Обчислювальні методи
Лабораторна робота 2
Розв'язування нелінійних рівнянь

Увага: Інструкції що стосуються перебігу лабораторних занять складені є в частині 2 цього документу.

1 Теоретична частина

Методи розв'язування нелінійних рівнянь:

- метод бісекції
- метод неправдивого положення(превело falsi)
- метод січних
- метод дотичних (Ньютона)
- метод ітерацій простих (Банаха)
- метод Пегаса
- метод Мюллера
- метод Брента (Matlab, Octave → `fzero`)
- i in.

Таблиця 1: Ряди збіжності методів

Метод	Ряд
бісекції	1
правело falsi	1
січних	$\frac{1}{2}(1+\sqrt{5}) \approx 1.62$
Брента	1.8
Ньютона	2
Ейлера	3

1.1 Метод бісекції

Цей метод полягає на ітераційному визначенні щораз менших проміжок, що до яких маємо певність, що містять нуль функції. Закладімо, що дана є функція $f(x)$, яка є неперервна в проміжку $[a, b]$ і приймає на його берегах вартість про протележні знаки $(f(a)f(b) < 0)$. В такому випадку проміжок $[a, b]$ мусить заключати корінь функції. Наближеним рішенням можемо визнати пункт лежачий в середині проміжка $[a, b]$, тобто:

$$c = \frac{a+b}{2} \tag{1}$$

В разі коли $f(c) = 0$, нульове місце було відшукано. Однак за правилом $f(c) \neq 0$ і пошуки треба продовжувати. Нуль знаходитиметься або в проміжку $[a, c]$, або в проміжку $[c, b]$. Щоб вибрати відповідний проміжок вистачить перевірити вартість функції в пункті . Якщо $f(a)f(c) < 0$, продовжуємо процес в проміжку $[a, c]$, якщо $f(c)f(b) < 0$, означає це, що в наступній ітерації вартість треба буде підставити в місце a. Ітерацію можна перервати якщо знайдено вартість $f(c) < tol$, де tol є заданою через нас вартістю.

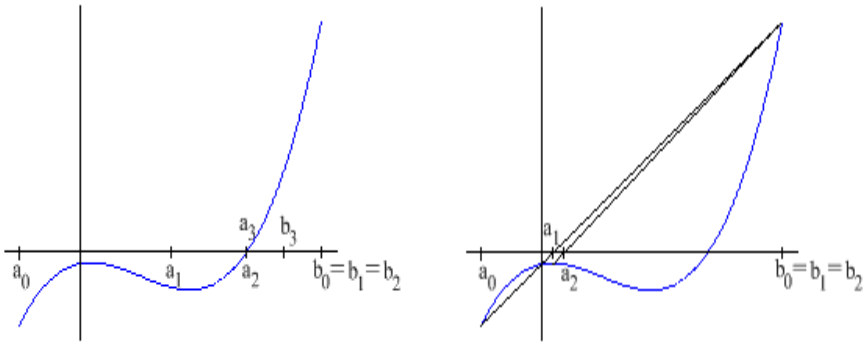
Метод бісекції є вільно збіжний(збіжність є лінійною), тому що взагалі не користується інформацією яку дає форма функції $f(x)$. Її безсумнівною перевагою натомість є те, що вона є конкретною. Bisekcja поводиться добре там, де інші (швидші) методи мають проблеми. В обрахунках нумеричних є особливо рідковживана. За те часто застосовується вона до початкового наближення проміжутка заключаючого нульвого місця функції, щоб потім скористатися швидшими методами.

```
Dane: a, b, M, δ, ϵ
Wyniki: n,x̃, f(x̃)
   fa ← f(a); fb ← f(b);
   e ← b − a;
   if sgn(fa) = sgn(fb) then
      return error;
   end if
   for n ← 1 to M do
      e ← e/2;
      c ← a + e; {c = (a + b)/2}
      fc ← f(c);
      if |e| < δ or |fc| < ϵ then
         return n, c, fc
      end if
      if sgn(fc) ≠ sgn(fa) then
         b ← c; fb ← fc;
      else
         a ← c; fa ← fc;
      end if
   end for
```

Алгоритм методу бісекції, записаний в псевдокоді.

