

(curs 1-51)

1. Fundamente

Operatiile aritmetice fundamentale : adunarea si inmultirea.
Sunt necesare pentru a evalua un polinom $P(x)$ intr-un punct x .

1.1. Evaluarea unui polinom:

$$P(x) = 2x^4 + 3x^3 - 3x^2 + 5x - 1, \quad x = \frac{1}{2}$$

I. Metoda directă

$$P\left(\frac{1}{2}\right) = 2 * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} + 3 * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} - 3 * \frac{1}{2} * \frac{1}{2} + 5 * \frac{1}{2} - 1 = \frac{5}{4}$$

Numarul de înmultiri necesare este 10, la care se adauga 4 adunari

II. Găsim puterile numărului $\frac{1}{2}$

$$\frac{1}{2} * \frac{1}{2} = \left(\frac{1}{2}\right)^2$$

$$\left(\frac{1}{2}\right)^2 * \frac{1}{2} = \left(\frac{1}{2}\right)^3$$

$$\left(\frac{1}{2}\right)^3 * \frac{1}{2} = \left(\frac{1}{2}\right)^4.$$

- acum, putem face o simplă însumare a termenilor:

$$P\left(\frac{1}{2}\right) = 2 * \left(\frac{1}{2}\right)^4 + 3 * \left(\frac{1}{2}\right)^3 - 3 * \left(\frac{1}{2}\right)^2 + 5 * \frac{1}{2} - 1 = \frac{5}{4}.$$

- avem 3 înmulțiri ale lui $1/2$, plus alte 4 înmulțiri
- am redus numărul total de înmulțiri la 7, cu aceleași 4 adunări
- este reducerea de la 14 la 11 operații o îmbunătățire semnificativă?
- dacă polinomul trebuie evaluat pentru diferite valori ale lui x , de mai multe ori pe secundă, atunci diferența poate fi esențială

III. Înmultirea imbricată (metoda lui Horner)

$$\begin{aligned} P(x) &= -1 + x(5 - 3x + 3x^2 + 2x^3) = \\ &= -1 + x(5 + x(-3 + 3x + 2x^2)) = \\ &= -1 + x(5 + x(-3 + x(3 + 2x))) \end{aligned}$$

Evaluarea se face dinăpreză interior spre exterior.

In general, un polinom de gradul d poate fi evaluat utilizând aceasta metoda folosind d înmulțiri și d adunări.

înmulțim $\frac{1}{2} * 2$,	adunăm $+ 3 \rightarrow 4$
înmulțim $\frac{1}{2} * 4$,	adunăm $- 3 \rightarrow -1$
înmulțim $\frac{1}{2} * -1$,	adunăm $+ 5 \rightarrow \frac{9}{2}$
înmulțim $\frac{1}{2} * \frac{9}{2}$,	adunăm $- 1 \rightarrow \frac{5}{4}$

Forma generală a unui polinom

$$c_1 + c_2 x + c_3 x^2 + c_4 x^3 + c_5 x^4 = c_1 + x(c_2 + x(c_3 + x(c_4 + x(c_5))))$$

Pentru interpolare, vom avea nevoie de forma:

$$c_1 + (x - r_1)(c_2 + (x - r_2)(c_3 + (x - r_3)(c_4 + (x - r_4)(c_5))))$$

unde r_1, r_2, r_3, r_4 - puncte de bază

Exemplu 1

- găsiți o metodă eficientă pentru evaluarea polinomului
 $P(x) = 4x^5 + 7x^8 - 3x^{11} + 2x^{14}$
- ideea este de a factoriza x^5 din fiecare termen și de a scrie polinomul în x^3 :

$$\begin{aligned} P(x) &= x^5(4 + 7x^3 - 3x^6 + 2x^9) \\ &= x^5 * (4 + x^3 * (7 + x^3 * (-3 + x^3 * (2)))). \end{aligned}$$

- pentru fiecare punct x , trebuie să calculăm mai întâi $x * x = x^2$, $x * x^2 = x^3$, și $x^2 * x^3 = x^5$
- aceste trei înmulțiri, împreună cu înmulțirea lui x^5 , și cele trei înmulțiri și trei adunări necesare evaluării polinomului de gradul 3 în x^3 dau numărul total de operații necesar evaluării acestui polinom, și anume 7 înmulțiri și 3 adunări

