecou (t = 2h/V). În Figura 8.5 este prezentată forma de undă iar în Figura 8.5 este prezentat răspunsul interferenţei între undele seismice.

```
clear
%% editez semnalul (x0) şi funcţia de reflectivitate (r0):
x0 = [0 0.3 0.5 1.6 0.6 0.4 -0.01 -0.05 -0.25 -0.05 -0.01 0]
r0 = [-0.48 zeros(1,15) 0.7]

%% NU MODIFICAȚI %%%%%%%%%%%%%%%%%%%%%%%%%%
nx0 = length(x0)
nr0 = length(r0)
ny = nx0+ nr0-1
nx = ny - nx0
nr = ny - nr0

clear r x
x = [x0 zeros(1,nx)]
r = [r0 zeros(1,nr)]
clear out_y y flip_out_x out_x out_r
if nr0 < nx0
    for iy=1:ny
        out_x(iy)= x(iy);
        flip_out_x = fliplr(out_x);
        out_r(iy)=r(iy);
        y = flip_out_x*out_r';
        out_y(iy) = y;
    end
else
    for iy = 1:ny
        if iy <= nx0
            iy;
            out_x(iy)= x0(iy);
            flip_out_x = fliplr(out_x);
            out_r(iy)=r0(iy);
            y = flip_out_x*out_r';
            out_y(iy) = y;
        else
            iy;
            out_x(iy)=x(iy);
            flip_out_x = fliplr(out_x);
            out_r(iy)=r(iy);
            y=flip_out_x*out_r';
            out_y(iy)=y;
        end
    end
end
out_y
plot(x0)
xlabel('Eşantioane timp')
ylabel('Amplitudine')
figure(2)
```

```
plot(out_y)
xlabel('Eşantioane timp')
ylabel('Amplitudine')
```

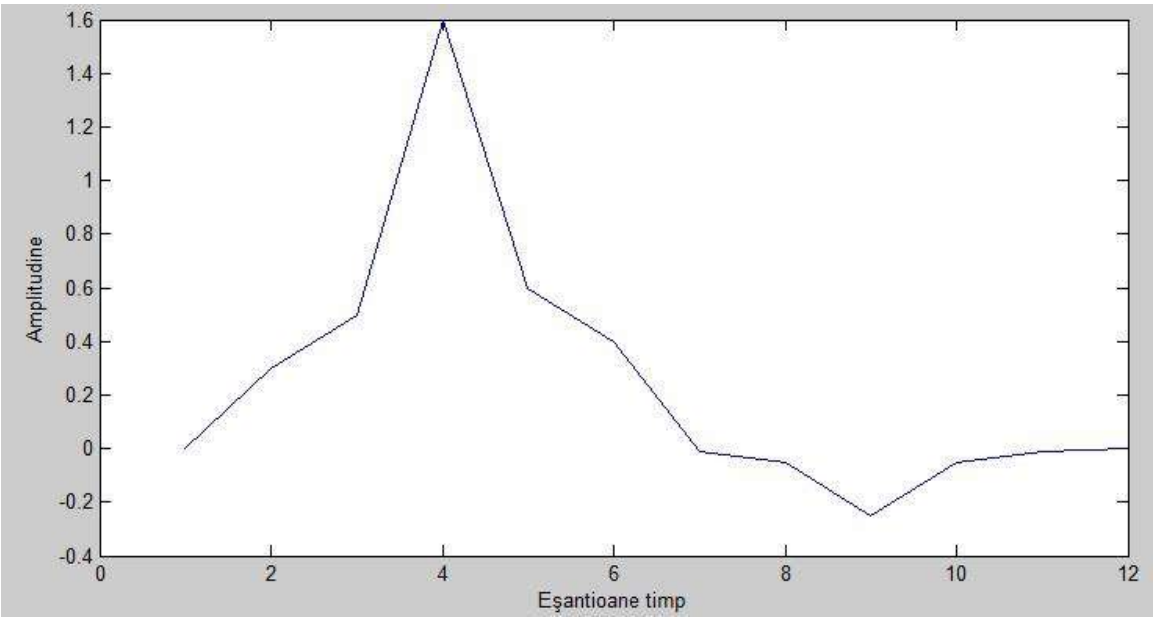

Figura 8.5: Forma de undă înainte de interferenţă

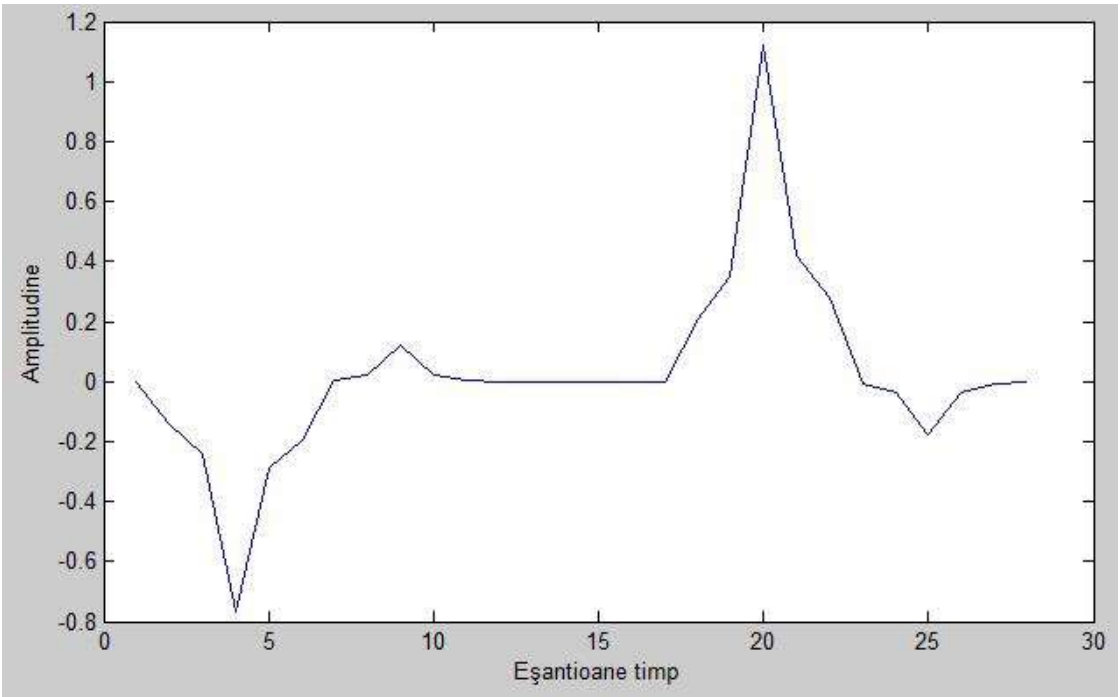

Figura 8.6: Forma de undă după interferenţă pentru o grosime a stratului de 64 ms: fără interferenţă

Exerciţii fără rezolvare afişată:

E1: Calculaţi şi reprezentaţi grafic undele de suprafaţă, undele reflectate şi undele frontale pentru modelul de mai jos. Calculele se efectuează pentru un dispozitiv cu împuşcare la capăt.

