Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Комп'ютерний практикум № 1
Розв'язання нелінійних рівнянь

предмет «Методи обчислень»

Роботу виконав:
Студент 3 курсу ФТІ, групи ФІ-91
Цибульник Антон Владиславович

Приймала:
Стьопочкіна Ірина Валеріївна

2022

Зміст

Завдання

1. Здійснити в якості допрограмового етапу аналіз та відокремлення коренів за допомогою теорем. Зокрема, визначити кількість дійсних коренів рівняння (теорема Гюа, теорема Штурма),відокремити дійсні корені рівняння (теорема про верхню межу). До аналізу комплексних коренів застосувати теорему про кільце. Результатом цього етапу повинна бути послідовність проміжків,кожен із яких містить лише один дійсний корінь рівняння.

2. Програмний етап полягає в тому, щоб уточнити корені рівняння методом бісекції, методом хорд, методом дотичних.

3. Порівняти отримані результати, зробити висновки, який метод приводить до меншої кількості ітерацій і чим це зумовлено.

1 Відокремлення коренів

Розглядається рівняння $f(x) = 0$, де $f(x) = P_5(x) = 6x^5 - 3x^4 + x^3 + 2x^2 - 4x + 2$.

1.1 Теорема про границі усіх коренів рівняння

Теорема: *Нехай* $A = \max a_i, i = \overline{0, n-1}, B = \max a_i, i = \overline{1, n}$, *тоді всі (і комплексні теж) корені* x^* *рівняння* $P_n(x) = 0$ *лежать у такому кільці:*

$$\frac{|a_0|}{B + |a_0|} \leqslant |x^*| \leqslant \frac{|a_n| + A}{|a_n|}.$$

Отже, для заданого рівняння $P_5(x) = 6x^5 - 3x^4 + x^3 + 2x^2 - 4x + 2 = 0$ матимемо:

$$\frac{2}{2+6} \leqslant |x^*| \leqslant \frac{6+4}{6} \Rightarrow 0.25 \leqslant |x^*| \leqslant 1.66.$$

1.2 Теорема про верхню межу додатніх коренів

Теорема: *нехай* $B = \max|a_i|$ *для* $a_i < 0$, *а також* $m = \max i$ *для* $a_i < 0$. *Тоді значення* $R = 1 + \sqrt[n-m]{\frac{B}{a_n}}$ *є верхнею межею всіх додатніх коренів:* $\forall x^+ \leqslant R : P_n(x^+) = 0$. *Для визначення нижньої межі додатніх коренів стосовно вихідного поліному робимо заміну* $x = \frac{1}{y}$, *а для верхньої та нижньої межі від'ємних коренів робимо відповідні заміни* $x = -\frac{1}{y}$ *та* $x = -y$.

1. Для $P(x) = 6x^5 - 3x^4 + x^3 + 2x^2 - 4x + 2 = 0 : n = 5, m = 4, B = 4 \Rightarrow$ отримуємо значення $R = 1 + \frac{4}{6} = \frac{5}{3}$;

2. Для $P_1 = y^n P(\frac{1}{y}) = 2y^5 - 4y^4 + 2y^3 + y^2 - 3y + 6 = 0 : n = 5, m = 4, B = 4 \Rightarrow$ отримуємо значення $R_1 = 1 + \frac{4}{2} = 3$;

3. Для $P_2 = P(-y) = -6x^5 - 3x^4 - x^3 + 2x^2 + 4x + 2 = 0$. Домноживши поліном на -1 матимемо $P_2 = 6x^5 + 3x^4 + x^3 - 2x^2 - 4x - 2 = 0 : n = 5, m = 2, B = 4 \Rightarrow$ отримуємо значення $R_2 = 1 + \sqrt[3]{2/3}$;

4. Для $P_3 = y^n P(-\frac{1}{y}) = 2y^5 + 4y^4 + 2y^3 - y^2 - 3y - 6 = 0 : n = 5, m = 2, B = 6 \Rightarrow$ отримуємо значення $R_3 = 1 + \sqrt[3]{3}$;

Отже, матимемо систему нерівностей $\frac{1}{R_1} \leqslant x^+ \leqslant R$, $-R_2 \leqslant x^- \leqslant -\frac{1}{R_3}$. Таким чином $x^+ \in [0.33, 1.66]$ й $x^- \in$ [-1.87,-0.41].