1.2. Numere binare

Un nr. în bază 2: ... $b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} \dots$

$$(4)_{10} = (100)_2$$

$$(\frac{3}{4})_{10} = (0.75)_{10} = (0.11)_2$$

1.2.1. Conversia din decimal în binar

$$(53.7)_{10} = (53)_{10} + (0.7)_{10}$$

Partea întreagă: $(53)_{10} = (110101)_2$

Partea fractionară:

0.7×2	$0.4 + 1$
0.4×2	$0.8 + 0$
0.8×2	$0.6 + 1$
0.6×2	$0.2 + 1$
0.2×2	$0.4 + 0$
0.4×2	$0.8 + 0$

Observăm că acest proces se repetă la infinit după 4 pași.

$$\Rightarrow (0.\overline{7})_{10} = (0.\underline{1011001100110\dots})_2 = (0.\overline{10110})_2$$

$$\Rightarrow (53.\overline{7})_{10} = (\underline{110101.10110})_2$$

1.2.2. Conversia din binar în decimal

$$(10101)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (21)_{10}$$

$$(0.\overline{1011})_2 = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2^2} + 1 \cdot \frac{1}{2^3} + 1 \cdot \frac{1}{2^4} = \left(\frac{11}{16}\right)_{10}$$

$$0 (0.\overline{1011})_2$$

Înmulțim nr. cu $2^4 \Rightarrow$ deplasare la stânga cu 4 pozitii

$$2^4 \cdot x = 1011.\overline{1011}$$

$$x = 0000.\overline{1011}$$

$$\Rightarrow x \cdot (2^4 - 1) = (1011)_2 = (11)_{10}$$

$$\Rightarrow x = (0.\overline{1011})_2 = \left(\frac{11}{16}\right)_{10}$$

$$0 x = (0.\overline{101})_2$$

$$y = 2^2 \cdot x = (10.\overline{101})_2$$

$$\{ y = (0.\overline{101})_2 = z$$

$$\Rightarrow 2^3 \cdot z = (101.\overline{101})_2$$

$$z = (000.\overline{101})_2$$

$$\Rightarrow z \cdot (2^3 - 1) = (101)_2 = 5 \Rightarrow z = \frac{5}{7}$$

$$\Rightarrow y = (10)_2 + \frac{5}{7} = 2 + \frac{5}{7} = \frac{19}{7}$$

$$\Rightarrow x = 2^{-2} \cdot y = \left(\frac{19}{28}\right)_{10}$$

1.3. Reprezentarea în virgula flotanta a numerelor reale

1.3.1. Formate de virgulă flotantă

Um număr în virgulă flotantă este format din trei parti:

- a) semnul (+ sau -)
 - b) mantisa (care conține și numărul de biti semnificativi)
 - c) exponentul

Există 3 nivele de precizie: I. simplă → 32
II. dublă → 64
III. extinsă (lungă) → 80

precizia	semnul	exponentul	mantisa
simplă	1	8	23
dublă	1	11	52
dublă lungă	1	15	64

Definiția 1

Numărul **epsilon mașină**, notat cu ϵ_{mach} , este distanța dintre 1 și cel mai mic număr în virgulă flotantă mai mare decât 1. Pentru standardul IEEE de virgulă flotantă în dublă precizie, avem că

$$\epsilon_{\text{mach}} = 2^{-52}.$$

- numărul zecimal $9.4 = (1001.0\overline{10})_2$ este aliniat la stânga astfel:

unde am pus într-un chenar primii 52 de biți ai mantisei

1.3.1 Formate de virgulă flotantă

Algoritmul 1 (Regula IEEE a rotunjirii la cea mai apropiată valoare)

Pentru dubla precizie, dacă al 53-lea bit de la dreapta virgulei binare este 0, atunci **rotunjim** în jos (trunchiem după bitul 52). Dacă bitul al 53-lea este 1, atunci rotunjim în sus (adunăm 1 la bitul 52), cu excepția cazului în care biții de după 1 sunt 0, caz în care 1 este adunat la bitul 52 dacă și numai dacă bitul 52 este 1.