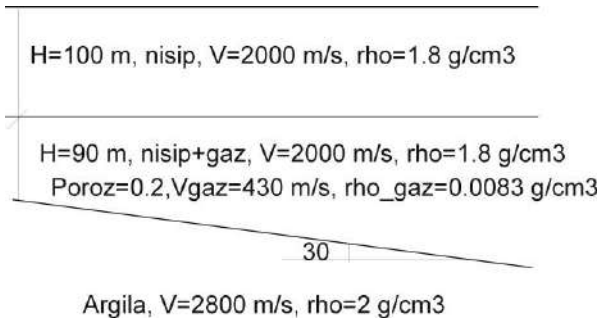

Figura 8.7: Model geologic folosit pentru obţinerea seismogramei sintetice

E2: Calculaţi şi reprezentaţi grafic undele de suprafaţă, undele reflectate şi undele frontale pentru modelul de mai jos. Calculele se efectuează pentru un dispozitiv cu împuşcare cu offset.

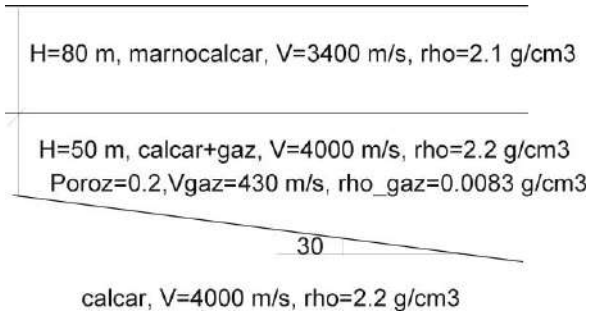

Figura 8.8: Model geologic folosit pentru obţinerea seismogramei sintetice

E3: Calculaţi şi reprezentaţi grafic undele de suprafaţă, undele reflectate şi undele frontale pentru modelul de mai jos. Calculele se efectuează pentru un dispozitiv cu împuşcare cu împuşcare la centru.

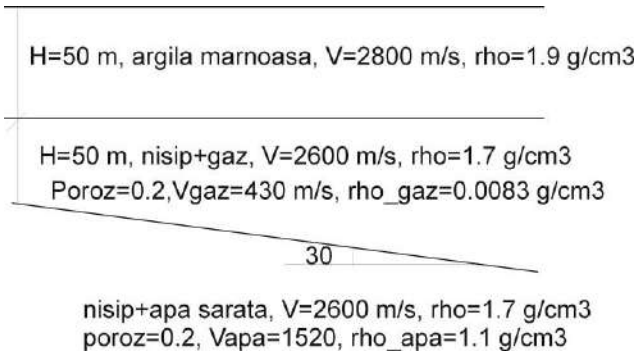

Figura 8.9: Model geologic folosit pentru obţinerea seismogramei sintetice

E4: Construiţi seismograme sintetice pentru dispozitive cu împuscare la capăt şi cu împuşcare cu offset.

E5: Calculaţi şi reprezentaţi o seismogramă cu formă de undă Ricker pentru modelul geologic descris la exercitţiul E1.

E6: Calculaţi şi reprezentaţi o seismogramă cu formă de undă Ricker pentru modelul geologic descris la exercitţiul E2.

E7: Calculaţi şi reprezentaţi o seismogramă cu formă de undă Ricker pentru modelul geologic descris la exercitţiul E3.

E8: Editaţi programul folosit pentru a obţine răspunsurile interferenţei seismice dintre undele reflectate de la prima şi a doua limită din modelele geologice prezentate în Figurile 8.7-8.9. Punctele în care se fac calculele sunt plasate la x = 0, 250, 500 şi 750 m.

Capitolul 9
Prelucrarea automată a datelor seismice

9.1 Citirea şi salvarea automată a datelor seismice

Un set de seismograme salvate în formatul SEG-Y poate fi citit seimsogramă cu seismogramă şi, apoi, salvat în formatul dorit, de exemplu Seismic Unix (SU). Funcţia `num2str` permite citirea automată a mai multor seismograme. Se definesc două variabile, una de intrare, `f_in`, şi a doua de ieşire, `f_out`, care conţin căile către fişierele analizate şi numele acestora. Verificarea programului se face comparând valorile eşantioanelor de timp selectate din seismograma salvată în formatul SEG-Y cu cele din aceeaşi seismogramă dar salvată în formatul SU.

```
clear

% date intrare
sh1 = 3
sh2 = 7

for ishot = sh1:sh2
    f_in = ['D:\MATLAB\SH',num2str(ishot),'_agc.sgy'];
    St = segyread(f_in);

    f_out = ['D:\MATLAB\SH',num2str(ishot),'_agc.su'];
    WriteSu_dt1(f_out,St);
end

% pentru verificare
St_sgy = segyread('D:\MATLAB\SH7_agc.sgy');
St_su = ReadSu('D:\MATLAB\SH7_agc.su');

>> St_sgy(200:205,30:35)

ans =
    1.0664   -0.2841    0.2924    0.1953   -0.3665   -0.4373
    1.0714   -0.6955    0.3686   -0.0530   -0.4691   -0.4035
    0.7478   -0.7501    0.0522   -0.3166   -0.5716   -0.3409
    0.3172   -0.6586   -0.0771   -0.5092   -0.6153   -0.2329
         0   -0.4272    0.0757   -0.6675   -0.5851   -0.0994
    0.1178         0    0.1233   -0.7926   -0.5402    0.0357

>> St_su(200:205,30:35)

ans =
    1.0664   -0.2841    0.2924    0.1953   -0.3665   -0.4373
    1.0714   -0.6955    0.3686   -0.0530   -0.4691   -0.4035
    0.7478   -0.7501    0.0522   -0.3166   -0.5716   -0.3409
    0.3172   -0.6586   -0.0771   -0.5092   -0.6153   -0.2329
         0   -0.4272    0.0757   -0.6675   -0.5851   -0.0994
    0.1178         0    0.1233   -0.7926   -0.5402    0.0357
```

9.2 Selectarea automată a datelor seismice

În măsurătorile seismice pasive, panourile de zgomot se salvează cu lungimi mari de timp, de exemplu 50 s sau 60 s. Panourile folosite în prelucrare au lungimi mult mai mici datorită limitărilor impuse de programele folosite. Selectarea şi salvarea automată a acestor panouri se poate face folosind funcţia `num2str` şi vectorii de timp cu dimensiunile dorite.

