

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**gr. IA-231, Chistol Maxim**

**Raport**

**pentru lucrarea de laborator Nr.1**

***la cursul de “Metode Numerice”***

Verificat:

**Vasile Moraru,** *conf. univ. dr.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău 2024**

**Tema:**

Rezolvarea numerică a ecuațiilor algebrice și transcendente

Scopul lucrării:

1) Să se separe toate rădăcinile reale ale ecuației f(x)=0 unde y=f(x) este o funcție reală de variabilă reală.

2) Să se determine o rădăcină reală a ecuaţiei date cu ajutorul metodei înjumătăţirii intervalului cu o eroare mai mică decât ε=10-2 .

3) Să se precizeze rădăcina obţinută cu exactitatea ε=10-6 utilizând

- metoda aproximațiilor succesive

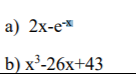
- metoda tangentelor (Newton)

- metoda secantelor.

4) Să se compare rezultatele luând în considerație numărul de iterații, evaluările pentru funcția şi derivată.

# Sarcina Individual.

**Varianta 6**



**Rezolvare:**

1. **Separarea rădăcinilor**

*Metoda grafică:*

1. f(x) = 2x-

Pentru efectuarea acestei metode este necesar de divizat f(x) = 2x-

2 funcții cu relația y(x) = g(x),deci: y(x) = g(x) => y(x) = g(x),2x =

y(x) = 2x

g(x) =

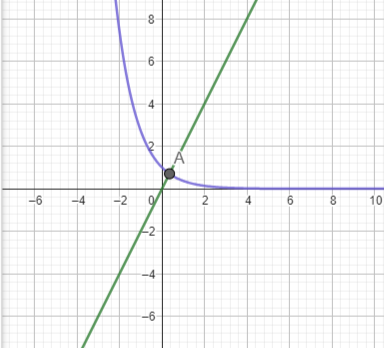
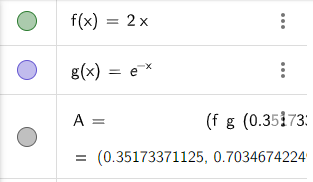


Figura 1 Reprezentarea grafică a functiei

A =(0.35; 0.7) doar un singur punct de intersecție,pe intervalul (0;1). Ecuația dată are doar **o soluție diferită de zero.**

Analizând funcțiile, deducem că rădăcina acesteia ia valori cuprinse în intervalul **[0, 1]**.