- pentru numărul 9.4 discutat anterior, cel de-al 53-lea bit din dreapta virgulei binare este 1 și este urmat de alți biți care nu sunt toți zero
 - conform regulii rotunjirii la cea mai apropiată valoare, trebuie să facem o rotunjire în sus, adică să adunăm un 1 la bitul 52
 - prin urmare, numărul în virgulă flotantă care îl reprezintă pe 9.4 este

Definiția 2

Notăm numărul în virgulă flotantă în dublă precizie IEEE asociat lui x , folosind regula rotunjirii la cea mai apropiată valoare, cu $\text{fl}(x)$.

- în operațiile aritmetice realizate în calculator, numărul real x este înlocuit cu sirul de biți $\text{fl}(x)$
- conform definiției de mai sus, $\text{fl}(9.4)$ este numărul în reprezentare binară dat de (7)
- am ajuns la această reprezentare în virgulă flotantă înălțurând partea infinită dată de $0.\overline{1100} \times 2^{-52} \times 2^3 = 0.\overline{0110} \times 2^{-51} \times 2^3 = 0.4 \times 2^{-48}$ din capătul din dreapta al numărului, și apoi adunând $2^{-52} \times 2^3 = 2^{-49}$ în pasul de rotunjire
- prin urmare,

$$\begin{aligned}\text{fl}(9.4) &= 9.4 + 2^{-49} - 0.4 \times 2^{-48} \\ &= 9.4 + (1 - 0.8)2^{-49} \\ &= 9.4 + 0.2 \times 2^{-49}. \end{aligned} \quad (8)$$

- cu alte cuvinte, un calculator care folosește reprezentarea în dublă precizie și regula rotunjirii la cea mai apropiată valoare face o eroare de 0.2×2^{-49} atunci când stochează numărul 9.4
- vom spune că 0.2×2^{-49} este **eroarea de rotunjire**
- ceea ce trebuie reținut este faptul că numărul în virgulă flotantă care îl reprezintă pe 9.4 nu este egal cu 9.4, deși este foarte aproape de această valoare
- pentru a cuantifica această apropiere, vom folosi definiția standard a erorii

Definiția 3

Fie x_c o versiune calculată a valorii exacte x . Atunci

eroarea absolută = $|x_c - x|$,

și

eroarea relativă = $\frac{|x_c - x|}{|x|}$,

dacă această din urmă cantitate există ($x \neq 0$).

Algoritmul 2 (Eroarea relativă de rotunjire)

În cadrul standardului IEEE, eroarea relativă de rotunjire $\text{fl}(x)$ este mai mică decât jumătate din numărul epsilon mașină:

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \frac{1}{2} \epsilon_{\text{mach}}. \quad (9)$$

- în cazul numărului $x = 9.4$, am găsit eroarea de rotunjire în (8), care trebuie să satisfacă (9):

$$\frac{|\text{fl}(9.4) - 9.4|}{9.4} = \frac{0.2 \times 2^{-49}}{9.4} = \frac{8}{47} \times 2^{-52} < \frac{1}{2} \epsilon_{\text{mach}}.$$

Exemplul 2

- găsiți reprezentarea în dublă precizie $f(x)$ și eroarea de rotunjire pentru $x = 0.4$
 - deoarece $(0.4)_{10} = (0.\overline{0110})_2$, alinierea la stânga a acestui număr binar ne dă:

- prin urmare, în conformitate cu regula de rotunjire, $f_1(0.4)$ este

- aici, 1 a fost adunat la bitul 52, ceea ce a determinat o schimbare și a bitului 51, ca urmare a transportului din adunarea binară
 - analizând cu atenție, am înălăturat $2^{-53} \times 2^{-2} + 0.\overline{0110} \times 2^{-54} \times 2^{-2}$ în cadrul trunchierii și am adunat $2^{-52} \times 2^{-2}$ prin rotunjirea în sus