În programul scris mai jos se citesc automat cinci panouri de zgomot cu 48 trase şi 2000 eşantioane de timp. Fiecare panou se împarte în alte două panouri cu 1000 eşantioane de timp fiecare apoi se salvează automat în formatul SU. Verificarea se face prin afişarea şi compararea valorilor de la începutul şi sfârşitul fiecărui panou cu o lungime mai mică decât a celor corespunzătoare ca poziţie din panoul iniţial.

```
clear
% date intrare
sh1 = 181
sh2 = 185
ntime = 2
nt1 = 1000
nt2 = 2*nt1

for ishot = sh1:sh2
    f_in = ['D:\MATLAB\shot',num2str(ishot),'_inline.sgy'];
    St = segyread(f_in);

    Stw1 = St(1:nt1,:);
    f1_out = ['D:\MATLAB\shot',num2str(ishot),'_1.su'];
    WriteSu_1dt(f1_out,Stw1);

    Stw2 = St(nt1+1:nt2,:);
    f2_out = ['D:\MATLAB\shot',num2str(ishot),'_2.su'];
    WriteSu_1dt(f2_out,Stw2);
end

% pentru verificare
St1 = ReadSu('D:\MATLAB\shot181_1.su');
St2 = ReadSu('D:\MATLAB\shot181_2.su');
St = segyread('D:\MATLAB\shot181_inline.sgy');

St(999:1002,1:5)
St1(999:1000,1:5)
St2(1:2,1:5)

ans =

     1     2     0    -5    -1
     2     0    -1    -7    -2
     2     0     0    -6     0
    -3     0    -1    -2    -5


ans =
```

```
     1     2     0    -5    -1
     2     0    -1    -7    -2


ans =

     2     0     0    -6     0
    -3     0    -1    -2    -5
```

Selectarea automată a unui grup de trase dintr-o înregistrare seismică şi rearanjarea traselor în poziţia lor corectă de pe profilul seismic se poate face folosind programul de mai jos:

```
clear
% date intrare
sh1 = 3
sh2 = 7

for ishot = sh1:sh2
    f_in = ['D:\MATLAB\SH',num2str(ishot),'_agc.sgy'];
    St = segyread(f_in);

    St_in = [St(:,1:24) fliplr(St(:,25:48))];
    f1_out = ['D:\MATLAB\SH',num2str(ishot),'_in_agc.su'];
    WriteSu_dt1(f1_out,St_in);

    St_xl = fliplr(St(:,49:72));
    f2_out = ['D:\MATLAB\SH',num2str(ishot),'_xl_agc.su'];
    WriteSu_dt1(f2_out,St_xl);
end

% pentru verificare
St_in = ReadSu('D:\MATLAB\SH3_in_agc.su');
plotimage(St_in)
xlabel('Număr de trase')
ylabel('Timp (ms)')

St_xl = ReadSu('D:\MATLAB\SH3_xl_agc.su');
plotimage(St_xl)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

Trasele din înregistrările `SH3_agc.sgy - SH7_agc.sgy` nu au fost salvate în poziţia lor corectă de pe profil (Figura 9.1). Fiecare înregistrare conţine trei grupuri a câte 24 de trase. Trasele cu numerele 1 – 48 au fost înregistrate folosind o întindere liniară de 48 de geofoane. Trasele cu numerele 49 – 72 au fost înregistrate folosind o întindere de 24 de geofoane orientate perpendicular pe cea de 48 de geofoane. Trasele cu numerele 1 – 48 şi 49 – 72 trebuie salvate în fişiere separate. În plus, trasele 25 – 48 trebuie aranjate în poziţia lor corectă folosind funcţia `fliplr`. În Figura 9.1 este prezentată seismograma `SH3_agc.sgy` înainte de aranjarea traselor. Versiunea corectată a acestei seismograme pentru primele 48 de trase este prezentată în Figura 9.2.

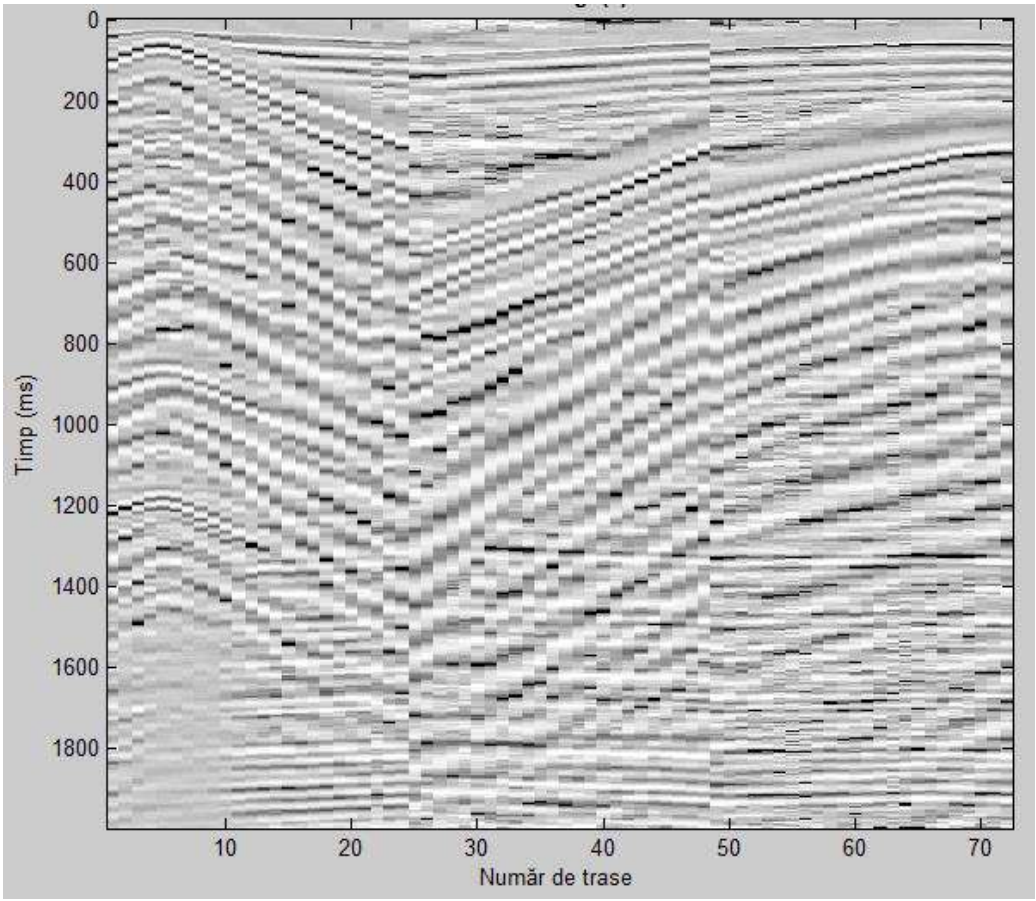

Figura 9.1: Seismogramă inainte de aranjarea traselor cu numerele 1 – 48

9.3 Atenuarea automată a zgomotului coerent

Zgomotul coerent prezent pe un set de date seismice înregistrate folosind geofoane individuale se poate atenua folosind o grupare de geofoane cu salvare automată a raspunsurilor grupărilor de geofoane. Un exemplu de program care defineşte şi aplică o grupare de geofoane liniară este scris mai jos. Seismogramele înainte şi după definirea grupărilor de geofoane sunt prezentate în Figurile 9.3 şi 9.4.

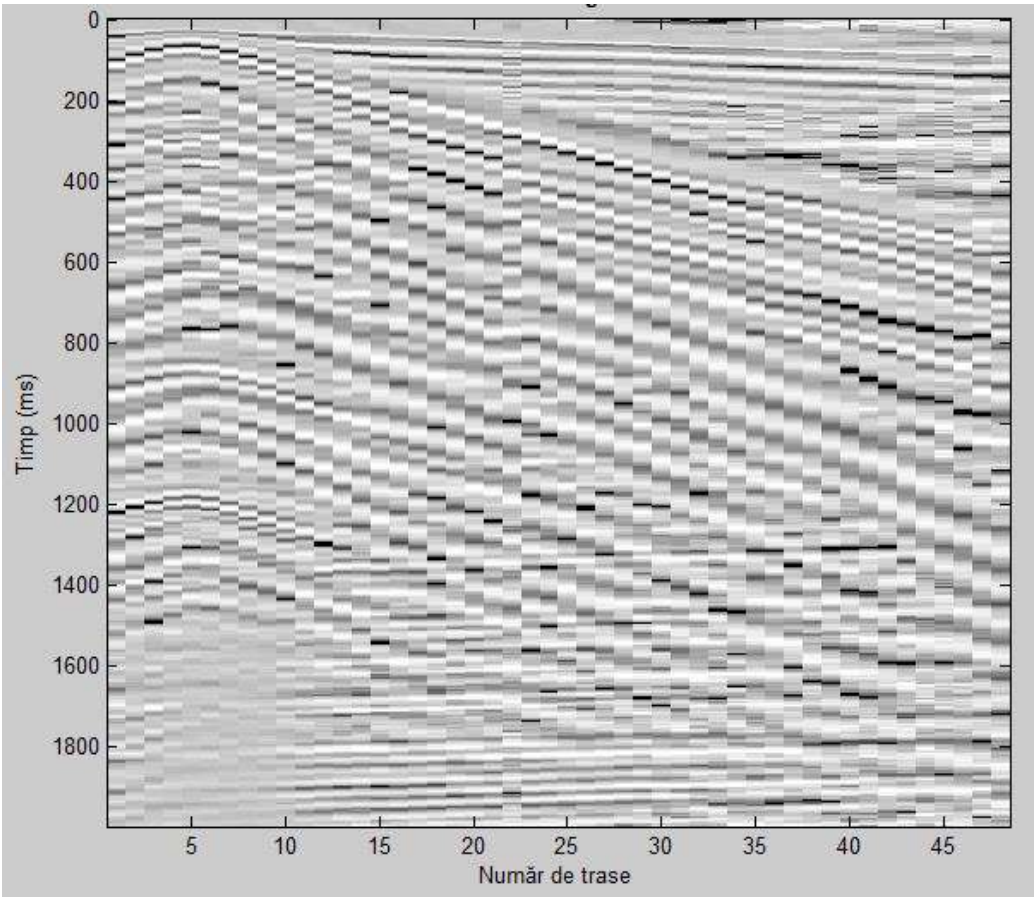

Figura 9.2: Seismogramă după aranjarea traselor cu numerele 1 – 48