*Metoda șirului lui Rolle:*

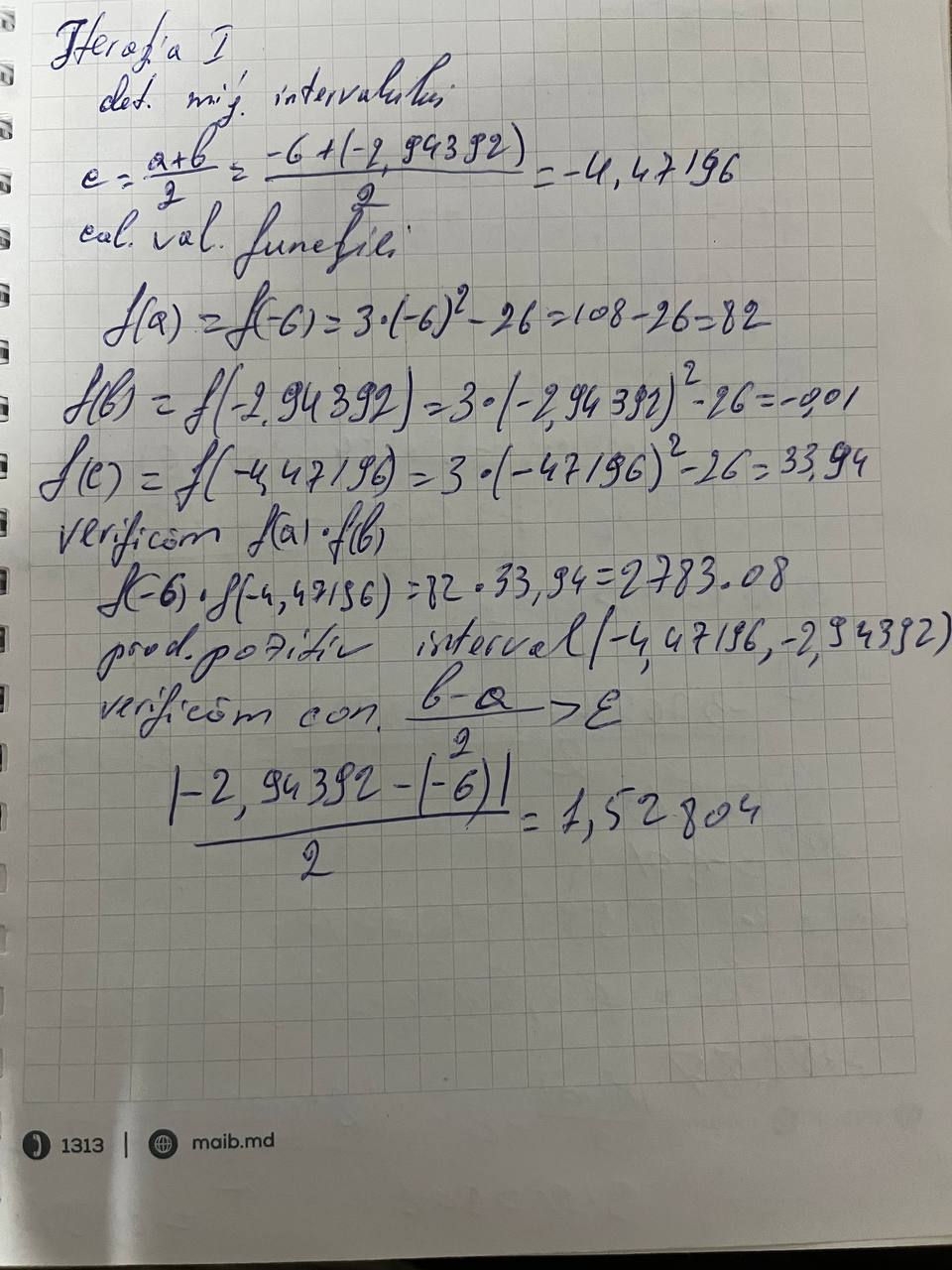
1. f(x) = -26x+43

Fiindcă ecuația respectivă este un polinom, separarea rădăcinilor se va efectua prin intermediul *metodei șirului lui Rolle*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | -6 | -2.94392 | 2.94392 | 4 |
|  | -17 | 94,02795 | -8,02795 | 3 |

Pe intervalul(-6;4) ecuatia are 3 radacini reale r->(-6;-2.94392), r->(-2.94392;2.94392) si r->(2.94392;4)