1.3.1 Formate de virgulă flotantă

- prin urmare,

$$\begin{aligned}
 \text{fl}(0.4) &= 0.4 - 2^{-55} - 0.4 \times 2^{-56} + 2^{-54} \\
 &= 0.4 + 2^{-54}(-1/2 - 0.1 + 1) \\
 &= 0.4 + 2^{-54}(0.4) \\
 &= 0.4 + 0.1 \times 2^{-52}.
 \end{aligned}$$

- observăm că eroarea relativă de rotunjire pentru 0.4 este $0.1/0.4 \times \epsilon_{\text{mach}} = 1/4 \times \epsilon_{\text{mach}}$, conform cu (9)

1.3.2. Reprezentarea numerelor în calculator

fiecare astfel de cuvânt are forma

$$se_1e_2\dots e_{11}b_1b_2\dots b_{52}, \quad (10)$$

unde mai întâi este stocat semnul, urmat de 11 biți reprezentând exponentul și de 52 de biți care urmează virgulei zecimale, care reprezintă mantisa

bitul de semn s este 0 pentru un număr pozitiv și 1 pentru un număr negativ.

cei 11 biți reprezentând exponentul vin din numărul binar pozitiv care rezultă din adunarea lui $2^{10} - 1 = 1023$ la exponent, cel puțin pentru exponentii între -1022 și 1023 .

- aceasta acoperă valorile lui $e_1 \dots e_{11}$ de la 1 la 2046, lăsând valorile 0 și 2047 pentru scopuri speciale, care vor fi detaliate ulterior
- numărul 1023 se numește **biasul exponent** al formatului în dublă precizie
- este folosit pentru a converti atât exponentii pozitivi cât și pe cei negativi în numere binare pozitive pentru a fi stocate în bițiile de exponent
- pentru precizia simplă și dublă lungă, valorile biasului exponent sunt 127 și respectiv 16383
- formatul hexazecimal constă din exprimarea celor 64 de biți ai reprezentării în virgulă flotantă (10) ca 16 numere hexazecimale
- prin urmare, primele 3 numerale hexazecimale reprezintă semnul și exponentul combinate, în timp ce ultimele 13 conțin mantisa

Exemplul 3

- găsiți reprezentarea în virgulă flotantă în dublă precizie a numărului real 9.4
- din (7), avem că semnul este $s = 0$, exponentul este 3, și cei 52 de biți ai mantisei de după virgula zecimală sunt

`0010|1100|1100|1100|1100|1100|1100|1100|1100|1100|1100|1100|1100|1101` $\rightarrow (2CCCCCCCCCCCCD)_{16}$.

- adunând 1023 la exponent obținem $1026 = 2^{10} + 2$, sau $(10000000010)_2$
- combinarea semn – exponent este $(010000000010)_2 = (402)_{16}$, dând formatul hexazecimal $4022CCCCCCCCCCCCD$

- ne întoarcem acum la valorile speciale ale exponentului 0 și 2047
- ultima, 2047, este folosită pentru a reprezenta ∞ dacă sirul de biți ai mantisei este format doar din zerouri și NaN, adică Not a Number, altfel
- deoarece 2047 este reprezentat prin unsprezece biți de 1, sau $e_1 e_2 \dots e_{11} = (111111111111)_2$, primii doisprezece biți ai lui Inf și -Inf sunt `0111|1111|1111` și, respectiv, `1111|1111|1111`, și cei 52 de biți rămași (mantisa) sunt zero
- numărul NaN începe de asemenea cu `1111|1111|1111` dar are o mantisă nenulă
- putem rezuma cele de mai sus în tabelul următor:

numărul	exemplu	format hexazecimal
+Inf	1/0	<code>7FF00000000000000</code>
-Inf	-1/0	<code>FFF00000000000000</code>
NaN	0/0	<code>FFFxxxxxxxxxxxxxx</code>

unde x-urile denotă biți care nu sunt toți egali cu zero

- exponentul special 0, adică $e_1 e_2 \dots e_{11} = (000\ 0000\ 0000)_2$, denotă de asemenea o abatere de la forma standard a unui număr în virgulă flotantă
 - în acest caz, numărul este interpretat ca numărul în virgulă flotantă nenormalizat

$$\pm 0.b_1 b_2 \dots b_{52} \times 2^{-1022}. \quad (11)$$

- adică, doar în acest caz, bitul cel mai din stânga nu este presupus a fi egal cu 1
 - aceste numere nenormalizate se numesc numere în virgulă flotantă **subnormale**
 - ele extind gama numerelor foarte mici cu încă câteva ordine de mărime
 - prin urmare, $2^{-52} \times 2^{-1022} = 2^{-1074}$ este cel mai mic număr diferit de zero care este reprezentabil în dublă precizie
 - cuvântul corespunzător care va fi stocat în calculator este

