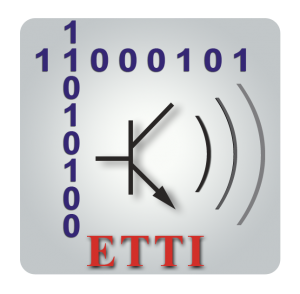
UNIVERSITATEA POLITEHNICA din BUCUREȘTI

**Facultatea de Electronică, Telecomunicații și Tehnologia Informației**

Proiect 2

**Clasificarea Semnalelor EMG**

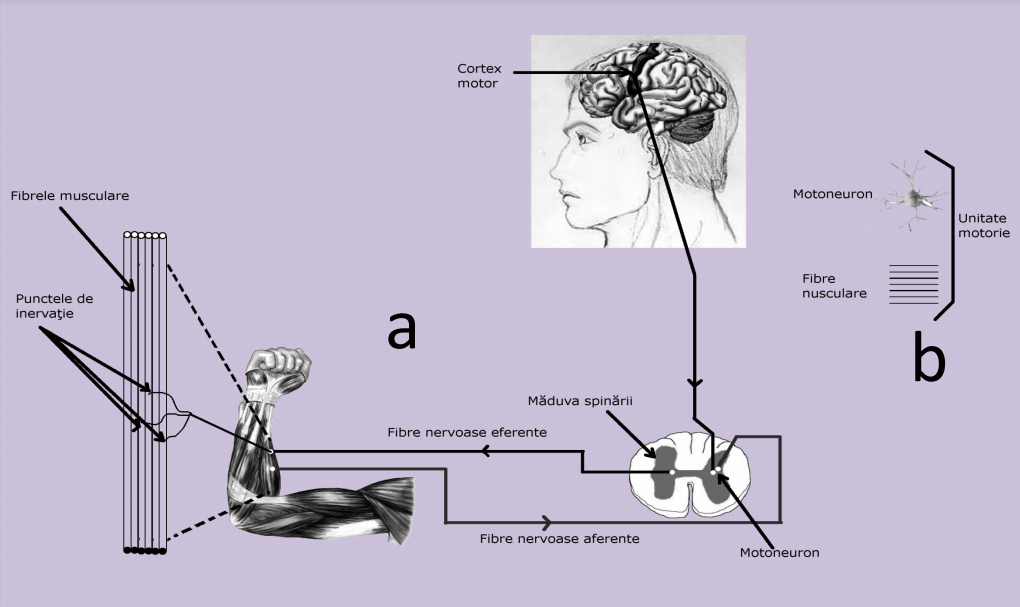
**Nume și Prenume: Vlad Alexandru | Pașalău Răzvan**

**Grupa: 434Ab**

**Profesor coordonator: Ovidiu Grigore**

**Introducere**

Semnalul electromiografic (EMG) reprezintă manifestarea electrică a activităţii musculare, aşa cum este aceasta regăsită la nivelul ţesutului muscular (EMG intramuscular) sau la nivelul pielii (EMG de suprafaţă). Semnalele EMG sunt folosite în aplicaţii de diagnosticare a unor miopatii şi neuropatii, dar şi ca semnale de comandă în cadrul protezelor de membru superior. Algoritmi complecşi pot transforma semnalul achiziţionat, de la muşchii rămaşi în urma unor amputaţii sau de la alţi muşchi din corp, în comenzi pentru proteze cu multe grade de libertate care să înlocuiască, cât mai bine posibil, membrul amputat.



Modul de generare a semnalului EMG (a): Iniţiată la nivelul cortexului motor, intenţia de mişcare este comunicată braţului prin intermediul unor impulsuri electrice care, odată ajunse la muşchi, comandă contracţia acestora şi în consecinţă efectuarea mişcării. (b) Motoneuronul care mediază o astfel de conexiune, alături de fibrele musculare pe care le inervează poartă numele de unitate motorie

**Tipuri de boli detectate de program**

**Miopatiile:**

Miopatiile (boli musculare) se prezinta ca boli neurologice care se manifesta prin simptome pur motorii (scadere de forta musculara, oboseala, dureri musculare, atrofii musculare).  Majoritatea miopatiilor se prezinta ca fiind simetrice si predominant proximale. Debutul poate fi acut sau cronic (in functie de tipul miopatiei).

