4.1 Теңдеулердің түбірлері

$f(x) = 0$ шешімдерін табыңыз, мұнда f функциясы берілген.

Кіріспе

Инженерлік талдауда жиі кездесетін мәселе келесідей: $f(x)$ функциясы берілген, $f(x) = 0$ болатын x мәндерін анықтаңыз. Шешімдері (x мәндері) $f(x) = 0$ теңдеуінің түбірлері немесе f (x) функциясының нөлдері ретінде белгілі.

Әрі қарай жалғастырмас бұрын, функция тұжырымдамасын қарап шығу пайдалы болуы мүмкін.

$$y = f(x)$$

теңдеуі үш элементтен тұрады: кіріс мәні x, шығыс мәні y және y есептеуге арналған f ережесі. Егер f ережесі көрсетілген болса, функция берілген деп аталады. Сандық есептеулерде ереже әрқашан компьютерлік алгоритм болып табылады. Бұл функция мәлімдемесі болуы мүмкін, мысалы

$$f(x) = cosh(x) \cdot cos(x) - 1$$

немесе жүздеген немесе мыңдаған код жолдарын қамтитын күрделі процедура. Алгоритм әрбір x кірісі үшін y шығысын шығаратын болса, ол функция ретінде сипатталады.

Теңдеулердің түбірлері нақты немесе комплекс болуы мүмкін. Комплекс түбірлер сирек есептелінеді, өйткені олардың физикалық мәні сирек болады. Ерекше жағдайда көпмүшелік теңдеу құрайды

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

мұнда комплекс түбірлер мағыналы болуы мүмкін (мысалы, әлсіретілген тербелістерді талдау сияқты). Алдымен теңдеулердің нақты түбірлерін табуға назар аударамыз. Көпмүшелердің комплекс нөлдері соңында қарастырылады.

Жалпы алғанда, теңдеудің кез келген санды (нақты) түбірлері болуы немесе мүлде түбірлері болмауы мүмкін. Мысалы,

$$\sin x \ - x = 0$$

бір түбірі бар, атап айтқанда $x = 0$, ал мына жағдайда

$$\mathrm{tg}\, x - x = 0$$

түбірлерінің саны шексіз көп ($x = \ 0, \pm 4{,}493, \pm 7{,}725, \ldots$).

Түбірлерді табудың барлық әдістері бастапқы нүктені (яғни, түбірді бағалау) қажет ететін итерациялық процедуралар болып табылады. Бұл бастапқы нүктені бағалау өте маңызды; нашар бастапқы мән жинақталмауы мүмкін немесе ол «дұрыс

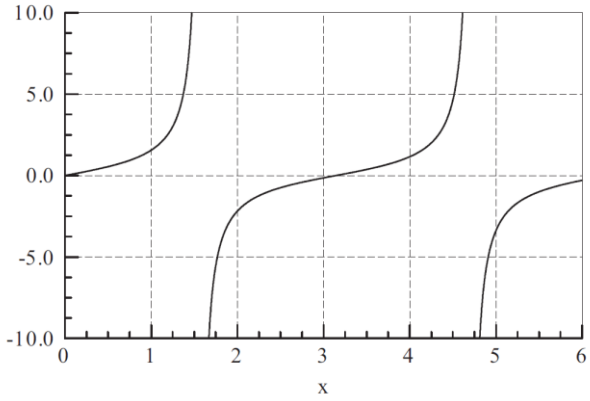
емес» түбірге (ізделгеннен басқа түбір) жақындауы мүмкін. Түбірдің құндылығын бағалаудың әмбебап тәсілі жоқ. Егер теңдеу физикалық есеппен байланысты болса, онда мәселенің мазмұны (физикалық түсінік) түбірдің шамамен орналасуын ұсынуы мүмкін. Әйтпесе, түбірлерді жүйелі түрде сандық іздеуге болады. Осындай іздеу әдістерінің бірі төменде сипатталған. Функцияның графигі – түбірлерді табудың тағы бір құралы, бірақ бұл бағдарламалау мүмкін емес визуалды процедура.

Мәселені түбірін табу алгоритміне бермес бұрын, бір қадам алға басып, түбір жатқан аралықты (оның төменгі және жоғарғы шекараларын анықтаңыз) анықтау ұсынылады. Осы тарауда сипатталған әдістерде алдын ала аралықты табу, шын мәнінде, міндетті болып табылады.