- trebuie făcută distincția între cel mai mic număr reprezentabil 2^{-1074} și $\epsilon_{\text{mach}} = 2^{-52}$
 - sunt multe numere mai mici decât ϵ_{mach} care sunt reprezentabile, chiar dacă adunându-le la 1 nu vom obține niciun efect
 - pe de altă parte, numerele în dublă precizie mai mici decât 2^{-1074} nu pot fi reprezentate deloc
 - numerele subnormale includ și cel mai important număr, 0
 - de fapt, reprezentarea subnormală include două numere în virgulă flotantă diferite, $+0$ și -0 , care sunt tratate în calcule ca fiind același număr real
 - reprezentarea în virgulă flotantă a lui $+0$ are bitul de semn $s = 0$, biții de exponent $e_1 \dots e_{11} = 00000000000$, și mantisa 52 de zerouri; pe scurt, toți cei 64 de biți sunt zero
 - formatul hexazecimal pentru $+0$ este 0000000000000000
 - pentru numărul -0 , totul este exact la fel, cu excepția bitului de semn $s = 1$
 - formatul hexazecimal pentru -0 este 8000000000000000

1.3.3 Adunarea numerelor în virgulă flotantă

- adunarea în calculator constă din alinierea virgulelor zecimale ale celor două numere de adunat, adunarea lor, și stocarea rezultatului tot ca un număr în virgulă flotantă
 - adunarea propriu-zisă poate fi realizată cu precizie mai mare (cu mai mult de 52 de biți) deoarece are loc într-un registru dedicat special pentru acest scop
 - după adunare, rezultatul trebuie rotunjit înapoi la 52 de biți după virgula binară pentru a fi stocat ca un număr în virgulă flotantă
 - de exemplu, adunarea lui 1 la 2^{-53} va apărea după cum urmează:

- acest număr este stocat ca $1.0 \times 2^0 = 1$, conform regulii de rotunjire
 - prin urmare, $1 + 2^{-53}$ este egal cu 1 în aritmetică în dublă precizie IEEE

Exemplul 4

- găsiți suma în virgulă flotantă în dublă precizie $(1 + 3 \times 2^{-53}) - 1$
 - bineînțeles, în aritmetică reală, răspunsul este 3×2^{-53}
 - însă calculele în aritmetică în virgulă flotantă pot da un rezultat diferit
 - observăm că $3 \times 2^{-53} = 2^{-52} + 2^{-53}$
 - prima adunare este

- acesta este cazul de excepție pentru regula rotunjirii
 - deoarece bitul 52 din sumă este 1, trebuie să facem o rotunjire în sus, ceea ce înseamnă să adunăm 1 la bitul 52
 - după transport, obținem

care este reprezentarea lui $1 + 2^{-51}$

- prin urmare, după ce scădem 1, rezultatul va fi 2^{-51} , care este egal cu $2\epsilon_{\text{mach}} = 4 \times 2^{-53}$
 - din nou, se poate observa diferența dintre aritmetică în calculator și aritmetică exactă

2. Rezolvarea ecuațiilor

Def 1: Funcția $f(x)$ are o rădăcină în $x=r$ dacă $f(r)=0$

Def 2: r este un punct fix al funcției g dacă $g(r) = r$

Proprietate: Orice ecuație $f(x) = 0$ poate fi transformată într-o problemă de punct fix.