Clasificarea miopatiilor se face cel mai frecvent dupa tipul procesului patologic care a determinat miopatia:

* distrofii musculare (boli musculare mostenite caracterizate prin slabiciune musculara progresiva aparuta la copil sau adolescent): distrofia musculara Duchenne, distrofia musculara Becker, distrofia musculara Emery-Dreifuss, distrofia musculara oculofaringiana, distrofiile musculare de centuri
* miopatii inflamatorii (miozite): polimiozita, dermatomiozita, miozita cu corpi de incluzie
* miopatii infectioase: trichineloza, toxoplasmoza, cisticercoza, HIV, infectii cu virusuri gripale, boala Lyme
* miopatii toxice si medicamentoase (alcool, steroizi, statine, Colchicina, Amiodarona)
* miopatii metabolice (aparute in deficite enzimatice mostenite): cea mai comuna este boala McArdle
* miopatii endocrine (in boli tiroidiene, ale glandelor suprarenale sau de hipofiza)
* miopatii congenitale
* miopatii asociate cu paralizii periodice
* miotoniile

**Neuropatiile:**

Neuropatiile sunt clasificate astfel:

• **Mononeuropatii**: Sunt afectiuni ale nervilor periferici limitate la un singur nerv. Cea mai frecventa mononeuropatie este cea de nerv median prin incarcerarea lui la nivelul canalului carpian (sindromul de canal carpian), urmata de mononeuropatia de nerv peronier prin incarcerarea lui la nivelul capului fibular. Mononeuropatiile pot fi leziuni difuze sau localizate in functie de mecanismul de producere.

• **Mononeuropatii** **multiplex**: Sunt afectiuni asimetrice ale mai multor nervi (spinali sau cranieni) in diferite grade de evolutie, de cele mai multe ori determinate de vasculite (poliarterita nodoasa, sindromul Churg-Strauss, lupusul eritematos sistemic, granulomatoza Wegener, boala cronica hepatica activa, artrita reumatoida), dar si de diabet zaharat, infectii (boala Lyme, infectia HIV), incarcerari multiple (dobandite sau ereditare), infiltratii neoplazice (limfomul si leucemia). Netratate sau nediagnosticate la timp, ajung sa aiba un tablou clinic similar unei polineuropatii.

• **Polineuropatii**: Sunt afectiuni generalizate ale nervilor periferici (difuze sau multifocale, simetrice sau asimetrice) ce pot fi de natura demielinizanta sau axonala (de cele mai multe ori mecanismul este mixt). Tabloul clinic poate fi mai mult senzitiv sau mai mult motor, dar cele mai multe au un aspect mixt senzitivo-motor. Au zeci poate chiar sute de cauze posibile (boli metabolice, boli autoimune, toxice, infectii, neoplazii etc) si rolul investigatiei EMG este de a determina un aspect electrofiziologic cat mai exact al polineuropatiei care orienteaza catre posibilele etiologii. Nu vom mentiona decat ca cea mai frecventa etiologie a polineuropatiei o reprezinta de departe diabetul zaharat.

**Metode folosite pentru abordarea temei**

Pentru a rezolva problema, am parcurs următorii pași:

1. Am descarcat fișierele *emg\_healthy.txt*, *emg\_myopathy.txt*, *emg\_neuropathy.txt* de pe site-ul [*https://phyisionet.org*](https://phyisionet.org).
2. Cu ajutorul programului *Matlab*, am vizualizat cele 3 tipuri de semnale, energiile pe benzi (Transformata Fourier Discreta).
3. Cu ajutorul programului Visual Studio, unde am implementat algoritmul, am deschis toate cele 3 fișiere și am extras doar valorile.
4. Am parcurs semnalul segmentându-l automat cu ajutorul instrucțiunii *for*. Numărul si lungimea segmentului pot fi date de către utilizator.