1.2 Метод неправдивого положення(правело falsi)

Першим способом на те, щоб використати інформацію про вигляд функції, до прискорення збіжності пошуку нуля, є апроксимація функції $f(x)$ за



(a) (b)

Рис. 1: (a) Метод бісекції; (b) - правело falsi

допомогою прямої лінії. У метода regula falsi, подібно як методу połowienia починаємо від проміжка $[a, b]$, на якому здійснена є умова $f(a)f(b) < 0$. Наближеною вартістю коріня в кожному наступному кроці буде місце переходу осі OX і лінії перехідної через пункти $(a, f(a))$ і $(b, f(b))$, tzn:

$$c = b - \frac{b-a}{f(b) - f(a)} f(b) \tag{2}$$

Якщо нуль знаходиться в проміжку $[a, c]$, то змінну залишаємо без змін і підставляємо $b = c$; у протилежному випадку ми приписуємо $a = c$, у свою чергу b залишається таке ж.

Dane: a, b, M, δ, ϵ
Wyniki: n,$\tilde{x}$, $f(\tilde{x})$
  $fa \leftarrow f(a)$; $fb \leftarrow f(b)$;
  $e \leftarrow b - a$;
  if $\text{sgn}(fa) = \text{sgn}(fb)$ **then**
    return error;
  end if
  for $n \leftarrow 1$ to M **do**
    $e \leftarrow e/2$;
    $c \leftarrow b - (b-a)/(fb - fa) \cdot fb$
    $fc \leftarrow f(c)$;
    if $|e| < \delta$ **or** $|fc| < \epsilon$ **then**
      return n, c, fc
    end if
    if $\text{sgn}(fc) \neq \text{sgn}(fa)$ **then**
      $b \leftarrow c$; $fb \leftarrow fc$;
    else
      $a \leftarrow c$; $fa \leftarrow fc$;
    end if
  end for

Алгоритм правела falsi, записаний в псевдокоді.

1.3 Метод січних

Подібно як в методі regula falsi, так і в тій функція $f(x)$, є апроксимована лінією прямою. Однак метод siecznych не вимагає, щоб в кожній ітерації провіряти проміжутку, в якому знаходиться корінь. Щоб розпочати алгортм необхідні є два наближення кореня $(x0, x1)$ і не мусить бути здійснена умова $f(x0)f(x1) < 0$. Наступні апроксимовані вартості коріня знаходимо (подібно як у вищезгаданому методі) як місце перетину осі OX і лінії, що проходить через два останні апроксимації коріня $(xk-1, f(xk-1))$ і $(xk, f(xk))$.

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k) \tag{3}$$

Dane: x_{n-1}, x_n, M, δ, ϵ
Wyniki: n,$\tilde{x}$, $f(\tilde{x})$
  $f_{n-1} \leftarrow f(x_{n-1})$; $f_n \leftarrow f(x_n)$;
  for $n \leftarrow 1$ to M **do**
    $x_{n+1} = x_n - (x_n - x_{n-1})/(f_n - f_{n-1}) \cdot f_n$;
    $f_{n+1} = f(x_{n+1})$;
    $x_{n-1} \leftarrow x_n$; $f_{n-1} \leftarrow f_n$;
    $x_n \leftarrow x_{n+1}$; $f_n \leftarrow f_{n+1}$;
    if $|x_n - x_{n-1}| < \delta$ **or** $|f_n| < \epsilon$ **then**
      return n, x_n, f_n;
    end if
  end for

Алгоритм методу січних записаний в псевдокоді.