1.3 Теорема про верхню межу (спосіб Лагранжа)

Теорема: *Представимо поліном $P(x)$ у вигляді $P(x) = F(x) + \Phi(x)$, де $F(x)$ містить всі поспіль старші члени рівняння з додатніми коефіцієнтами, а також всі члени з від'ємними коефіцієнтами. $\Phi(x)$ містить решту членів з додатніми коефіцієнтами. Нехай $\exists \alpha > 0 : F(\alpha) > 0$, тоді $\forall x^+ \leqslant \alpha : P_n(x^+) = 0$.*

Складемо поліноми $F(x)$ та $\Phi(x)$:

$$F(x) = 6x^5 - 3x^4 - 4x, \ \Phi(x) = x^3 + 2x^2 + 2$$

Уточнимо верхню межу додатніх коренів:

$$F(1.07) > 0 \Rightarrow x^+ \leqslant 1.07.$$

Отже, після уточнення способом Лагранжа матимемо такі проміжки коренів: $x^+ \in [0.33, 1.07]$ й $x^- \in$ [-1.87,-0.41].

1.4 Теорема Гюа про наявність комплексних коренів

Теорема: *якщо $\exists k : 0 < k < n : a_k^2 < a_{k-1} \cdot a_{k+1}$, то рівняння має хоча б одну пару комплексно спряжених коренів.*

Перевіримо систему нерівностей для відповідних значень $a_5 = 6, a_4 = -3$, $a_3 = 1, a_2 = 2, a_1 = -4$ та $a_0 = 2$.

При значенні $k = 4$ отримуємо хибну нерівність: $9 < 6$

При значенні $k = 3$ отримуємо хибну нерівність: $1 < -6$

При значенні $k = 2$ отримуємо хибну нерівність: $4 < -4$

При значенні $k = 1$ отримуємо хибну нерівність: $16 < 4$

Отже, теорема не дає змоги зробити висновки про наявність комплексно спряжених коренів.

1.5 Теорема Штурма про чередування коренів

Теорема: *нехай* $f(x) = P_n(x)$ *– поліном без кратних коренів. Утворимо таку послідовність многочленів:* $P_0(x) = P_n(x), P_1(x) = P_n'(x),$, *а для усіх наступних* $P_{i+1}(x) = -[P_{i-1}(x) \bmod P_i(x)]$, *де* $i = \overline{1, n-1}$, *тобто кожний наступний многочлен є залишком від ділення двох попередніх многочленів, взятим з протилежним знаком. Тоді кількість дійсних коренів полінома* $P_0(x)$ *на довільному відрізку* $[a, b]$ *дорівнює різниці між кількістю змін знаку (КЗЗ) у цій послідовності при* $x = a$ *та* $x = b$.

Створимо поліноми:

$$P_0(x) = 6x^5 - 3x^4 + x^3 + 2x^2 - 4x + 2$$

$$P_1(x) = 30x^4 - 12x^3 + 3x^2 + 4x - 4$$

$$P_2(x) = -[P_0(x) \bmod P_1(x)] = -\frac{1}{50}(8x^3 + 63x^2 - 156x + 96)$$

$$P_3(x) = -[P_1(x) \bmod P_2(x)] = -\frac{81374}{32}x^2 + \frac{41575}{8}x - 2975$$

$$P_4(x) = -[P_2(x) \bmod P_3(x)] = -\frac{16949504}{264875625}x + \frac{2399168}{37839375}$$

$$P_5(x) = -[P_3(x) \bmod P_4(x)] = 322,32$$

	0,33	**1,07**	**-1,87**	**-0,41**
$P_0(x)$	+	+	−	+
$P_1(x)$	−	+	+	−
$P_2(x)$	−	−	−	−
$P_3(x)$	−	−	−	−
$P_4(x)$	+	−	+	+
$P_5(x)$	+	+	+	+
КЗЗ	2	2	3	2

Табл. 1: Знаки поліномів Штурма

Отже, на проміжку додатніх значень [0.33,1.07] немає дійсних коренів, а на проміжку від'ємних значень [-1.87,-0.41] є один дійсний корінь.