În continuare, am folosit următorii algoritmi:

1. **Transformata Fourier (discretă)** – cu ajutorul acestei funcții, am calculat energiile pe benzi ale segmentelor extrase
2. **void** FFT(**const** unsigned **int** N, vector<**double**> values, vector<**double**> &values\_fft)
3. {
5. **for** (unsigned **int** k = 0; k < N; k++)
6. {
7. complex<**double**> sum(0.0, 0.0);
8. **for** (unsigned **int** j = 0; j < N; j++)
9. {
10. **int** integers = -2 \* j \* k;
11. complex<**double**> exponent(0.0, PI / N \* (**double**)integers);
12. sum += values[j] \* exp(exponent);
13. }
14. values\_fft.push\_back(abs(sum) / N);
15. }
16. }
17. **RMS (media pătratică)** – această funție calculează radicalul valorii medii a energiilor pe benzi ale fiecărui segment.
18. **double** RMS(vector<**double**> v)
19. {
20. **double** sum = 0;
22. **for** (unsigned i = 0; i < v.size(); i++)
23. {
24. sum = sum + (v[i] \* v[i]);
25. }
27. **return** sqrt(sum / v.size());
28. }
29. **KNN (k-nearest neighbour)** – cu ajutorul acestei funcții, decidem dacă semnalul introdus reprezintă semnalul unei persoane bolnave sau sănătoase. Acest algoritm calculează distanțele dintre semnalul de test (cel introdus) și semnalele din baza de date (cele cunoscute deja). Funcția afișează diagnosticul in funcție de numărul de puncte cele mai apropiate de tipul de semnal. La finalul algoritmului se face o comparație între numărul de puncte, iar cel mai mare scor va reprezenta starea pacientului (dacă este sănătos, are miopatie sau neuropatie).
30. unsigned KNN(Point arrh[], Point arrm[], Point arrn[], unsigned n, Point p)//aici dau ca parametrii vectori ce contin valori pt fiecare tip
31. {                                                                          //tip de bola si un vector de test pentru care voi primi un rezultat
32. **for** (unsigned i = 0; i < n; i++)
33. {//am calculat distantele pctului de test fata de fiecare tip de punct
34. arrh[i].dist = sqrt((arrh[i].x - p.x) \* (arrh[i].x - p.x) + (arrh[i].y - p.y) \* (arrh[i].y - p.y));
35. arrm[i].dist = sqrt((arrm[i].x - p.x) \* (arrm[i].x - p.x) + (arrm[i].y - p.y) \* (arrm[i].y - p.y));
36. arrn[i].dist = sqrt((arrn[i].x - p.x) \* (arrn[i].x - p.x) + (arrn[i].y - p.y) \* (arrn[i].y - p.y));
37. }//in fiecare tip de boala am salvat distanta fata de pctul de test

40. unsigned freq1 = 0;
41. unsigned freq2 = 0;
42. unsigned freq3 = 0;
43. //am calculat distanta minima iar apoi am numarat cate din pctele de test erau mai apropiate de unul din simptome
44. **for** (unsigned i = 0; i < n; i++)
45. {
46. **double** min = 1000;
47. **if** (arrh[i].dist < min)
48. min = arrh[i].dist;
49. **if** (arrm[i].dist < min)
50. min = arrm[i].dist;
51. **if** (arrn[i].dist < min)
52. min = arrn[i].dist;
54. **if** (arrh[i].dist == min)
55. freq1++;
56. **if** (arrm[i].dist == min)
57. freq2++;
58. **if** (arrn[i].dist == min)
59. freq3++;
60. }
62. **int** max = 0;
63. **if** (freq1 > max)
64. max = freq1;
65. **if** (freq2 > max)
66. max = freq2;
67. **if** (freq3 > max)
68. max = freq3;
70. //tipul de simptom cel mai des intalnit ne da verdictul final si boala de care sufera pacientul
71. **if** (max == freq1)
72. {
73. **return** 1;
74. }
75. **if** (max == freq2)
76. {
77. **return** 2;
78. }
79. **if** (max == freq3)
80. {
81. **return** 3;
82. }
84. }