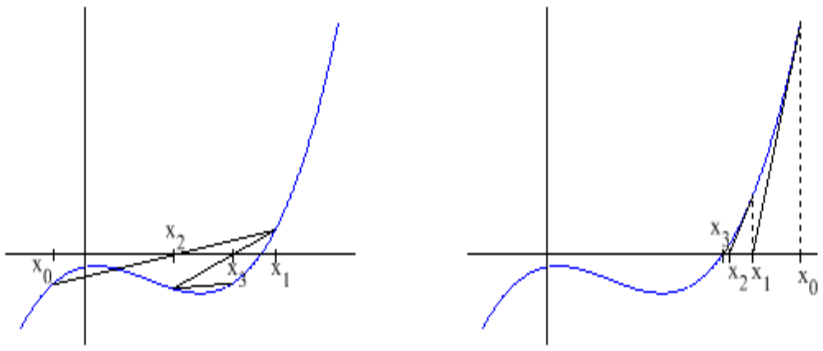
1.4 Метод дотичних/Ньютона

У методі Ньютона як апроксимацію функці $f(x)$, приймаємо дотичною до функції у x_k - цьому місці. Наступним, наближення місця нульвого, є перетин дотичної з осью OX. Мінусом такого підходу є те, що додатково є необхідне знайомоство похідної заданої функції. Метод цей також не гарантує збіжності процесу. Є вон однак найшвидшою з основних методів. Щоб облічити $k+1$-ше наближення кореня функції $f(x)$, належить застосувати формулу:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{4}$$

Dane: x_0, M, δ, ϵ
Wyniki: n,$\tilde{x}$, $f(\tilde{x})$
  $x_n \leftarrow f(x_0)$; $f_n \leftarrow f(x_n)$;
  if $|f_n| < \epsilon$ **then**
    return 0, x_n, f_n;
  end if
  for $n \leftarrow 1$ to M **do**
    $x_{n+1} \leftarrow x_n - f_n/f'(x_n)$;
    $f_{n+1} \leftarrow f(x_{n+1})$;
    if $|x_{n+1} - x_n| < \delta$ **or** $|f_n + 1| < \epsilon$ **then**
      return n, x_{n+1}, f_{n+1};
    end if
    $x_n \leftarrow x_{n+1}$;
    $f_n \leftarrow f_{n+1}$;
  end for

Алгоритм методу дотичних записаний в псевдокоді.



(a) (b)

Рис. 2: (a) метод січних; (b) - метод стичних

У протилежності до двох перших методів, методи січних і стичних не гарантують збіжності.

1.5 Умови stopu

1. $|x_n - x_{n-1}| < tol_1 \wedge |x_{n+1} - x_n| \geqslant |x_n - x_{n-1}|$
2. grube stopowanie: $|x_n - x_{n-1}| < tol_1$
3. grube stopowanie: $|f(x_n)| < tol_2$

1.6 Функції бібліотечні Octave: fzero, roots

У програмах Octave/Matlab існують дві функції які слугують до знаходження місць зерових. Перша з них це `fzero`. Імплементує вона метод Brenta-Dekkera, опублікованого в 1973 році, яка є сполученням методів bisekcji, siecznych зворотної квадратової інтерполяції. Найпростішим способом його виклику є:

```
>> fzero(fun, x0);
```

fun є заданою функцією, коріні якої шукаємо. Параметр x0, може бути вартістю скалярною (перше наближення) або також вектором двохелементовим (тоді елементи вектора **мусять** заключати корінь). У цьому другому випадку $fun(x0(1))\cdot fun(x0(2))$ мусить бути менше від нуля.

Функція

```
>> roots(c);
```

обчислює корінь многочлену, коефіцієнтами якого є елементи вектора c. Наприклад якщо нас цікавить знаходження нулів функції $x5 - 3x4 + 2x2 - 9x + 5$, тоді параметр c є вектором постаті: [1 -3 0 2 -9 5].

1.7 Завдання

У програмі Octave ознайомся з дією і способами виклику нижче поданих функцій:

- eval
- sign
- error
- disp

2 Завдання до виконання

Виконай наступні завдання:

1. Завантаж файли згідно з інструкціями і збережи їх в каталозі з назвою `Lab2`, створеним на своєму диску. Файл `main.m` буде скриптом, з якого викликатимемо всі писані через нас функції і команди.

   УВАГА: Ми не поміщаємо ці файли в головному каталозі Octave'a! Щоб могти викликати файли ми змінюємо поточний каталог (→ Laboratorium 0 → polecenia `cd, pwd, dir`)

2. Напиши функцію `fun1.m` імплементацією зразка

   $$f(x) = x^3 + x^2 - 3x - 3$$

   При написанню функції, слід ужити оператори **табличного** множення чи піднесення до степеня. У протилежному випадку спроба її виклику для векторів, поверне помилку. Виклик цієї функції поміщений усередині скрипту `main.m`.