$$l(x) : x^3 + x - 1 = 0$$

$$1) \quad x^3 + x - 1 = 0 \quad | + 1 - x \Rightarrow \quad x = 1 - x^3 = g(x)$$

$$2) \quad x^3 + x - 1 = 0 \quad | + 1 - x \quad \Rightarrow \quad x^3 = 1 - x \quad \Rightarrow \quad x = \sqrt[3]{1-x} = g(x)$$

$$\begin{aligned}
 3) \quad x^3 + x - 1 &= 0 \mid +2x^3 + 1 \Rightarrow 3x^3 + x = 2x^3 + 1 \\
 &\Rightarrow x(3x^2 + 1) = 2x^3 + 1 \\
 &\Rightarrow x = \frac{2x^3 + 1}{3x^2 + 1} = g(x)
 \end{aligned}$$

2. 1. Metoda bisectiei $\rightarrow f(x) = 0$

2.1.1. Găsirea limitelor între care se află o rădăcină

\rightarrow verificăm dacă $\exists \text{ o rădăcină}$

$$\left. \begin{array}{l} [a, b] \\ f(a) \cdot f(b) < 0 \\ f - \text{continuă și } r \in [a, b] \end{array} \right\} \Rightarrow f(r) = 0$$

Teorema 1 (Teorema valorii intermediare)

Fie f o funcție continuă pe intervalul $[a, b]$. Atunci f ia orice valoare între $f(a)$ și $f(b)$. Mai precis, dacă y este un număr între $f(a)$ și $f(b)$, atunci există un număr c care satisfacă $a \leq c \leq b$ astfel încât $f(c) = y$.

Teorema 2

Fie f o funcție continuă pe $[a, b]$, care satisfacă $f(a)f(b) < 0$. Atunci f are o rădăcină între a și b , adică există un număr r care satisfacă $a < r < b$ și $f(r) = 0$.

- În Figura 1, $f(0)f(1) = (-1)(1) < 0$
- există o rădăcină imediat la stânga lui 0.7
- cum putem rafina presupunerea inițială despre locația rădăcinii cu mai multe zecimale exacte?

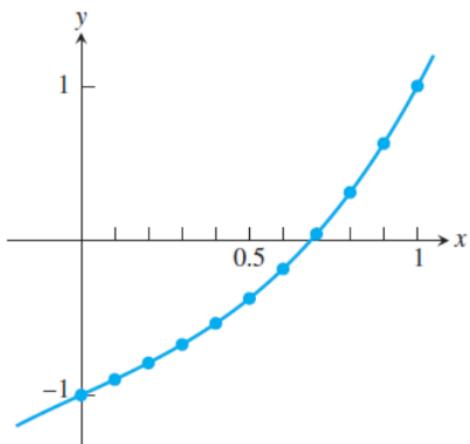


Figura 1: Graficul funcției $f(x) = x^3 + x - 1$. Funcția are o rădăcină între 0.6 și 0.7.

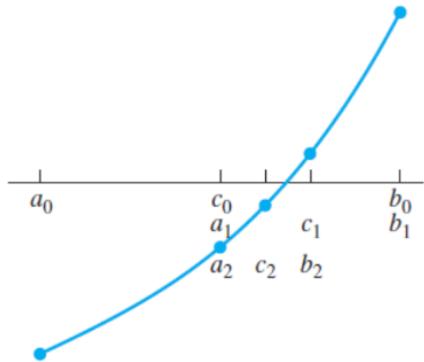


Figura 2: Metoda bisecției. În primul pas, semnul lui $f(c_0)$ este verificat. Deoarece $f(c_0)f(b_0) < 0$, asignăm $a_1 = c_0$, $b_1 = b_0$, și intervalul este înlocuit cu jumătatea lui dreaptă $[a_1, b_1]$. În al doilea pas, subintervalul este înlocuit cu jumătatea lui stângă $[a_2, b_2]$.

Algoritmul 3 (Metoda bisecției)

Dându-se un interval inițial $[a, b]$ astfel încât $f(a)f(b) < 0$

```

while  $(b - a)/2 > \text{TOL}$ 
     $c = (a + b)/2$ 
    if  $f(c) = 0$ , stop, end
    if  $f(a)f(c) < 0$ 
         $b = c$ 
    else
         $a = c$ 
    end
end

```

Intervalul final $[a, b]$ conține o rădăcină.