```
clear
% date de intrare
sh1 = 1
sh2 = 5

nxg = 100  % număr de geofoane
nt = 2001 % număr de eşantioane de timp
nx1 = 5 % număr de geofoane în grupare
ax = 1:1:nx1
nxout = nxg - nx1 % număr trase după însumarea de trase
dxG = 1 % interval de grup = dxg

for ishot = sh1:sh2
    f_in = ['D:\Matlab\2025_',num2str(ishot),'_agc.sgy'];
    St = segyread(f_in);

    for ixout = 1:nxout
        out = (ixout - 1) + ax;
        Stw = St(:,out);
        Saw = zeros(nt,1);
        for ix=1:nx1
            Saw(:) = Saw(:) + Stw(:,ix);
```

```
        end
        Sawn = Saw/nx1;
        San(:,ixout) = Sawn;
    end % ixout-loop

    f_out = ['D:\Matlab\2025_',num2str(ishot),'_add.su'];
    writesu_1dt(f_out,San);
end

% pentru verificare
St = segyread('D:\Matlab\2025_1_agc.sgy');
plotimage(St)
xlabel('Număr de trase')
ylabel('Timp (ms)')
San = readsu('D:\Matlab\2025_1_add.su');
plotimage(San)
xlabel('Număr de trase')
ylabel('Timp (ms)')
```

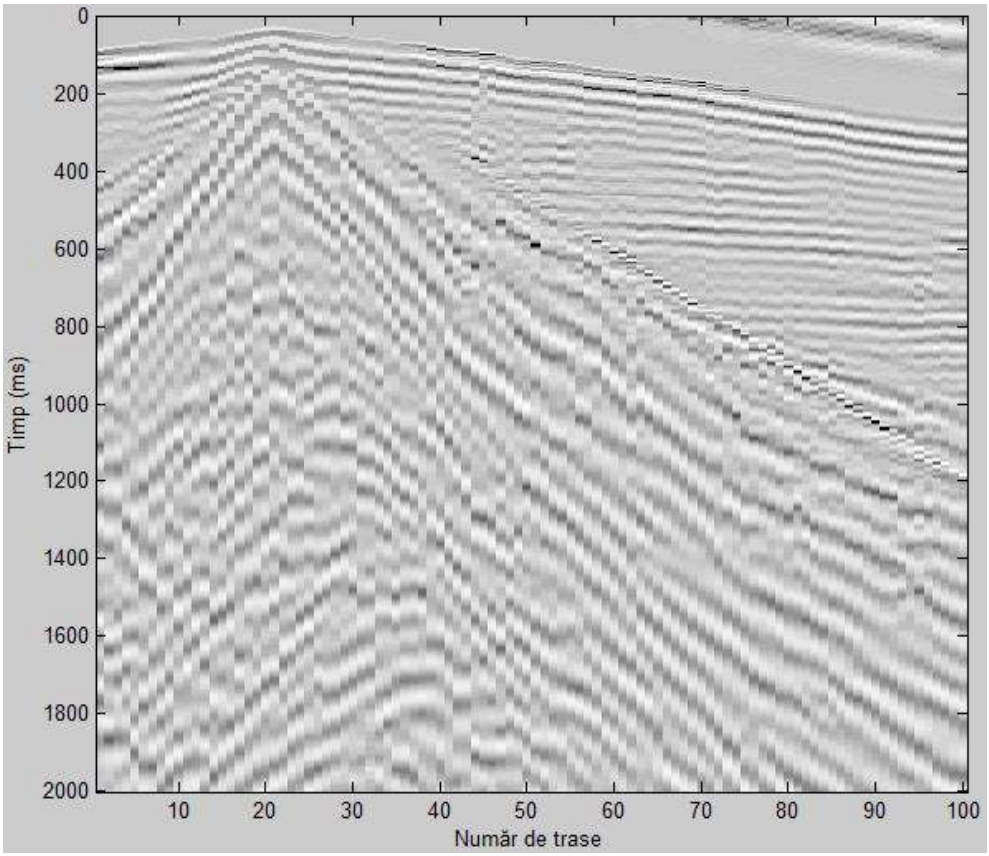

Figura 9.3: Seismogramă înainte de gruparea de geofoane

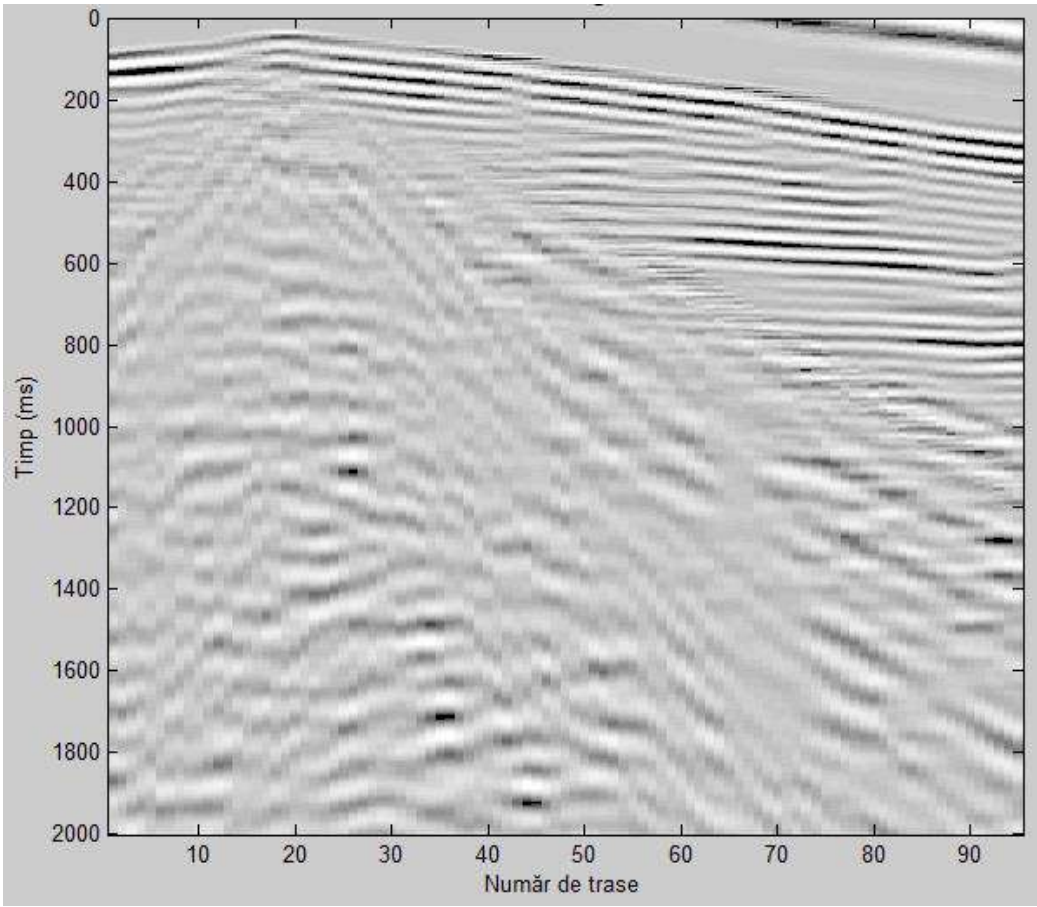

Figura 9.4: Seismogramă după gruparea de geofoane

9.4 Însumarea automată a seismogramelor

Prelucrarea datelor seismice pasive foloseşte metode în care mai multe panouri de zgomot sunt însumate într-unul singur. Programul de mai jos poate fi folosit pentru însumarea unui număr mare de panouri de zgomot. Pentru început se scrie şi se rulează programul pentru un număr mic de panouri de zgomot, de exemplu pentru patru panouri, şi se verifică rezultatele. Dacă rezultatele sunt corecte, programul se poate rula pe sute sau mii de panouri de zgomot.

Pentru normalizare, răspunsul însumării se poate împarţi la numărul de panouri însumate.

```
clear

% pentru verificare mărime panou:
St1 = segyread('D:\Matlab\shot181_inline.sgy');
nt = max(size(St1))

% definire panou însumat:
```

```
St2 = segyread('D:\Matlab\shot182_inline.sgy');
St_add = St1+St2;
writesu_1dt('D:\Matlab\shot_add.su',St_add);

% adunare panouri
npanel=3
for ipanel=1:npanel
    f_in=['D:\Matlab\shot',num2str(ipanel+182),'_inline.sgy'];
    St = segyread(f_in);

    Sa = readsu('D:\Matlab\shot_add.su');
    St_add = St+Sa;
    writesu_1dt('D:\Matlab\shot_add.su',St_add);

end

% pentru verificare
Sa = readsu('D:\Matlab\shot_add.su');
St1 = segyread('D:\Matlab\shot181_inline.sgy');
St2 = segyread('D:\Matlab\shot182_inline.sgy');
St3 = segyread('D:\Matlab\shot183_inline.sgy');
St4 = segyread('D:\Matlab\shot184_inline.sgy');
St5 = segyread('D:\Matlab\shot185_inline.sgy');

Sa(1:6,1:6)
St1(1:6,1:6)+St2(1:6,1:6)+St3(1:6,1:6)+St4(1:6,1:6)+St5(1:6,1:6)

ans =

    -2    -9    -6    -2     3    -6
    -2    -5    -6    -1    -1    -9
     2     5    -1    -8     2    -5
    -7     0    -1    -9   -16    -3
    -2    -4    -4    -5   -16    -2
     7    -4     1   -12     2     0


ans =

    -2    -9    -6    -2     3    -6
    -2    -5    -6    -1    -1    -9
     2     5    -1    -8     2    -5
    -7     0    -1    -9   -16    -3
    -2    -4    -4    -5   -16    -2
     7    -4     1   -12     2     0
```

9.5 Însumarea automată a traselor dintr-un set de date seismice

Analizele de frecvenţă efectuate folosind Transformata Fourier 1D necesită, uneori, însumarea tuturor traselor din fiecare panou de zgomot pentru a se obţine o singură trasă pentru fiecare panou. Însumarea de trase se efectuează în domeniul timp. Programul de mai jos poate fi folosit pentru analiza unui număr foarte mare de panouri de zgomot (sute, mii de panouri de zgomot). Pentru verificarea programului, însumarea se

execută numai pentru cinci panouri şi se compară rezultatul însumării traselor dintr-un singur panou, efectuată separat, cu valorile conţinute de prima trasă obţinută din folosirea programului. Dacă valorile coincid se rulează programul pe întregul set de date seismice pasive analizat.