Қосымша іздеу әдісі

Түбірлердің жуықтап орналасуы функцияның графигін салу арқылы жақсы анықталады. Көбінесе бірнеше нүктеге негізделген өте өрескел график ақылға қонымды бастапқы мәндерді қамтамасыз ету үшін жеткілікті. Түбірлерді анықтау және аралықты алудың тағы бір пайдалы құралы - қосымша іздеу әдісі. Оны түбірлерді есептеу үшін де бейімдеуге болады, бірақ осымен есептеудің пайдасы аз, өйткені осы сабақта сипатталған басқа әдістер тиімдірек.

Қосымша іздеу әдісінің негізгі идеясы қарапайым: $f(x_1)$ және $f(x_2)$ мәндері қарама-қарсы таңбаларға ие болса, онда $(x_1; x_2)$ аралықта кем дегенде бір түбір бар. Егер интервал жеткілікті аз болса, онда бір түбір болуы мүмкін. Осылайша, $f(x)$ нөлдерін функцияны x аралықтарында бағалау және таңбасының өзгерісін іздеу арқылы анықтауға болады.



4.1-сурет. $\mathrm{tg}\, x$ графигі.

Қосымша іздеу әдісімен бірнеше қиындықтар бар:

• Іздеу қадамы x түбірлер аралығынан үлкенірек болса, жақын орналасқан екі түбірді бірін таба алмауы мүмкін.

• Еселі түбір (сәйкес келетін екі түбір) анықталмайды.

• $f(x)$ белгілі бір ерекшеліктерді (полюстерін) түбірлер деп қабылдауы мүмкін. Мысалы, $f(x) = \text{tg}\,x$ белгісін $x = \pm\frac{1}{2}n\pi,\ n = 1,3,5,\ldots$ кезінде өзгертеді 4.1-суретте көрсетілгендей. Дегенмен, бұл орындар шынайы нөлдер емес, өйткені функция Ox осін кесіп өтпейді.

rootsearch

Бұл функция пайдаланушы ұсынған $f(x)$ функциясының нөлін (a,b) интервалында dx қадамдарымен іздейді. Ол іздеу сәтті болған жағдайда түбірдің шекараларын $(x1,x2)$ қайтарады; $x1 = x2 = None$ түбірлердің анықталмағанын көрсетеді. Бірінші түбір (a-ға жақын түбір) анықталғаннан кейін, келесі түбірді табу үшін rootsearch-ті $x2$-ге ауыстыру арқылы қайта шақыруға болады. Бұл rootsearch түбірді анықтағанша қайталануы мүмкін.

```
## module rootsearch
''' x1,x2 = rootsearch(f,a,b,dx).
  f(x) ең кіші түбірінің шекаралары (x1,x2) үшін
  dx қадамдарымен (a,b) аралығын іздейді.
  Түбірлер анықталмаса, x1 = x2 = None қайтарады.
'''
from numpy import sign
def rootsearch(f,a,b,dx):
    x1 = a; f1 = f(a)
    x2 = a + dx; f2 = f(x2)
    while sign(f1) == sign(f2):
        if x1 >= b: return None,None
        x1 = x2; f1 = f2
        x2 = x1 + dx; f2 = f(x2)
    else:
    return x1,x2
```

МЫСАЛ 4.1

$x^3 - 10x^2 + 5 = 0$ түбірі $(0,1)$ интервалында жатыр. Бұл түбірді төрт сандық дәлдікпен есептеу үшін rootsearch пайдаланыңыз.

Шешуі. Төрт таңбалы дәлдікті алу үшін бізге $x = 0.0001$-ден үлкен емес іздеу қадамы қажет. $(0, 1)$ аралығын x қадамдарымен іздеу осылайша 10 000 функцияны есептеуді талап етеді. Келесі бағдарлама түбірді төрт кезеңде жабу арқылы функция есептеулер санын 40-қа дейін азайтады, әрбір кезең 10 іздеу аралығын (және осылайша функцияны 10 рет есептейді) қамтиды.

```
#!/usr/bin/python
## example4_1
from rootsearch import *

def f(x): return x**3 - 10.0*x**2 + 5.0

x1 = 0.0; x2 = 1.0
for i in range(4):
    dx = (x2 - x1)/10.0
    x1,x2 = rootsearch(f,x1,x2,dx)
x = (x1 + x2)/2.0
print('x =', '{:6.4f}'.format(x))
input("Press return to exit")
```