2 Уточнення коренів

2.1 Метод бісекції

```
epsilon = 0.001

def func(x):
    return 6*x**5 - 3*x**4 + x**3 + 2*x**2 - 4*x + 2

def bisection(a,b):
    stop = False
    iter = 0

    while(stop != True):
        c = (a+b)/2
        if (func(a) * func(c) <= 0):
            b = c
        if (func(b) * func(c) <= 0):
            a = c

        if (abs(a-b) < epsilon):
            stop = True
            x_final = c

        iter = iter + 1

    return round(x_final,8)

print(bisection(-1.87,-0.41))
```

Результати й проміжні кроки

Ітерація 0: відрізок [-1.14,-0.41],
значення критерія |a-b| = 0.73,
наближене значення кореня = -1.14

Ітерація 1: відрізок [-1.14,-0.775],
значення критерія |a-b| = 0.365,
наближене значення кореня = -0.775

Ітерація 2: відрізок [-0.9575,-0.775],
значення критерія |a-b| = 0.1825,
наближене значення кореня = -0.9575

Ітерація 3: відрізок [-0.9575,-0.86625],
значення критерія |a-b| = 0.09125,
наближене значення кореня = -0.86625

Ітерація 4: відрізок [-0.9575,-0.911875],
значення критерія |a-b| = 0.045625,
наближене значення кореня = -0.911875

Ітерація 5: відрізок [-0.9575,-0.9346875],
значення критерія |a-b| = 0.0228125,
наближене значення кореня = -0.9346875

Ітерація 6: відрізок [-0.94609375,-0.9346875],
значення критерія |a-b| = 0.01140625,
наближене значення кореня = -0.94609375

Ітерація 7: відрізок [-0.94039063,-0.9346875],
значення критерія |a-b| = 0.00570313,
наближене значення кореня = -0.94039063

Ітерація 8: відрізок [-0.94039063,-0.93753906],
значення критерія |a-b| = 0.00285156,
наближене значення кореня = -0.93753906

Ітерація 9: відрізок [-0.93896484,-0.93753906],
значення критерія |a-b| = 0.00142578,
наближене значення кореня = -0.93896484

Ітерація 10: відрізок [-0.93896484,-0.93825195],
значення критерія |a-b| = 0.00071289,
наближене значення кореня = -0.93825195

Кількість ітерацій = 11, епсилон = 0.001, корінь = -0.93825195

2.2 Метод хорд

```
epsilon = 0.001

def func(x):
    return 6*x**5 - 3*x**4 + x**3 + 2*x**2 - 4*x + 2

def chord(a,b):
    stop = False
    iter = 0

    while(stop != True):
        c = (a*func(b) - b*func(a))/(func(b) - func(a))
        if (func(a) * func(c) <= 0):
            b = c
        if (func(b) * func(c) <= 0):
            a = c

        if (abs(func(c)) < epsilon):
            stop = True
            x_final = c

        iter = iter + 1

    return round(x_final,8)

print(chord(-1.87,-0.41))
```

Результати й проміжні кроки

Ітерація 0: відрізок [-1.87,-0.44267267],
значення критерія f(c) = 3.85867154,
наближене значення кореня = -0.44267267

Ітерація 1: відрізок [-1.87,-0.47549292],
значення критерія f(c) = 3.94745974,
наближене значення кореня = -0.47549292

Ітерація 2: відрізок [-1.87,-0.50827898],
значення критерія f(c) = 4.0147223,
наближене значення кореня = -0.50827898

Ітерація 12: відрізок [-1.87,-0.78685289],
значення критерія f(c) = 2.93877116,
наближене значення кореня = -0.78685289