1. **Metoda înjumătățirii intervalului**



1. Fugura 2 Prima iteratie f(x) = -26x+43

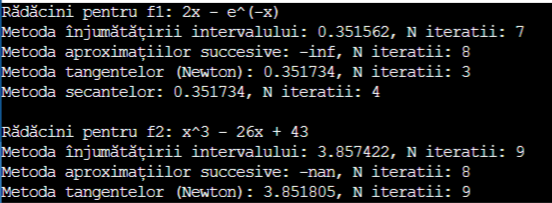


Figura 3 Outputul programului

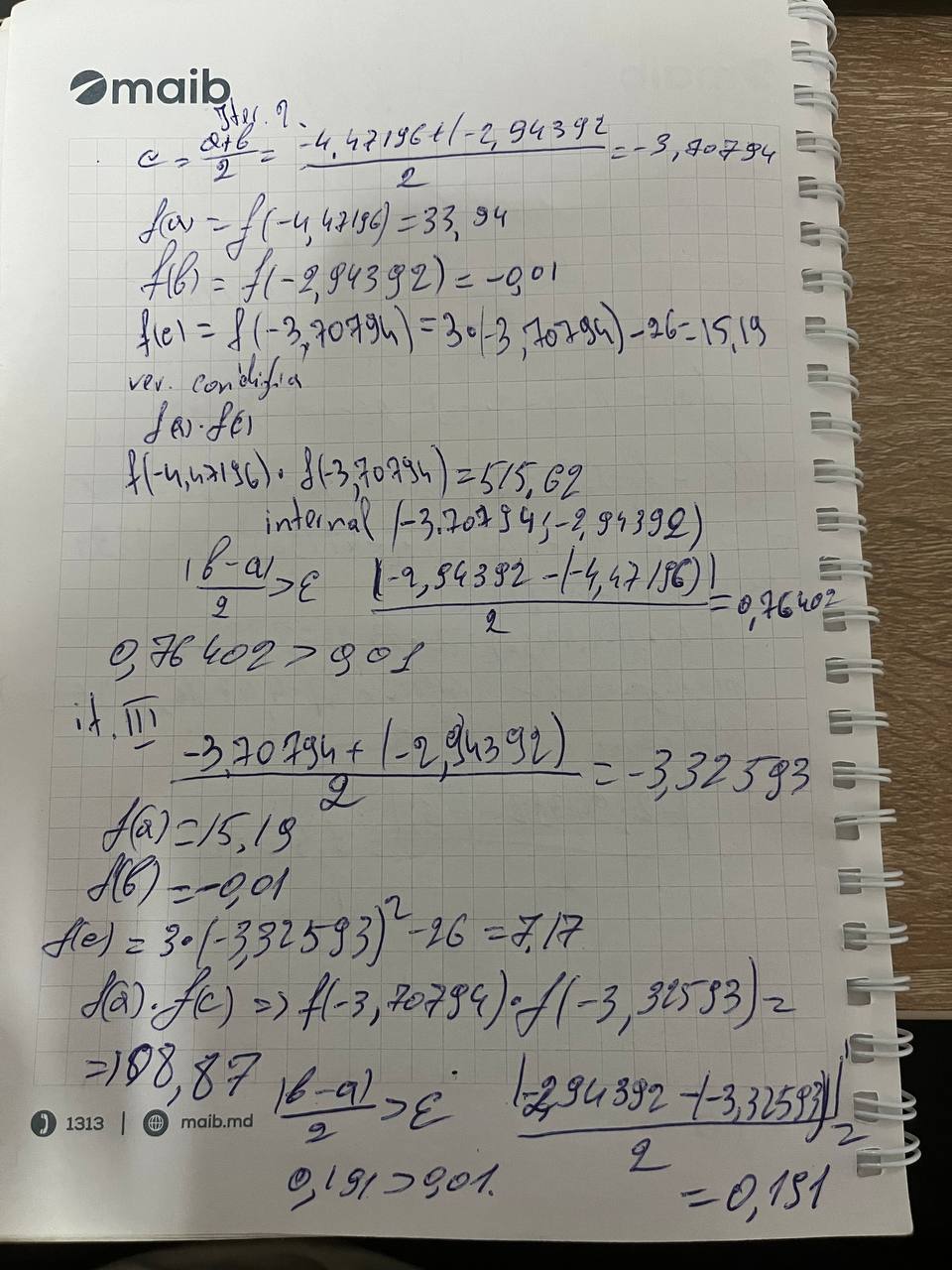
****

Figura 4 Iteratie 2 si 3 f(x) = -26x+43

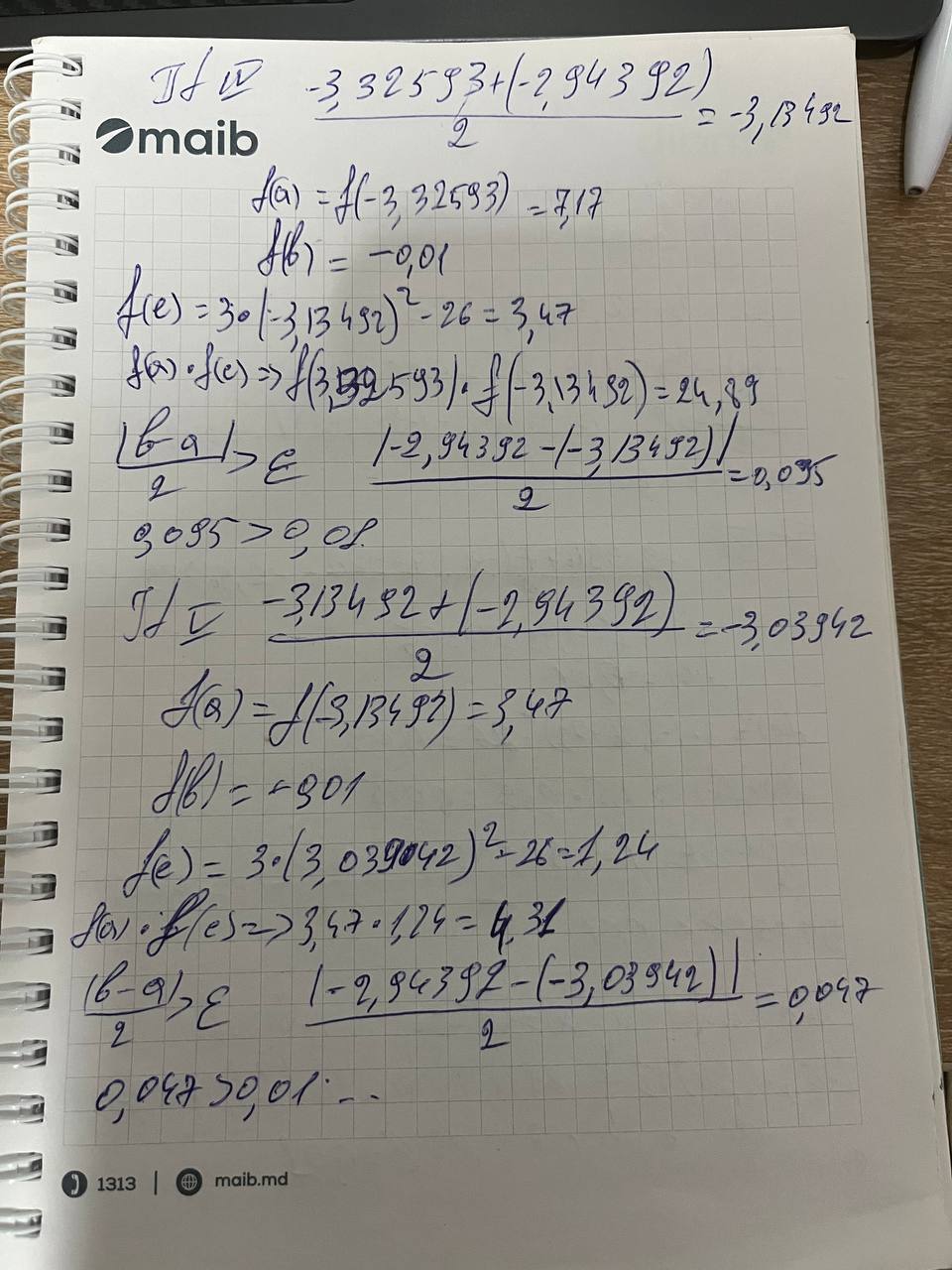