Нәтижесі

```
x = 0.7346
```

Екіге бөлу әдісі

$f(x) = 0$ түбірі (x_1, x_2) аралықта жақшаға алынғаннан кейін, оны тұйықтау үшін бірнеше әдістерді қолдануға болады. Екіге бөлу әдісі аралық жеткілікті аз болғанша екі есе қысқарту арқылы жүзеге асырады. Бұл әдіс интервалды жартыға бөлу әдісі ретінде де белгілі. Екіге бөлу түбірлерді есептеуге арналған ең жылдам әдіс емес, бірақ ол ең сенімді әдіс. Түбір жақшаға алынғаннан кейін, екіге бөлу әрқашан оны тұйықтайды.

Екіге бөлу әдісі қосымша іздеу сияқты принципті қолданады: Егер (x_1, x_2) аралықта түбір болса, онда $f(x_1) \cdot f(x_2) < 0$ және $f(x_2)$ қарама-қарсы таңбаларға ие болады. Аралықты екі есе азайту үшін $f(x_3)$есептейміз, мұндағы $x_3 = \frac{1}{2}(x_1 + x_2) -$ интервалдың ортасы. Егер $f(x_2)$ және $f(x_3)$ таңбалары қарама-қарсы болса, онда түбір (x_3, x_2) ішінде болуы керек, және біз x_1-ді x_3-ке ауыстыру арқылы жазамыз. Әйтпесе, түбір (x_1, x_3) орналасады, бұл жағдайда x_2-ні x_3-ке ауыстырады. Кез келген

жағдайда жаңа интервал (x_1, x_2) бастапқы интервалдың жарты өлшеміне тең. Екіге бөлу интервал аз ε мәніне дейін азайғанша қайталанады, осылайша

$$|x_2 - x_1| \leq \varepsilon$$

Белгіленген ε мәніне жету үшін қажетті екіге бөлу санын есептеу оңай. Бастапқы Δx интервалы бір бөліктен кейін $\Delta x/2$-ге, екі бөліктен кейін $\Delta x/2^2$-ге, ал n бөліктен кейін $\Delta x/2^n$-ге дейін азаяды. $\Delta x/2^n = \varepsilon$ орнату және n-ге қатысты шешу арқылы біз Итерация саны

$$n = \frac{\ln \Delta x/\varepsilon}{\ln 2} \qquad (1)$$

аламыз n бүтін сан болуы керек болғандықтан, n жоғары дөңгелектеу пайдаланылады (n жоғары дөңгелектеу – n үлкен ең кіші бүтін сан).

Bisection модулі

Бұл функция $(x1, x2)$ интервалында жататыны белгілі $f(x) = 0$ түбірін есептеу үшін екіге бөлу әдісін пайдаланады. tol-ға дейінгі аралықты азайту үшін қажетті екіге бөлу n саны (1) теңдіктен есептеледі. $switch = 1$ параметрін орнату арқылы біз әрбір аралық жартыға қысқарту кезінде f(x) шамасының азаюын тексереміз. Олай болмаса, бірдеңе дұрыс емес болуы мүмкін («түбір» мүлде түбір емес, полюс болуы мүмкін) және $root = None$ қайтарылады. Бұл мүмкіндік әрқашан қажет болмағандықтан, әдепкі мән - $switch = 0$. Бағдарламаны тоқтату үшін пайдаланылатын error.err функциясы пайдалыналады.

```
## module bisection
''' root = bisection(f,x1,x2,switch=0,tol=1.0e-9).
    f(x)=0 түбірін екіге бөлу арқылы табады.
    Түбір (x1,x2) аралықта жатуы керек.
    switch = 1 орнатылса root = None қайтарады
    егер f(x) екіге бөлінгенде мәні артса
'''
import math
from numpy import sign
import sys
def err(string):
    print(string)
    input('Press return to exit')
    sys.exit()
```

```
def bisection(f,x1,x2,switch=1,tol=1.0e-9):
    f1 = f(x1)
    if f1 == 0.0: return x1
    f2 = f(x2)
    if f2 == 0.0: return x2
    if sign(f1) == sign(f2):
        err('Түбір жақшада жоқ')
    n = int(math.ceil(math.log(abs(x2 - x1)/tol)/math.log(2.0)))
    for i in range(n):
        x3 = 0.5*(x1 + x2); f3 = f(x3)
        if (switch == 1) and (abs(f3) > abs(f1)) \
                        and (abs(f3) > abs(f2)):
            return None
        if f3 == 0.0: return x3
        if sign(f2)!= sign(f3): x1 = x3; f1 = f3
        else: x2 = x3; f2 = f3
    return (x1 + x2)/2.0
```