**Codul complet al programului (C++):**

1. #include <iostream>
2. #include <string>
3. #include <cmath>
4. #include <stdlib.h>
5. #include <fstream>
6. #include <complex>
7. #include <algorithm>
8. #include <vector>
9. #include <regex>
11. **using** **namespace** std;
13. #define PI 3.14159265359
15. unsigned KNN(Point arrh[], Point arrm[], Point arrn[], unsigned n, Point p);
16. **double** RMS(vector<**double**> v);
17. **void** FFT(**const** unsigned **int** N, vector<**double**> values, vector<**double**>& values\_fft);
19. **struct** Point//structura ce contine coordonatele si distantele fiecarui punct;
20. {
21. **double** x, y;
22. **double** dist;
23. };

26. unsigned KNN(Point arrh[], Point arrm[], Point arrn[], unsigned n, Point p)//aici dau ca parametrii vectori ce contin valori pt fiecare tip
27. {                                                                          //tip de bola si un vector de test pentru care voi primi un rezultat
28. **for** (unsigned i = 0; i < n; i++)
29. {//am calculat distantele pctului de test fata de fiecare tip de punct
30. arrh[i].dist = sqrt((arrh[i].x - p.x) \* (arrh[i].x - p.x) + (arrh[i].y - p.y) \* (arrh[i].y - p.y));
31. arrm[i].dist = sqrt((arrm[i].x - p.x) \* (arrm[i].x - p.x) + (arrm[i].y - p.y) \* (arrm[i].y - p.y));
32. arrn[i].dist = sqrt((arrn[i].x - p.x) \* (arrn[i].x - p.x) + (arrn[i].y - p.y) \* (arrn[i].y - p.y));
33. }//in fiecare tip de boala am salvat distanta fata de pctul de test

36. unsigned freq1 = 0;
37. unsigned freq2 = 0;
38. unsigned freq3 = 0;
39. //am calculat distanta minima iar apoi am numarat cate din pctele de test erau mai apropiate de unul din simptome
40. **for** (unsigned i = 0; i < n; i++)
41. {
42. **double** min = 1000;
43. **if** (arrh[i].dist < min)
44. min = arrh[i].dist;
45. **if** (arrm[i].dist < min)
46. min = arrm[i].dist;
47. **if** (arrn[i].dist < min)
48. min = arrn[i].dist;
50. **if** (arrh[i].dist == min)
51. freq1++;
52. **if** (arrm[i].dist == min)
53. freq2++;
54. **if** (arrn[i].dist == min)
55. freq3++;
56. }
58. **int** max = 0;
59. **if** (freq1 > max)
60. max = freq1;
61. **if** (freq2 > max)
62. max = freq2;
63. **if** (freq3 > max)
64. max = freq3;
66. //tipul de simptom cel mai des intalnit ne da verdictul final si boala de care sufera pacientul
67. **if** (max == freq1)
68. {
69. **return** 1;
70. }
71. **if** (max == freq2)
72. {
73. **return** 2;
74. }
75. **if** (max == freq3)
76. {
77. **return** 3;
78. }
80. }
82. **double** RMS(vector<**double**> v)
83. {
84. **double** sum = 0;
86. **for** (unsigned i = 0; i < v.size(); i++)
87. {
88. sum = sum + (v[i] \* v[i]);
89. }
91. **return** sqrt(sum / v.size());
92. }
94. **void** FFT(**const** unsigned **int** N, vector<**double**> values, vector<**double**> &values\_fft)
95. {
97. **for** (unsigned **int** k = 0; k < N; k++)
98. {
99. complex<**double**> sum(0.0, 0.0);
100. **for** (unsigned **int** j = 0; j < N; j++)
101. {
102. **int** integers = -2 \* j \* k;
103. complex<**double**> exponent(0.0, PI / N \* (**double**)integers);
104. sum += values[j] \* exp(exponent);
105. }
106. values\_fft.push\_back(abs(sum) / N);
107. }
108. }
110. **int** main()
111. {
112. unsigned nr = 0;
113. ifstream myfile;
114. string line;
115. vector<string> valori\_healthy, valori\_myopathy, valori\_neuropathy;
116. string x;
117. vector<**double**> val\_healthy, val\_myopathy, val\_neuropathy, rms\_healthy, rms\_myopathy, rms\_neuropathy, val\_fft;
118. **int** a;
120. //Valorile semnalului pacientului sanatus
121. myfile.open("emg\_healthy.txt");
123. **while** (myfile >> line)
124. {
125. **if** (nr % 2 == 1)
126. valori\_healthy.push\_back(line);
127. nr++;
128. }
130. myfile.close();
132. vector<**double**> val\_double\_healthy(valori\_healthy.size());
134. transform(valori\_healthy.begin(), valori\_healthy.end(), val\_double\_healthy.begin(),
135. [](string **const**& val) {**return** stod(val); });
137. a = 0;
138. **for** (**int** i = 0; i < val\_double\_healthy.size() / 35; i++)
139. {
140. **for** (**int** j = a; j < a + 35; j++)
141. {
142. val\_healthy.push\_back(val\_double\_healthy[j]);
143. }
144. FFT(35, val\_healthy, val\_fft);
145. rms\_healthy.push\_back(RMS(val\_fft));
146. val\_fft.clear();
147. a += 15;
148. }
150. myfile.open("emg\_myopathy.txt");
152. **while** (myfile >> line)
153. {
154. **if** (nr % 2 == 1)
155. valori\_myopathy.push\_back(line);
156. nr++;
157. }
159. myfile.close();
161. vector<**double**> val\_double\_myopathy(valori\_myopathy.size());
163. transform(valori\_myopathy.begin(), valori\_myopathy.end(), val\_double\_myopathy.begin(),
164. [](string **const**& val) {**return** stod(val); });