****

Figura 5 Iteratiile 4,5 f(x) = -26x+43

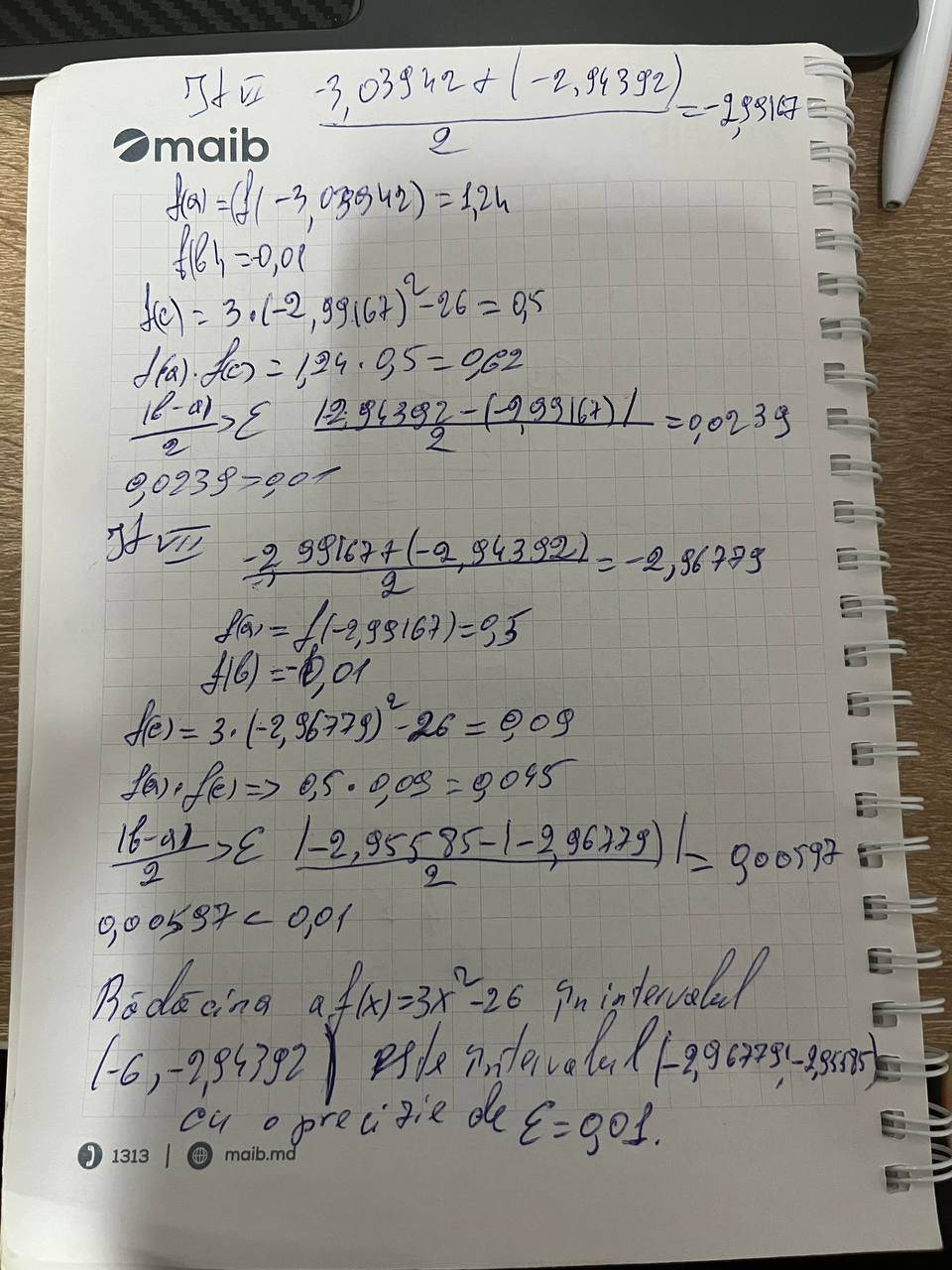
****

Figura 6 Iteratie 6,7 f(x) = -26x+43

1. **Metoda aproximațiilor succesive**

**a)**       f(x) = 2x-

Pentru a aplica această metodă, formăm o nouă funcție din conform relațiilor:

0 ⬄ =

,