МЫСАЛ 4.2

(0, 1) төрт таңбалы дәлдік аралығында орналасқан $x^3 - 10x^2 + 5 = 0$ түбірін табу үшін екіге бөлу әдісін пайдаланыңыз. Процедура неше рет функцияны есептейді?

Шешуі. Бағдарламасы:

```
from bisection import *

def f(x): return x**3 - 10.0*x**2 + 5.0

x = bisection(f, 0.0, 1.0, tol = 1.0e-4)
print('x =', '{:6.4f}'.format(x))
input("Press return to exit")
```

Біз дәлдікті төрт маңызды санға дейін шектеу үшін ε = 0,0001 (tol = 1,0e-4) орнатқанымызды ескеріңіз. Нәтижесі

```
x = 0.7346
```

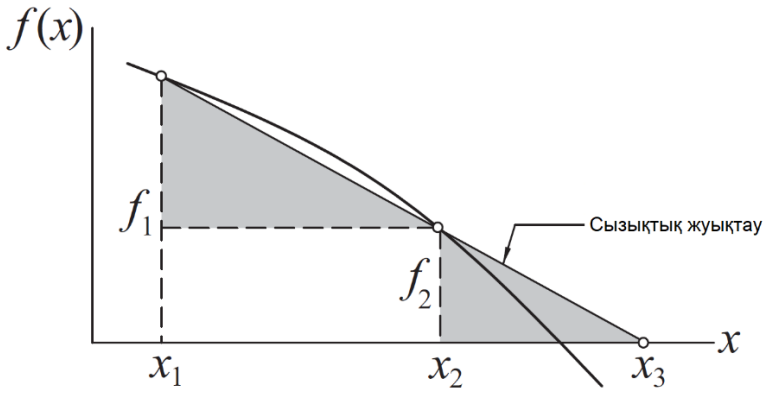
(1)теңдеуіне сәйкес.

$$n = \frac{\ln(\Delta x/\varepsilon)}{\ln 2} = \frac{\ln(1.0/0.0001)}{\ln 2} = 13.29$$

Демек, екіге бөлу арналған циклдегі функция бағалауларының саны $[13.29] = 14$. Ішкі бағдарламаның басында 2 рет есептеледі, барлығы 16 рет функция есептелінеді.

Қиюшылар және жалған орын әдістері

Қиюшылар және жалған орын әдістері бір-бірімен тығыз байланысты. Екі әдіс те түбірдің екі бастапқы бағалауын талап етеді, мысалы, x_1 және x_2. $f(x)$ функциясы түбірдің жанында шамамен сызықты болып қабылданады, сондықтан түбірдің жақсартылған x_3 мәнін x_1 және x_2 арасындағы сызықтық интерполяция арқылы бағалауға болады.



2-сурет Сызықты интерполяция

2-суретке сілтеме жасай отырып, ұқсас үшбұрыштар (суретте боялған) мына қатынасты береді

$$\frac{f_2}{x_3 - x_2} = \frac{f_1 - f_2}{x_2 - x_1}$$

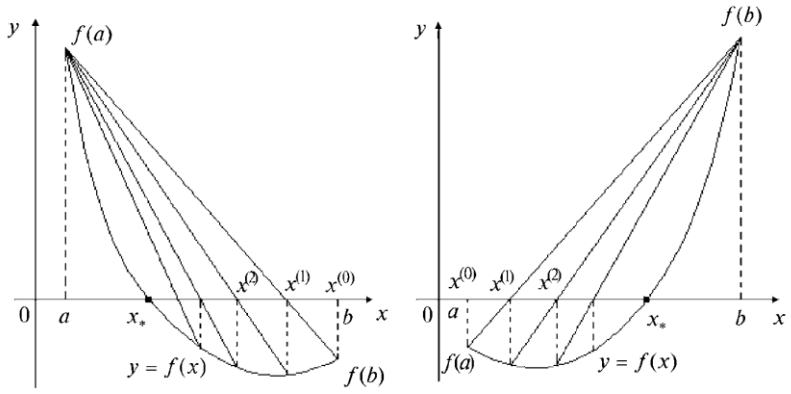
мұнда $f_i = f(x_i)$ белгісін қолдандық. Осылайша, түбірдің жақсартылған бағалауы