167. a = 0;
168. **for** (**int** i = 0; i < val\_double\_myopathy.size() / 35; i++)
169. {
170. **for** (**int** j = a; j < a + 35; j++)
171. {
172. val\_myopathy.push\_back(val\_double\_myopathy[j]);
173. }
174. FFT(35, val\_myopathy, val\_fft);
175. rms\_myopathy.push\_back(RMS(val\_fft));
176. val\_fft.clear();
177. a += 15;
178. }
180. myfile.open("emg\_neuropathy.txt");
182. **while** (myfile >> line)
183. {
184. **if** (nr % 2 == 0)
185. valori\_neuropathy.push\_back(line);
186. nr++;
187. }
189. myfile.close();
191. vector<**double**> val\_double\_neuropathy(valori\_neuropathy.size());
193. transform(valori\_neuropathy.begin(), valori\_neuropathy.end(), val\_double\_neuropathy.begin(),
194. [](string **const**& val) {**return** stod(val); });

197. a = 0;
198. **for** (**int** i = 0; i < val\_double\_neuropathy.size() / 35; i++)
199. {
200. a = 0;
201. **for** (**int** j = a; j < a + 35; j++)
202. {
203. val\_neuropathy.push\_back(val\_double\_neuropathy[j]);
204. }
205. FFT(35, val\_neuropathy, val\_fft);
206. rms\_neuropathy.push\_back(RMS(val\_fft));
207. val\_fft.clear();
208. a += 15;
209. }
211. Point vect1[35];
212. Point vect2[35];
213. Point vect3[35];
215. **for** (unsigned i = 0; i < 35; i++)
216. {
217. vect1[i].x = i;
218. vect1[i].y = rms\_healthy[i];
220. vect2[i].x = i;
221. vect2[i].y = rms\_myopathy[i];
223. vect3[i].x = i;
224. vect3[i].y = rms\_neuropathy[i];
226. }
228. **int** f = 10;
229. **int** stare\_pacient = KNN(vect1, vect2, vect3, f, vect1[2]); //ultimul parametru trebuie o coordonata de test pe baza caruia programul sa spuna de ce sufera pacientul
231. **if** (stare\_pacient == 1)
232. {
233. cout << "Sanatos" << endl;
234. }
235. **if** (stare\_pacient == 2)
236. {
237. cout << "Miopatie" << endl;
238. }
239. **if** (stare\_pacient == 3)
240. {
241. cout << "Neuropatie" << endl;
242. }
244. system("pause");
245. **return** 0;
246. }