Calculăm funcția și verificăm convergența funcției

b)      f(x) = -26x+43

Pentru a aplica această metodă, formăm o nouă funcție din conform relațiilor:

0 ⬄ =

,

Calculăm funcția și verificăm convergența funcției

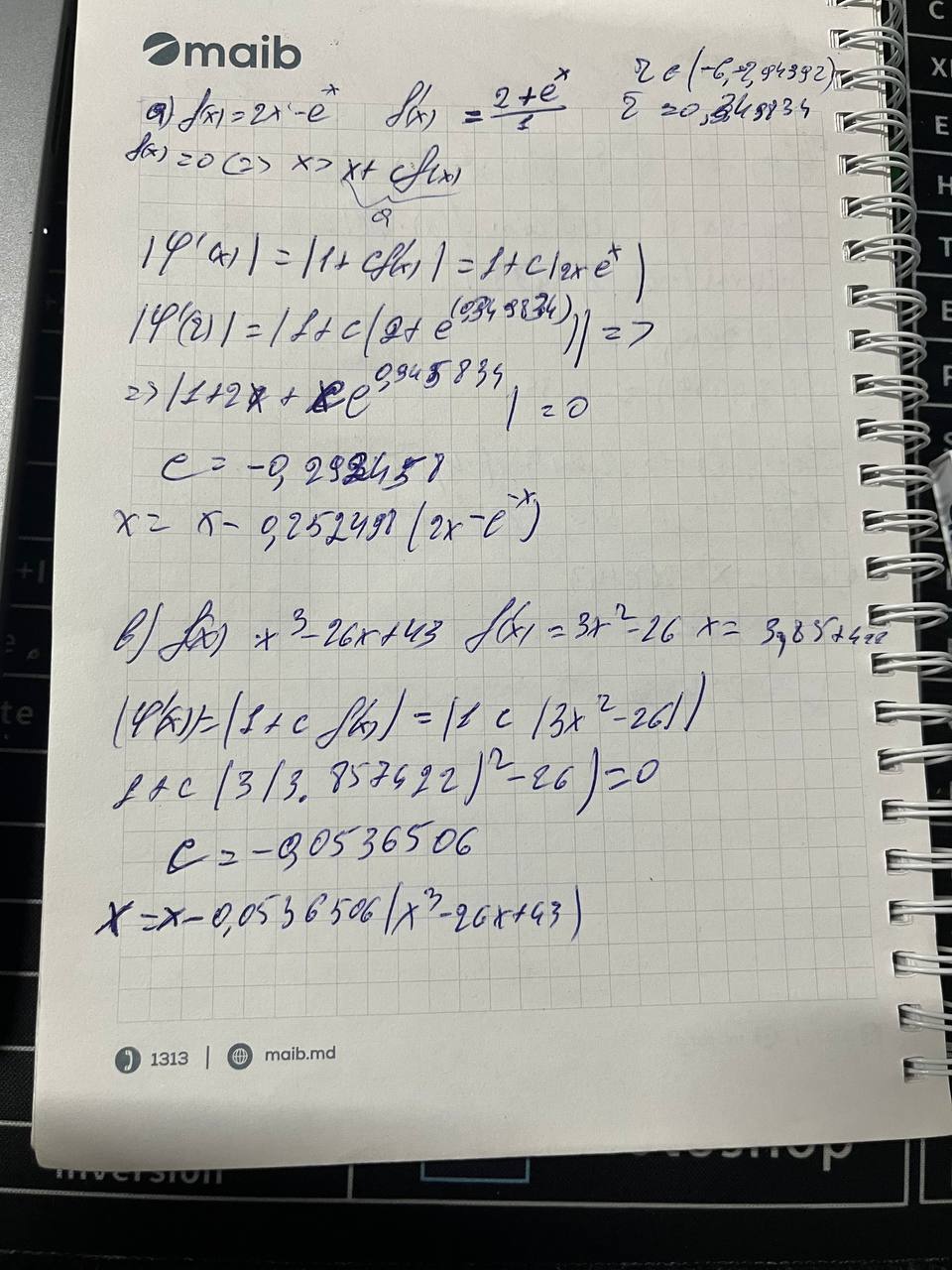


Figura 7 Metoda aproximațiilor succesive(a,b)