$$x_3 = x_2 - f_2 \frac{x_2 - x_1}{f_2 - f_1} \qquad (2)$$

Жалған орын әдісі (regula falsi ретінде белгілі) түбір x_1 және x_2 аралығында жату қажет. Жақсартылған түбір (2)теңдеуден есептелгеннен кейін, x_1 немесе x_2 мәні x_3-ке ауыстырылады. f_3-тің таңбасы f_1-мен бірдей болса, $x_1 \leftarrow x_3$ меншіктейміз; әйтпесе $x_2 \leftarrow x_3$ таңдаймыз. Осылайша, түбір әрқашан (x_1, x_2) аралығында жатады. Содан кейін процедура жинақталғанша қайталанады.

Қиюшылар әдісі жалған орын әдісінен екі жолмен ерекшеленеді: ол түбірді алдын ала жақшаға алуды қажет етпейді және ол түбірдің ескі алдыңғы бағасын жоққа шығарады (яғни, x_3 есептелгеннен кейін, біз $x_1 \leftarrow x_2, x_2 \leftarrow x_3$ мәнін береміз).

Қиюшылар әдісінің жинақтылығын суперсызықты етіп көрсетуге болады, қателігі $E_{k+1} = c \cdot E_k^{1.618}$ (1.618... көрсеткіші «алтын қима» қатынасы). Жалған орын әдісі үшін жинақтылықтың нақты тәртібін есептеу мүмкін емес. Жалпы, бұл сызықтыққа қарағанда біршама жақсырақ, бірақ көп емес. Дегенмен, жалған орын әдісі әрқашан түбірді жақшаға алатындықтан, ол сенімдірек. Біз бұл әдістерді әрі қарай зерттемейміз, өйткені олардың екеуі де жинақталу реті бойынша Риддер әдісінен төмен.



2-сурет Қиюшылар әдісінің сызбасы

Қиюшы теңдеуі

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(b)-f(a)}$$

$y = 0$ деп ескеріп, итерациялық формула мынадай түрде жазылады

$$x^{(0)} = b$$

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f(x^{(k)}) - f(a)}\left(x^{(k)} - a\right)$$

$$x^{(0)} = a$$

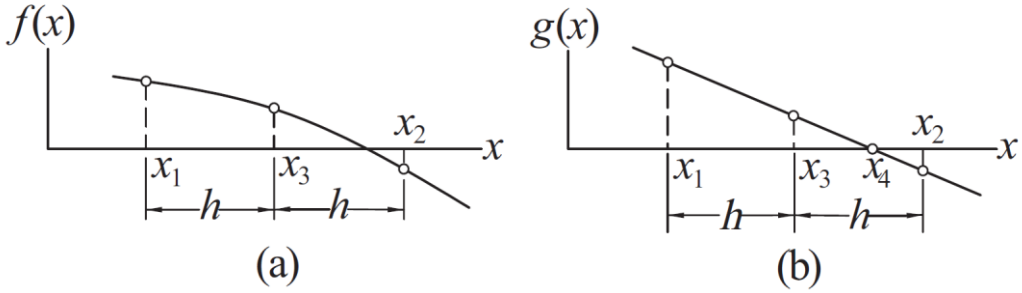
$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f(b) - f(x^{(k)})}\left(b - x^{(k)}\right)$$

Риддер әдісі

Риддер әдісі – жалған позиция әдісінің тиімді модификациясы. Түбір (x_1, x_2) аралығында деп есептей отырып, алдымен $f_3 = f(x_3)$ есептейміз, мұндағы x_3 - жақшаның ортасы, 3(а) суретте көрсетілгендей. Әрі қарай келесі функцияны енгіземіз

$$g(x) = f(x)e^{(x-x_1)Q} \qquad \text{(a)}$$

мұндағы Q тұрақтысы (x_1, g_1), (x_2, g_2) және (x_3, g_3) 3(b) суретінде көрсетілгендей түзу сызықта жатуын талап ету арқылы анықталады. Бұрынғыдай, біз қолданатын белгі $g_i = g(x_i)$. Түбірдің жақсартылған мәні содан кейін $f(x)$ емес, $g(x)$ сызықтық интерполяциясы арқылы алынады.