```
clear

% pentru verificare mărime panou:
St1 = segyread('D:\Matlab\shot181_inline.sgy');
nt = max(size(St1))
nxg = min(size(St1))

% însumare trase dintr-un singur panou
npanel=5
San_panel = zeros(nt,npanel);

for ipanel=1:npanel
    f_in=['D:\Matlab\shot',num2str(ipanel+180),'_inline.sgy'];
    St = segyread(f_in);
    clear Sa San
    Sa = zeros(nt,1);
    for ix=1:nxg
        Sa(:)=Sa(:)+St(:,ix);
    end

    San = Sa/nxg; % normalizare
    San_panel(:,ipanel)=San; % salvare trasă din fiecare panou
end
plotimage(San_panel)
xlabel('Număr de trase')
ylabel('Timp (ms)')

% pentru verificare mediez trasele dintr-un panou şi compar rezultatul
% cu trasa corespunzatoare din San_panel:

St = segyread('D:\Matlab\shot181_inline.sgy');
nt = max(size(St))
nxg = min(size(St))

Sa = zeros(nt,1);
for ix=1:nxg
    Sa(:) = Sa(:) + St(:,ix);
end
San = Sa/nxg;
San(1:6,1)
San_panel(1:6,1)

ans =

   -0.1667
    0.0208
    0.1042
    0.5208
    0.5625
```

```
    0.2917
ans =

   -0.1667
    0.0208
    0.1042
    0.5208
    0.5625
    0.2917
```

9.6 Analiza spectrală automată a traselor dintr-un set de date seismice

Transformata Fourier 1D se poate aplica pentru fiecare trasă din fiecare panou de zgomot analizat cu salvarea fiecărui panou de spectre în parte. Programul de mai jos poate fi folosit pentru această analiză. Pentru verificare, se aplică transformata pe o singură trasă şi se compară cu trasa corespunzătoare acesteia din fişierul rezultat după rularea programului. În Figura 9.5 este prezentat spectrul de amplitudine al unei trase selectate dintr-un panou de zgomot obținut prin aplicarea directă a transformatei Fourier. Spectrul de amplitudine pentru aceeași trasă dar calculat folosind programul descris mai jos este prezentat în Figura 9.6.

```
clear
% pentru verificare mărime panou:
St1 = segyread('D:\MATLAB\panel_1.sgy');
nt = max(size(St1))
nxg = min(size(St1))
dt = 0.001
nf = pow2(floor(log2(nt)+1)) % număr eșantioane frecvență
df = 1./(nf*dt) % interval eșantionare în frecvență
freq = df*(1:nf);
nf1 = 100 % număr eșantioane pentru zoom

npanel = 4
for ipanel=1:npanel
    f_in = ['D:\MATLAB\panel_',num2str(ipanel),'.sgy'];
    St = segyread(f_in);
    for ix = 1:nxg
        T1 = St(:,ix);
        abs_T1 = abs(fft(T1,nf));
        abs_St(:,ix) = abs_T1;
    end
    f_out = ['D:\MATLAB\amp_',num2str(ipanel),'.su'];
    WriteSu_dt1(f_out,abs_St);
end

% pentru verificare
St1 = segyread('D:\MATLAB\panel_1.sgy');
plot(abs(fft(St1(:,1),nf)))
xlabel('Eșantioane de frecvență')
ylabel('Amplitudine')
axis([0 nf1 0 50])
```

```
amp = ReadSu('D:\MATLAB\amp_1.su');
figure(2)
plot(amp(:,1))
xlabel('Eşantioane de frecvenţă')
ylabel('Amplitudine')
axis([0 nf1 0 50])
```

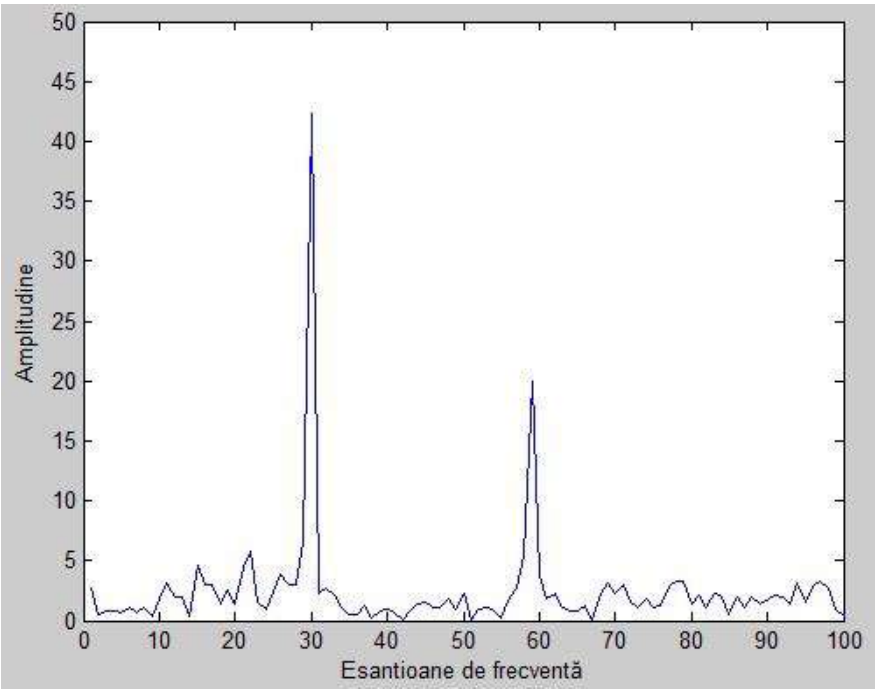

Figura 9.5: Primele 100 de eşantioane de frecvenţă selectate din spectrul de amplitudine al TF 1D aplicată pe prima trasă din panoul `panel_1.sgy`

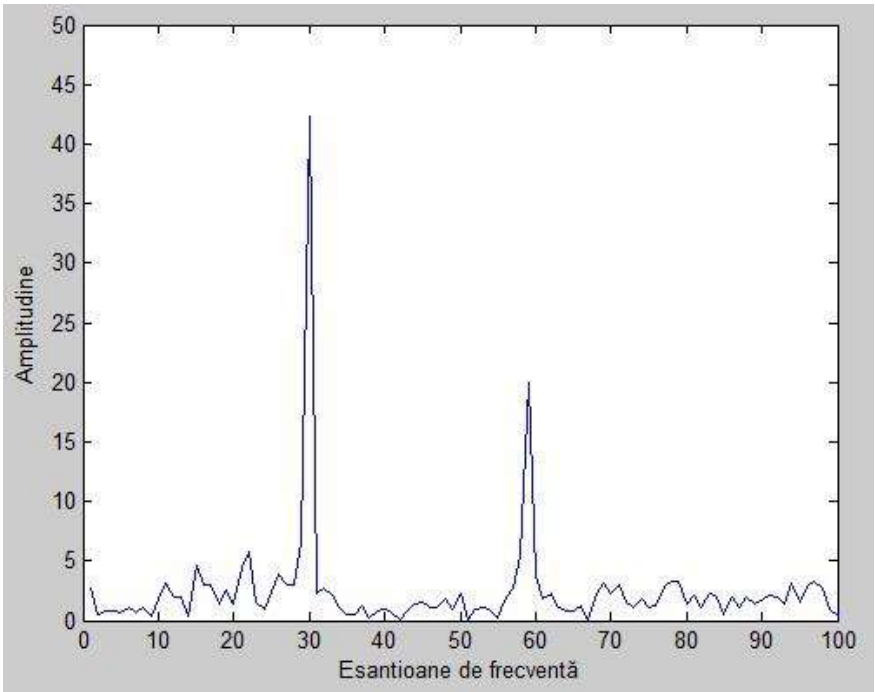

Figura 9.6: Primele 100 de eşantioane de frecvenţă selectate din spectrul de amplitudine al trasei corespunzătoare celei folosite în Figura 9.5 calculat folosind programul descris mai sus

9.7 Însumarea automată a spectrelor dintr-un set de date seismice

Analiza datelor seismice pasive în domeniul de frecvenţă necesită, uneori, însumarea spectrelor de amplitudine obtinute pentru fiecare trasă dintr-un panou de zgomot intr-un singur spectru pe panou. Apoi, toate spectrele de amplitudine 1D sunt salvate într-un singur fişier, urmând a fi folosit mai departe în fluxul de prelucrare a datelor seismice pasive.

Un exemplu de program care face această analiză şi salvare este descris mai jos. Pentru verificare se calculează şi însumează toate spectrele de amplitudine 1D dintr-un panou de zgomot şi se compară cu spectrul de amplitudine 1D corespunzător acelui panou calculat folosind programul de mai jos.