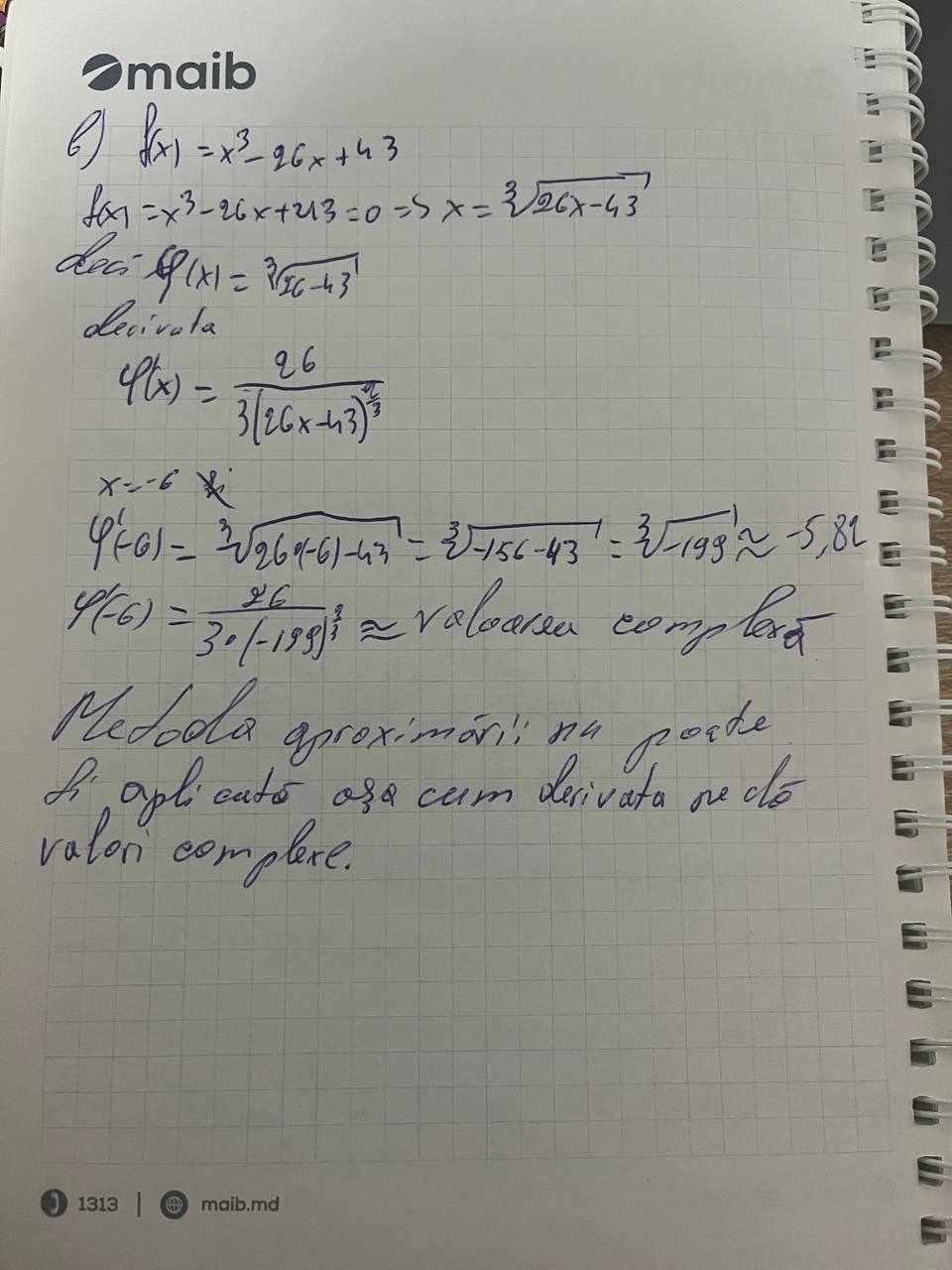