3-сурет. Риддер әдісінде қолданылған график.

Енді толығырақ қарастырайық. (a) теңдеуден мынаны аламыз

$$g_1 = f_1, \qquad g_2 = f_2 e^{2hQ}, \qquad g_3 = f_3 e^{hQ} \qquad (b)$$

мұндағы $h = (x_2 - x_1)/2$. 3b-суреттегі үш нүктенің түзу бойында жату талабы $g_3 = (g_1 + g_2)/2$ немесе

$$f_3 e^{hQ} = \frac{1}{2}(f_1 + f_2 e^{2hQ})$$

e^{hQ}-қа қатысты квадрат теңдеу. Шешімі:

$$e^{hQ} = \frac{f_3 \pm \sqrt{f_3^2 - f_1 \cdot f_2}}{f_2} \qquad (c)$$

(x_1, g_1) және (x_3, g_3) нүктелеріне негізделген сызықтық интерполяция енді жақсартылған түбір

$$x_4 = x_3 - g_3 \frac{x_3 - x_1}{g_3 - g_1} = x_3 - f_3 e^{hQ} \frac{x_3 - x_1}{f_3 e^{hQ} - f_1}$$

мұнда соңғы теңдікте біз (b)теңдеулерді қолдандық. Соңғы қадам ретінде біз (c) теңдеуден e^{hQ} орнына қоямыз және кішкене түрлендіруден кейін

$$x_4 = x_3 \pm (x_3 - x_1) \frac{f_3}{\sqrt{f_3^2 - f_1 \cdot f_2}} \qquad (3)$$

Дұрыс нәтиже $f_1 - f_2 > 0$ болса, қосу таңбасын және $f_1 - f_2 < 0$ болса минус таңбасын таңдау арқылы алынатынын көрсетуге болады. x_4 есептегеннен кейін түбір мен теңдеу үшін жаңа жақшалар анықталады. (3) қайтадан қолданылады.

Процедура x_4-тің екі тізбектес мәндерінің арасындағы айырмашылық тым аз болғанша қайталанады.

(3) теңдеудегі Риддердің итерациялық формуласы өте пайдалы қасиетке ие: егер x_1 және x_2түбірді қамтыса, онда x_4 әрқашан (x_1, x_2) интервалында жатады. Басқаша айтқанда, түбір жақшаға алынғаннан кейін ол жақшада қалады, бұл әдісті өте сенімді етеді. Кемшілігі - әрбір итерацияда функцияны екі рет есептеуді қажет етеді.

Риддер әдісі квадраттық жинақталатынын көрсетуге болады, бұл оны қиюшылар немесе жалған орын әдісіне қарағанда жылдамырақ етеді. Бұл әдіс $f(x)$ туындысын есептеу мүмкін емес немесе қиын болған жағдайда қолданылады.

```
## module error
''' err(string).«Жолды» басып шығарады
және бағдарламаны аяқтайды.
'''
```

ridder модулі

Төменде Риддер әдісінің бастапқы коды берілген:

```
## module ridder
''' root = ridder(f,a,b,tol=1.0e-9).
f(x) = 0 түбірін Риддер әдісімен табады.
Түбір жақшада болуы керек (a,b).
'''
import math
from numpy import sign
import sys
def err(string):
    print(string)
    input('Press return to exit')
    sys.exit()
def ridder(f,a,b,tol=1.0e-9):
    fa = f(a)
    if fa == 0.0: return a
    fb = f(b)
    if fb == 0.0: return b
    if sign(f2)!= sign(f3): x1 = x3; f1 = f3
    for i in range(30):
```

```
        # Риддер формуласынан жақсартылған x түбірін есептеу
          c = 0.5*(a + b); fc = f(c)
          s = math.sqrt(fc**2 - fa*fb)
          if s == 0.0: return None
          dx = (c - a)*fc/s
          if (fa - fb) < 0.0: dx = -dx
          x = c + dx; fx = f(x)
        # Жинақтылықты тексеру
          if i > 0:
              if abs(x - xOld) < tol*max(abs(x),1.0): return x
          xold = x
        # Түбірді мүмкіндігінше мықтап қайта бекіту
          if sign(fc) == sign(fx):
              if sign(fa)!= sign(fx): b = x; fb = fx
              else: a = x; fa = fx
          else:
              a = c; b = x; fa = fc; fb = fx
          return None
          print('Тым көп итерация')
```