Aproximarea rădăcinii este $(a + b)/2$.

```

function  $x = \text{metoda\_bisectiei}(f, a, b, \text{tol})$ 

if  $f(a) * f(b) \geq 0$ 
    error('Conditia  $f(a) * f(b) < 0$  nu este satisfacuta!')
end

while  $(b - a) / 2 > \text{tol}$ 
     $c = (a + b) / 2;$ 
    if  $f(c) == 0$ 
        break;
    end
    if  $f(a) * f(c) < 0$ 
         $b = c;$ 
    else
         $a = c;$ 
    end
end

 $x = (a + b) / 2;$ 

```

- verificăm valoarea funcției la mijlocul $c = (a + b)/2$ al intervalului
- deoarece $f(a)$ și $f(b)$ au semne opuse, ori $f(c) = 0$ (în care caz am găsit o rădăcină și suntem gata), ori semnul lui $f(c)$ este opus semnului lui $f(a)$ sau $f(b)$
- dacă $f(c)f(a) < 0$, de exemplu, suntem asigurați că o soluție se află în intervalul $[a, c]$, a cărui lungime este jumătate din cea a intervalului inițial $[a, b]$
- dacă, însă, $f(c)f(b) < 0$, atunci putem spune același lucru despre intervalul $[c, b]$
- în ambele cazuri, un pas reduce problema la găsirea unei rădăcini pe un interval care are lungimea egală cu jumătate din lungimea intervalului inițial
- acest pas poate fi repetat pentru a localiza rădăcina din ce în ce mai exact
- la fiecare pas, limitele între care se află rădăcina sunt altele, reducând incertitudinea despre localizarea soluției, pe măsură ce intervalul devine din ce în ce mai mic

Exemplul 5

$$f(x) = x^3 + x - 1 \quad , \quad [0, 1]$$

$$\cdot f(0) \cdot f(1) = (-1)(1) = -1 < 0 \Rightarrow \exists \text{ rădăcină}$$

$$c_0 = \frac{0+1}{2} = \frac{1}{2} = 0,5$$

$$f\left(\frac{1}{2}\right) = -\frac{3}{8} < 0$$

\Rightarrow nou interval $[a_1, b_1] = [\frac{1}{2}, 1]$, deoarece $f\left(\frac{1}{2}\right) f(1) < 0$

$$f(c_1) = f\left(\frac{\frac{3}{4}}{2}\right) = \frac{11}{64} > 0$$

\Rightarrow nou interval $[a_2, b_2] = [\frac{1}{2}, \frac{3}{4}]$

continuând în această manieră, obținem următoarele intervale:

i	a_i	$f(a_i)$	c_i	$f(c_i)$	b_i	$f(b_i)$
0	0.0000	—	0.5000	—	1.0000	+
1	0.5000	—	0.7500	+	1.0000	+
2	0.5000	—	0.6250	—	0.7500	+
3	0.6250	—	0.6875	+	0.7500	+
4	0.6250	—	0.6562	—	0.6875	+
5	0.6562	—	0.6719	—	0.6875	+
6	0.6719	—	0.6797	—	0.6875	+
7	0.6797	—	0.6836	+	0.6875	+
8	0.6797	—	0.6816	—	0.6836	+
9	0.6816	—	0.6826	+	0.6836	+

Dim talel \Rightarrow soluția se află între $a_7 \approx 0,6816$

$$C_g \approx 0,6826$$

Mijlocul acestui interval este $c_{1,0} \approx 0,6821$ (aproximarea rădăcinii)

- deși problema a fost să găsim o rădăcină, am găsit de fapt un interval $[0.6816, 0.6826]$ care conține o rădăcină; cu alte cuvinte, rădăcina este $r = 0.6821 \pm 0.0005$
- va trebui să fim mulțumiți cu o aproximare
- bineînțeles, această aproximare poate fi îmbunătățită, dacă este necesar, prin efectuarea mai multor pași din metoda bisecției