3. Якщо `fun1.m` буде написаний правельно скрипт `main.m` повинен генерувати викрес функції $f(x)$ в проміжку $< -3, 3 >$.

4. На підставі викресу можна визначити проміжок ізоляції коріня додатнього, функції $f(x)$. Шукатимемо його за допомогою методів, обговорених на викладі. Проміжок цей вже сформульований через змінні `a`, `b` усередині скрипту `main.m`.

5. Доповни код функції, імплементуючого методу bisekcji. Знаходиться він у файлі `f_bisect.m`. Її виклик виглядає так
   `[y, yc] = f_bisect('fun', a, b, k_max, tol)`

   де

fun	функція, для якої шукаємо нульові місця
a,b	границі проміжка, в якому знаходиться корінь
tol	вимагана докладність
k_max	максимальна кількість ітерації
c	знайдено наближення кореня
yc	вартість функції для знайденого наближення кореня

   Місця, які вимагають доповнення, є відмічені в коді так як нижче:

   ```
   %%% TODO >>>
   % Obliczenie kolejnego przyblizenia pierwiastka
   % c = ..
   %%%<<<<
   ```

що означає, що лінію розпочатою від

`% c=..`

належить розкоментувати і зробити в ній імплементацію відповідного зразка, сумісного з описом в лінії вище.

Увага: Значник `%%%TODO >>` виступає в файлі `f_bisect.m` в двох місцях, тобто редагуємо дві лінії функції `f_bisect.m` .

З метою спрощення алгоритму, як критерій stopu застосовано останню умову з частини 1.5.

Зверни увагу на спосіб обчислення вартості функції, що подається як аргумент до функції `f_bisect` за допомогою функції бібліотечної `eval` . Метод той буде використаний підчас написання чергових функцій.

6. Порівняй створину функцію з псевдокодом алгоритму, поданим на лекції.

7. Скопіюй файл `f_bisect.m` і зміни назву копії на `f_rfalsi.m` . Дивлячись на код методу bisekcji заімплементуй метод regula falsi.

8. Скопіюй файл `f_rfalsi.m` і зміни назву копії на `f_secant.m`. Дивлячись на код regula falsi заімплементуй метод siecznych.

9. Скопіюй файл `f_secant.m` і зміни назву копії на `f_newton.m`. Дивлячись на код regula falsi заімплементуй метод stycznych. Функцію слід сформулювати так, щоб була сумісна з її викликом, що знаходиться в останній лінії скрипту `main`.

   В разі цього методу необхідним буде додатково створення функції, обчислюючої похудну функції `fun1`. В `main.m` функція та викликана як `dfun1`. (Пам'ятай про зміну цієї назви, якщо ти назвеш файл з функцією облічаючою похідною інакше ніж `dfun1.m`).

10. Порівняй збіжність окремих методів шукаючи нульові місця для цих самих параметрів.

11. Скористайся функціями, обговореними в пункті 1.6 (`fzero` і `roots`), щоб обрахувати корені рівняння

$$x^3 + x^2 - 3x - 3 = 0 \tag{5}$$

12. Запропонуй такі вхідні дані, для методу stycznych/siecznych, щоб для функції, що володіє місцем нульвим поблизу перших/-ого наближень/-ення, метод не був збіжний. Які ознаки функції спричинили розбіжність.

    Завдання то можна виконати модифікуючи початкове наближення для функції `fun1` або визначаючи іншу функцію в нових файлах.

13. Рівняння (5) можна перетворити до постаті, яка уможливлює використання методу простої ітерації $x = \Phi(x)$ на кілька способів, m.in.:

$$x = \frac{x^3+x^2-3}{3}\ ; \qquad x = \sqrt{-x^3+3x+3};$$

$$x = \frac{-x^3+3}{x-3}; \qquad x = (-x^2+3x+3)^{\frac{1}{3}}$$

Досліди збіжність методу простої ітерації для цих зразків, приймаючи різні початкові корені, np: $x_0 = 1$ lub $x_0 = 0.1$.