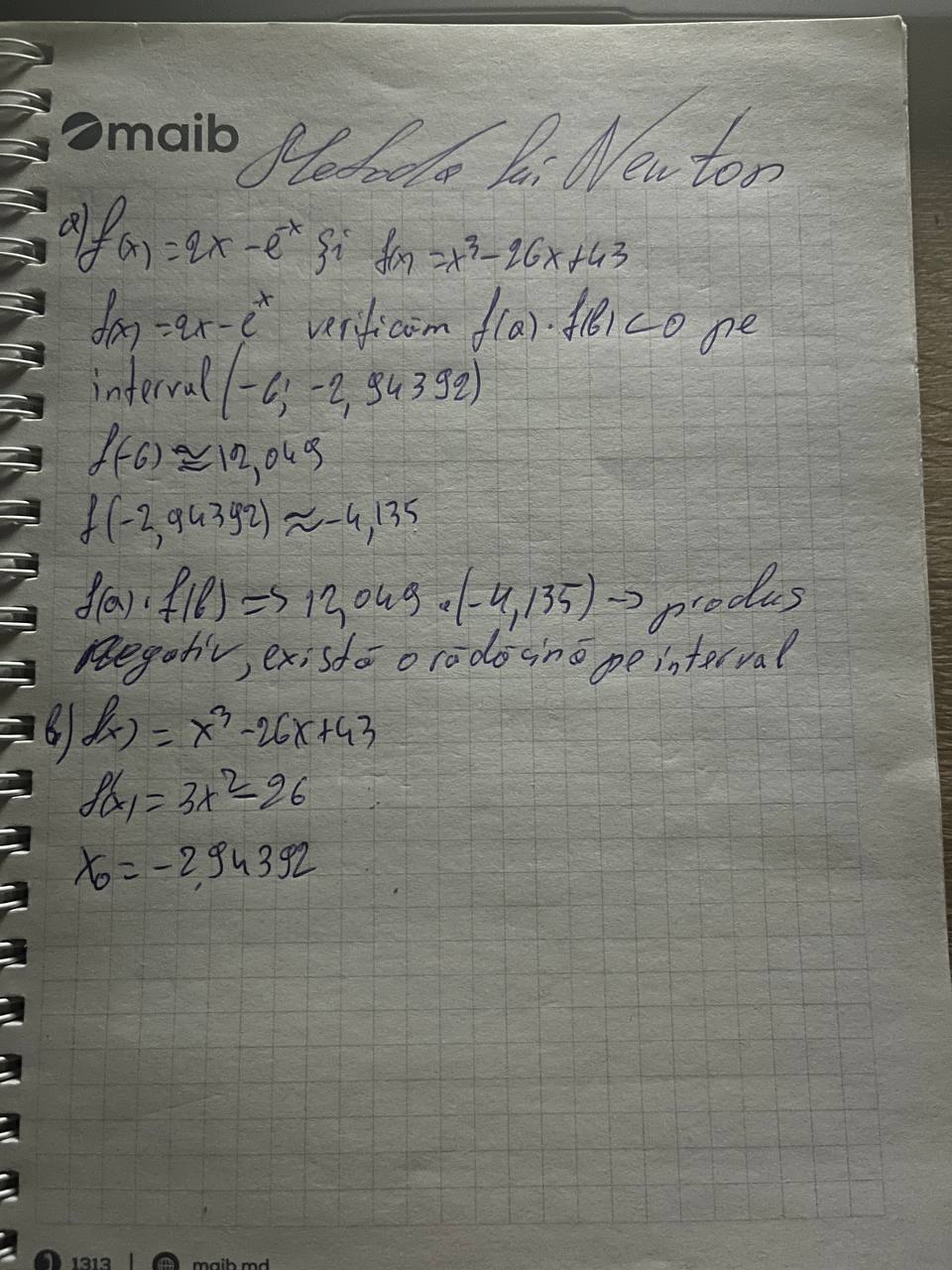