```
clear

% pentru verificare mărime panou:
St1 = segyread('D:\MATLAB\panel_1.sgy');
nt = max(size(St1))
nxg = min(size(St1))
dt = 0.001
nf = pow2(floor(log2(nt)+1)) % număr eşantioane frecvenţă
df = 1./(nf*dt) % interval eşantionare în frecvenţă
freq = df*(1:nf);
nf1 = 100 % număr eşantioane în frecvenţă pentru zoom
```

```
npanel = 4
for ipanel=1:npanel
    f_in = ['D:\MATLAB\panel_',num2str(ipanel),'.sgy'];
    St = segyread(f_in);

    for ix = 1:nxg
        T1 = St(:,ix);
        abs_T1 = abs(fft(T1,nf));
        abs_St(:,ix) = abs_T1;
    end
    clear abs_f abs_fn
    abs_f = zeros(nf,1);
    for ix = 1:nxg
        abs_f(:) = abs_f(:)+abs_St(:,ix);
    end
    abs_fn = abs_f/nxg;
    abs_fn_panel(:,ipanel) = abs_fn;
end

plotseis(abs_fn_panel(1:nf1,:))
ylabel('Eșantioane de frecvență')
xlabel('Număr de spectre de amplitudine 1D')

% pentru verificare
St4 = segyread('D:\MATLAB\panel_4.sgy');
for ix = 1:nxg
    T4 = St4(:,ix);
    abs_T4 = abs(fft(T4,nf));
    abs_St4(:,ix) = abs_T4;
end
clear abs_f_4 abs_fn_4
abs_f_4 = zeros(nf,1);
for ix = 1:nxg
    abs_f_4(:) = abs_f_4(:)+abs_St4(:,ix);
end
abs_fn_4 = abs_f_4/nxg;
figure(2)
plot(abs_fn_4(1:nf1))
xlabel('Eșantioane de frecvență')
ylabel('Amplitudine')