Ітерація 22: відрізок [-1.87,-0.90477167],
значення критерія f(c) = 0.86741587,
наближене значення кореня = -0.90477167

Ітерація 32: відрізок [-1.87,-0.93206699],
значення критерія f(c) = 0.17113642,
наближене значення кореня = -0.93206699

Ітерація 42: відрізок [-1.87,-0.93715797],
значення критерія f(c) = 0.03078225,
наближене значення кореня = -0.93715797

Ітерація 52: відрізок [-1.87,-0.93806415],
значення критерія f(c) = 0.00544241,
наближене значення кореня = -0.93806415

Ітерація 62: відрізок [-1.87,-0.93822407],
значення критерія f(c) = 0.0009593,
наближене значення кореня = -0.93822407

Кількість ітерацій = 63, епсилон = 0.001, корінь = -0.93822407

2.3 Метод Ньютона

```
epsilon = 0.001

def func(x):
    return 6*x**5 - 3*x**4 + x**3 + 2*x**2 - 4*x + 2

def derivative_func(x):
    return 30*x**4 - 12*x**3 + 3*x**2 + 4*x - 4

def newton(a,b):
    x = []
    stop = False
    iter = 0

    x.append(a)
    x.append(x[0] - func(x[0])/derivative_func(x[0]))
    if (x[1] > b):
        x[0]=b
        x[1] = x[0] - func(x[0])/derivative_func(x[0])

    if ((abs(x[1]-x[0]) < epsilon) and (abs(func(x[1])) < epsilon)):
        stop = True
        iter = iter + 1
        x_final = x[1]

    i = 1
    while(stop != True):
        x.append(x[i] - func(x[i])/derivative_func(x[i]))
        if (func(a) * func(x[i+1]) <= 0):
            b = x[i+1]
        if (func(b) * func(x[i+1]) <= 0):
            a = x[i+1]

        if ((abs(x[i+1]-x[i]) < epsilon) and (abs(func(x[i+1])) < epsilon)):
            stop = True
            x_final = x[i+1]

        i = i + 1
        iter = iter + 1

    return round(x_final,8)

print(newton(-1.87,-0.41))
```

Результати й проміжні кроки

Ітерація 0: відрізок [-1.22775069,-0.41],
значення критерія |f(x)| = 15.47938621, а значення |x[i+1]-x[i]| = 0.27326272467081925,
наближене значення кореня = -1.22775069

Ітерація 1: відрізок [-1.04772501,-0.41],
значення критерія |f(x)| = 3.95386968, а значення |x[i+1]-x[i]| = 0.18002568265600272,
наближене значення кореня = -1.04772501

Ітерація 2: відрізок [-0.95996629,-0.41],
значення критерія |f(x)| = 0.64076614, а значення |x[i+1]-x[i]| = 0.08775872001303997,
наближене значення кореня = -0.95996629

Ітерація 3: відрізок [-0.93930792,-0.41],
значення критерія |f(x)| = 0.02951474, а значення |x[i+1]-x[i]| = 0.020658365251669863,
наближене значення кореня = -0.93930792

Ітерація 4: відрізок [-0.93826086,-0.41],
значення критерія |f(x)| = 7.275e-05, а значення |x[i+1]-x[i]| = 0.0010470586152822037,
наближене значення кореня = -0.93826086

Ітерація 5: відрізок [-0.93825827,-0.41],
значення критерія |f(x)| = 0.0, а значення |x[i+1]-x[i]| = 2.5935824589096157e-06,
наближене значення кореня = -0.93825827

Кількість ітерацій = 6, епсилон = 0.001, корінь = -0.93825827

3 Висновки

На кінець передпрограмного етапу було отримано один проміжок $x \in$[-1.87,-0.41], на якому має бути один дійсний корінь початкового рівняння

$$P(x) = 6x^5 - 3x^4 + x^3 + 2x^2 - 4x + 2 = 0.$$

Найшвидшим методом уточнення коренів на отриманому відрізку виявився метод Ньютона (лише за 6 ітерацій), а найповільнішим – метод хорд (аж 63 ітерації).