Figura 7 Metoda aproximațiilor succesive(b)

1. **Metoda Newton (tangentelor)**
2. f(x) = 2x-



figura

Metoda Newton aplicată funcțiilor din punctele inițiale în intervalul (−6,−2.94392) poate produce aproximări progresive ale rădăcinilor funcțiilor respective. În primul caz (pentru 2x−), metoda converge destul de repede, în timp ce în al doilea caz (pentru −26x+43), poate necesita ajustări ale punctului inițial.

1. **Metoda secantelor**

Metoda secantelor se aseamănă cu metoda tangentelor, diferențe fiind doar la formula generală de calcul.

**codul:**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

// Funcțiile pentru ecuațiile date

double f1(double x) {

return 2 \* x - exp(-x); // f(x) = 2x - e^(-x)

}

double f2(double x) {

return pow(x, 3) - 26 \* x + 43; // f(x) = x^3 - 26x + 43

}

// Derivata funcției pentru metoda Newton

double f1\_derivative(double x) {

return 2 + exp(-x); // Derivata pentru f1

}

double f2\_derivative(double x) {

return 3 \* pow(x, 2) - 26; // Derivata pentru f2

}

// Metoda înjumătățirii intervalului

void bisection(double (\*func)(double), double a, double b, double epsilon) {

double c;

int k = 0;

while ((b - a) > epsilon) {

k++;

c = a + (b - a) / 2.0;

if (func(c) == 0.0) break; // Dacă c este o rădăcină exactă

if (func(a) \* func(c) < 0)

b = c; // Rădăcina este în [a, c]

else

a = c; // Rădăcina este în [c, b]

}

printf("Metoda înjumătățirii intervalului: %lf, N iteratii: %d\n", c, k);

}

// Metoda aproximațiilor succesive

void successive\_approximations(double (\*func)(double), double initial, double epsilon) {

double x = initial;

double y;

int k = 0;

do {

k++;

y = x; // Salvează x vechi

x = func(x); // Aproximare succesivă

} while (fabs(x - y) >= epsilon);

printf("Metoda aproximațiilor succesive: %lf, N iteratii: %d\n", x, k);

}

// Metoda tangentelor (Newton)

void newton\_method(double (\*func)(double), double (\*derivative)(double), double initial, double epsilon) {

double x = initial;

double x1;

int k = 0;

do {

k++;

x1 = x - func(x) / derivative(x); // Formula lui Newton

if (fabs(x1 - x) < epsilon) break; // Verifică convergența

x = x1;

} while (1);

printf("Metoda tangentelor (Newton): %lf, N iteratii: %d\n", x1, k);

}

// Metoda secantelor

void secant\_method(double (\*func)(double), double x0, double x1, double epsilon) {

double x2;

int k = 0;

do {

k++;

x2 = x1 - func(x1) \* (x1 - x0) / (func(x1) - func(x0)); // Formula secantei

x0 = x1;

x1 = x2;

} while (fabs(func(x2)) >= epsilon);

printf("Metoda secantelor: %lf, N iteratii: %d\n", x2, k);

}

int main() {

double epsilon = 1e-2;

double epsilon\_exact = 1e-6;

// Bisecția pentru prima funcție

printf("Rădăcini pentru f1: 2x - e^(-x)\n");

bisection(f1, 0, 1, epsilon); // Intervalul ales [0, 1]

successive\_approximations(f1, 0.5, epsilon\_exact); // Aproximare inițială în interval

newton\_method(f1, f1\_derivative, 0.5, epsilon\_exact);

secant\_method(f1, 0.1, 0.9, epsilon\_exact); // x0 = 0.1, x1 = 0.9

// Bisecția pentru a doua funcție

printf("\nRădăcini pentru f2: x^3 - 26x + 43\n");

bisection(f2, -6, -2.94392, epsilon); // Intervalul ales [-6, -2.94392]

successive\_approximations(f2, -4.5, epsilon\_exact); // Aproximare inițială în interval

newton\_method(f2, f2\_derivative, -4.5, epsilon\_exact);

secant\_method(f2, -5.5, -3.5, epsilon\_exact); // x0 = -5.5, x1 = -3.5

return 0;

}

# Concluzii:

Concluzia acestei lucrări evidențiază atingerea obiectivelor propuse prin utilizarea și compararea metodelor numerice pentru determinarea rădăcinilor reale ale unei ecuații neliniare. Am demonstrat că fiecare metodă utilizată – fie că este vorba de metoda înjumătățirii intervalului, metoda aproximațiilor succesive, metoda tangentelor (Newton) sau metoda secantelor – prezintă caracteristici distincte, care le fac adecvate pentru diferite tipuri de probleme.

Metodele numerice sunt esențiale în situațiile în care soluțiile analitice sunt dificil de obținut sau inexistente. Fiecare metodă a permis obținerea unei soluții cu precizia dorită, confirmându-se astfel că aceste tehnici sunt utile și eficiente. De asemenea, au fost îndeplinite cerințele de eroare impuse, ceea ce ne permite să avem încredere în rezultatele obținute și să le aplicăm în contexte practice.

Prin analiza diferitelor tehnici, lucrarea a oferit o înțelegere clară a compromisurilor între viteza de convergență și complexitatea calculului. În funcție de necesitățile fiecărui caz – fie că vorbim de precizie ridicată, viteză de calcul sau resurse computaționale limitate – se poate alege metoda cea mai potrivită. Astfel, am reușit să evidențiem importanța cunoașterii și aplicării corecte a acestor metode numerice în rezolvarea ecuațiilor neliniare, oferind o perspectivă practică pentru soluționarea unor probleme reale.