figure(3)
plot(abs_fn_panel(1:nf1,4))
xlabel('Eșantioane de frecvență')
ylabel('Amplitudine')
```

În Figura 9.7 sunt prezentate primele 100 de eșantioane de frecvență selectate din spectrele de amplitudine însumate obținute pentru fiecare panou în parte. Pentru comparare, primele 100 de eșantioane selectate din spectrul de amplitudine obținut din aplicarea directă a Transformatei Fourier 1D pe trasele din panoul `panel_4.sgy` și însumarea acestora sunt prezentate în Figura 9.8. Eșantioanele corespunzătoare ca poziție în panou obținute după analiza automată a datelor sunt prezentate în Figura 9.9. Din

analiza spectrelor rezultă că valorile corespund ceea ce înseamnă că programul a funcționat corect și poate fi aplicat pe întregul set de panouri de zgomot analizat.

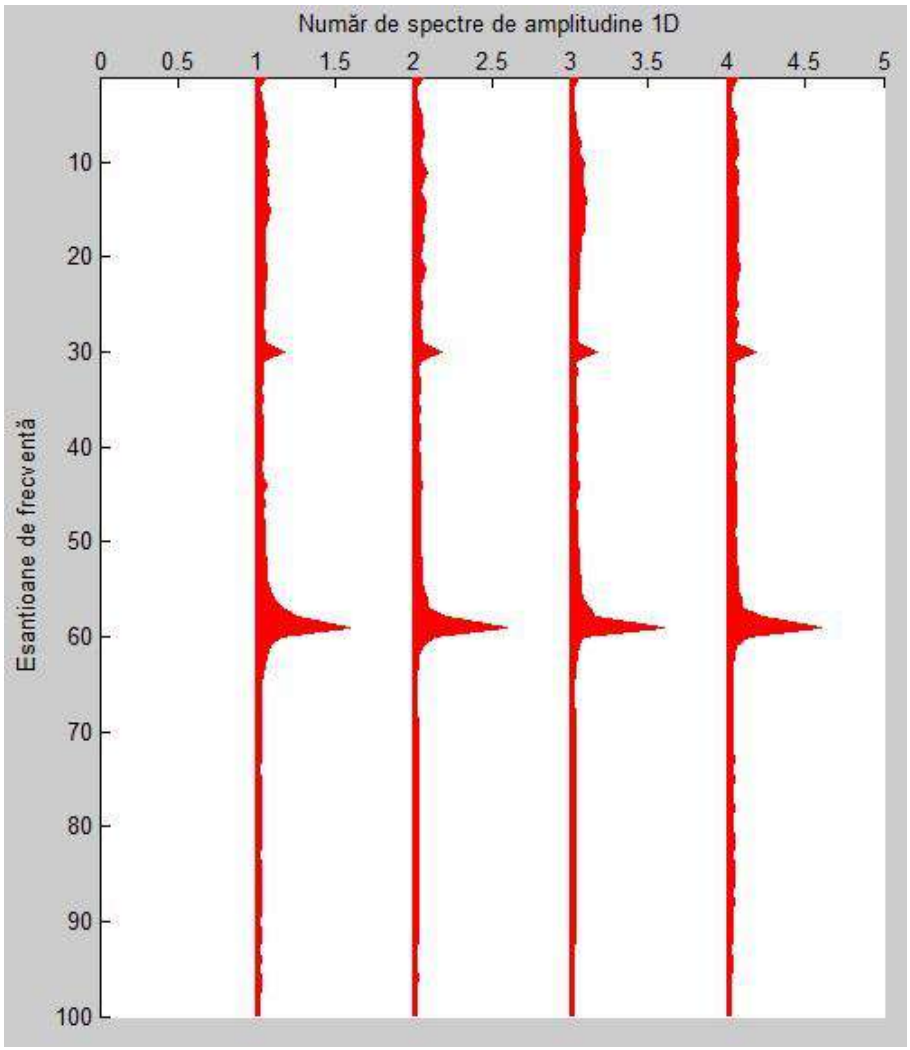

Figura 9.7: Primele 100 de eşantioane de frecvenţă selectate din spectrele de amplitudine 1D însumate pentru fiecare panou în parte

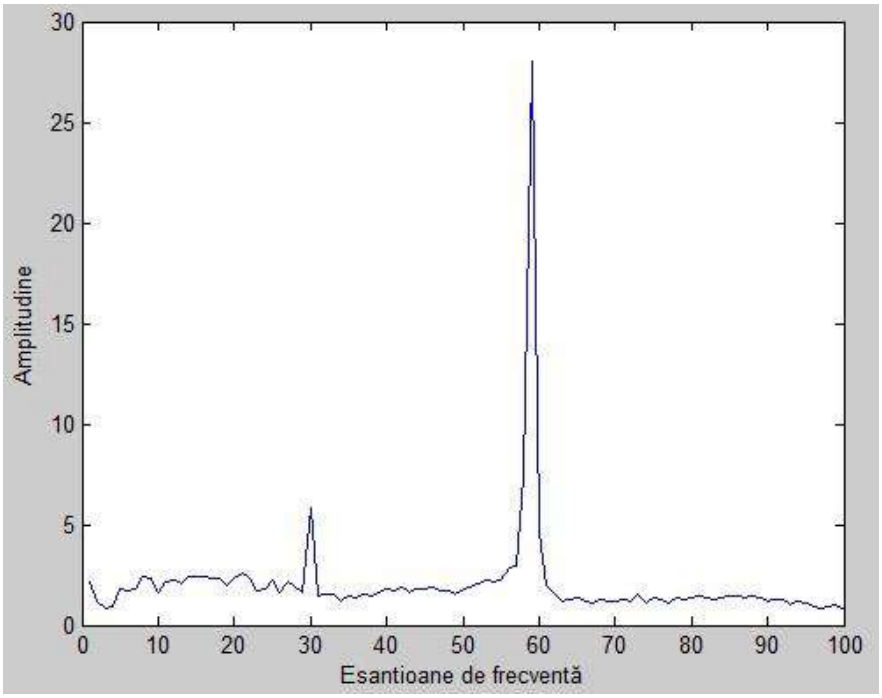

Figura 9.8: Primele 100 de eşantioane de frecvenţă selectate din spectrul de amplitudine obținut prin aplicarea directă a TF 1D pe trasele din panoul `panou_4.sgy` și a însumării spectrelor

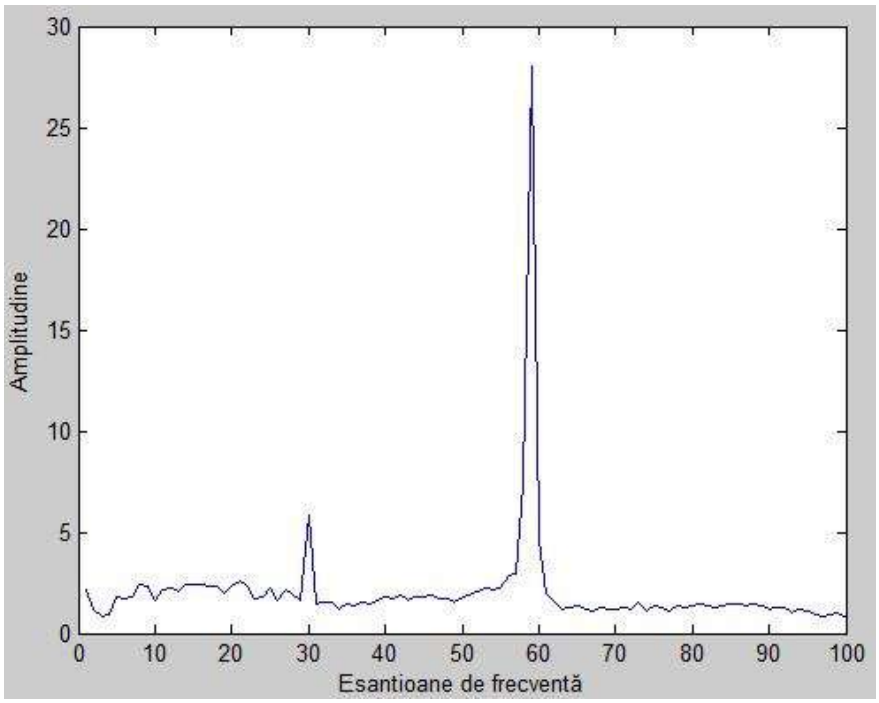

Figura 9.9: Primele 100 de eşantioane de frecvenţă selectate din spectrul de amplitudine corespunzător spectrului din Figura 9.8 dar obținut după însumarea automată a spectrelor pentru panoul `panou_4.sgy`

Bibliografie

Alsadi, H.N., 2016, Seismic hzdrocarbon exploration, Springer, p. 331.