Pași de rezolvare

- la fiecare pas al metodei bisecției, calculăm mijlocul $c_i = (a_i + b_i)/2$ al intervalului curent $[a_i, b_i]$, calculăm $f(c_i)$, și comparăm semnele
 - dacă $f(c_i)f(a_i) < 0$, asignăm $a_{i+1} = a_i$ și $b_{i+1} = c_i$
 - dacă, din contră, $f(c_i)f(a_i) > 0$, asignăm $a_{i+1} = c_i$ și $b_{i+1} = b_i$
 - fiecare pas necesită o nouă evaluare a funcției f și împarte intervalul care conține o rădăcină, reducându-i lungimea cu un factor de 2
 - după n pași de calcul al lui c și $f(c)$, am realizat $n+2$ evaluări de funcție, și cea mai bună estimare a soluției este mijlocul ultimului interval obținut
-
- dacă $[a, b]$ este intervalul inițial, atunci după n pași ai metodei bisecției, intervalul $[a_n, b_n]$ are lungimea $(b - a)/2^n$
 - alegând mijlocul $x_c = (a_n + b_n)/2$ obținem o aproximare a soluției r , care este la jumătate din lungimea intervalului de soluție adevărată
 - rezumând, după n pași din metoda bisecției, avem că

$$\text{Eroarea soluției} = |x_c - r| < \frac{b - a}{2^{n+1}}, \quad (12)$$

$$\text{Evaluări de funcție} = n + 2. \quad (13)$$

- o modalitate bună de a evalua eficiența metodei bisecției este de a ne întreba câtă acuratețe poate fi adusă de fiecare evaluare de funcție
- fiecare pas, sau fiecare evaluare de funcție, reduce incertitudinea în găsirea rădăcinii cu un factor de 2

Definiția 3

O soluție este **corectă cu p zecimale exacte** dacă eroarea este mai mică decât 0.5×10^{-p} .

Exemplul 6 - rădăcină?

$f(x) = \cos x - x$; $[0, 1]$ cu 6 zecimale

Eroarea după m pași:

$$\left\{ \frac{b-a}{2^{m+1}} \right\} = \frac{1}{2^{m+1}}$$

$$\frac{1}{2^{m+1}} < 0,5 \cdot 10^{-6}$$

$$\Rightarrow m > \frac{6}{\log_{10} 2} = 19.9$$

$\Rightarrow m = 20$ pași necesari

- aplicând metoda bisecției, obținem următorul tabel:

k	a_k	$f(a_k)$	c_k	$f(c_k)$	b_k	$f(b_k)$
0	0.000000	+	0.500000	+	1.000000	-
1	0.500000	+	0.750000	-	1.000000	-
2	0.500000	+	0.625000	+	0.750000	-
3	0.625000	+	0.687500	+	0.750000	-
4	0.687500	+	0.718750	+	0.750000	-
5	0.718750	+	0.734375	+	0.750000	-
6	0.734375	+	0.742188	-	0.750000	-
7	0.734375	+	0.738281	+	0.742188	-
8	0.738281	+	0.740234	-	0.742188	-
9	0.738281	+	0.739258	-	0.740234	-
10	0.738281	+	0.738770	+	0.739258	-
11	0.738769	+	0.739014	+	0.739258	-
12	0.739013	+	0.739136	-	0.739258	-
13	0.739013	+	0.739075	+	0.739136	-

k	a_k	$f(a_k)$	c_k	$f(c_k)$	b_k	$f(b_k)$
14	0.739074	+	0.739105	-	0.739136	-
15	0.739074	+	0.739090	-	0.739105	-
16	0.739074	+	0.739082	+	0.739090	-
17	0.739082	+	0.739086	-	0.739090	-
18	0.739082	+	0.739084	+	0.739086	-
19	0.739084	+	0.739085	-	0.739086	-
20	0.739084	+	0.739085	-	0.739085	-

- aproximarea rădăcinii cu şase zecimale exacte este 0.739085

- pentru metoda bisecției, întrebarea câți pași să facem este una simplă—trebuie doar să alegem precizia dorită și să găsim numărul de pași necesari, ca în (12)
- vom vedea că unii algoritmi mai puternici sunt adesea mai puțin predictibili și nu au un analog al (12)
- în acele cazuri, va fi nevoie să stabilim anumite „criterii de oprire” care vor determina circumstanțele în care algoritmul se va opri
- chiar și pentru metoda bisecției, precizia finită a aritmeticii în calculator va pune o limită numărului posibil de zecimale exacte