МЫСАЛ 4.4

$(0.6, 0.8)$ ішінде жататын $x^3 - 10x^2 + 5 = 0$ түбірін Риддер әдісімен анықтаңыз.

Шешуі. Бастапқы нүктелер

$$x_1 = 0.6 \qquad f_1 = 0.6^3 - 10 \cdot 0.6^2 + 5 = 1.6160$$

$$x_2 = 0.8 \qquad f_2 = 0.8^3 - 10 \cdot 0.8^2 + 5 = -0.8880$$

Бірінші Итерация. Екіге бөлу мына нүктені береді

$$x_1 = 0.7 \qquad f_1 = 0.7^3 - 10 \cdot 0.7^2 + 5 = 0.4430$$

Түбірдің жақсартылған мәнін енді Риддер формуласымен есептеуге болады:

$$s = \sqrt{f_3^2 - f_1 \cdot f_2} = \sqrt{0.4430^2 - 1.6160 \cdot (-0.8880)} = 1.2738$$

$$x_4 = x_3 \pm (x_3 - x_1)\frac{f_3}{s}$$

$f_1 > f_2$ болғандықтан, біз қосу белгісін пайдалануымыз керек. Сондықтан,

$$x_4 = 0.7 + (0.7 - 0.6)\frac{0.4430}{1.2738} = 0.7348$$

$$f_4 = 0.7348^3 - 10 \cdot 0.7348^2 + 5 = -0.0026$$

Түбір анық (x_3, x_4) интервалында жатқандықтан, келесілерді қоямыз

$$x_1 \leftarrow x_3 = 0.7 \qquad f_1 \leftarrow f_3 = 0.4430$$
$$x_2 \leftarrow x_4 = 0.7348 \qquad f_2 \leftarrow f_4 = -0.0026$$

келесі итерацияның бастапқы нүктелері болып табылады.

Екінші Итерация.

$$x_3 = 0.5(x_1 + x_2) = 0.5(0.7 + 0.7348) = 0.7174$$
$$f_3 = 0.7174^3 - 10 \cdot 0.7174^2 + 5 = 0.2226$$
$$s = \sqrt{f_3^2 - f_1 \cdot f_2} = \sqrt{0.2226^2 - 0.4430 \cdot (-0.0026)} = 0.2252$$
$$x_4 = x_3 \pm (x_3 - x_1)\frac{f_3}{s}$$

$f_1 > f_2$ болғандықтан, біз қайтадан қосу белгісін қолданамыз, осылайша

$$x_4 = 0.7174 + (0.7174 - 0.7)\frac{0.2226}{0.2252} = 0.7346$$
$$f_4 = 0.7346^3 - 10 \cdot 0.7346^2 + 5 = 0.0000$$

Осылайша, түбір $x = 0.7346$, кем дегенде төрт ондық таңбаға дейін дәл.

Қолданылған әдебиеттер тізімі

1) Шакенов Қ.Қ. Есептеу математикасы әдістері лекциялар курсы.Алматы,2019. – 193б
2) P. Dechaumphai, N. Wansophark. Numerical Methods in Science and Engineering Theories with MATLAB, Mathematica, Fortran, C and Python Programs. Alpha Science International Ltd. 2022
3) Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков Численные методы: Классический университетский учебник. —М.: Издательство «Бином. Лаб. знаний»,2020. — 636 с.
4) Вабищевич П.Н. Численные методы: Вычислительный практикум. — М.: Книжный дом «ЛИБРОКОМ», 2020. — 320 с.

Интернет ресурстар

1) https://docs.python.org/3/
2) http://math-hse.info/f/2018-19/py-polit/instruction_JN.pdf
3) https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html
4) https://colab.research.google.com/
5) https://planetcalc.ru/search/?tag=2874