Bensen, G.D., Ritzwoller, M.H., Barmin, M.P., Levshin, A.L., Lin, F., Moschetti, M.P., Shapiro, N.M., and Yang, Y., 2007, Processing seismic ambient noise data to obtain reliable broad-band surface waves dispersion measurements, Geophys. J. Int., doi:10.1111/j.1365-246X.2007.03374.x.

Box, G.E.P., and Jenkins, G., 1976, Time Series AbalysisȘ Forecasting and Control, Holden-Day.

Frigo, M., and Johnson, S.G., 1998, FFTW: An adaptive software architecture for the FFT, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, 1381-1384.

Levin, F.K., 1989, The effect of geophone arrays on random noise, Geophysics, 54(11), 1466-1473.

Margrave, G.F., 2003, Numerical methods of exploration seismology with algortihms in MATLAB, Lecture notes, University of Calgary, p. 225.

Panea, I., and Drijkoningen, G.G., 2008, The spatial data-adaptive minimum-variance distortionless-response beamformer on eismic single-sensor data, Geophysics, 73(5), Q29-Q42.

Panea, I., Draganov, D., Almagro Vidal, C., and Mocanu, V., 2014, Retrieval of reflections from ambient noise recorded in the Mizil area, Romania, Geophysics, 79(3), Q31-Q42.

Panea, I., and Bugheanu, D., 2017, Analysis of crooked-line 2D seismic reflection data recorded in areas with complex surface and subsurface conditions, Exploration Geophysics, 48(4), 493-503.

van der Burg, D.W., Drijkoningen, G.G., and Margrave, G.F., 2006, Delft internet-based introduction course on reflection processing with Matlab, First Break, 24(8), 57-59.

https://www.crewes.org/ResearchLinks/FreeSoftware/

https://www.mathworks.com/help/matlab/

https://en.wikipedia.org/wiki/Cross-correlation

http://wiki.seg.org