Лекція 4. Числові методи розв’язання

нелінійних рівнянь

1. Загальні визначення.

2. Метод половинного ділення (дихотомії, бісекції).

3. Метод хорд (помилкової точки, помилкового положення, regula falsi).

4. Метод Ньютона (дотичних).

5. Метод січних.

6. Метод Стефенсона.

7. Метод простої ітерації.

Математичними моделями багатьох об’єктів і процесів навколишнього світу є нелінійні рівняння і системи нелінійних рівнянь. Тому задача знаходження коренів нелінійних рівнянь виду:

|  |  |
| --- | --- |
| *F*(*x*) = 0 | (1) |

зустрічається в різних областях наукових (прикладних) досліджень (тут *F*(*x*) – деяка неперервна функція). Деяке число *x* називають *коренем рівняння* або нулем функції *F*(*x*), якщо *F*(*x*) ≡ 0.

Нелінійні рівняння можна розділити на два класи – *алгебраїчні* і *трансцендентні*.

***Алгебраїчними*** ***рівняннями*** називаються рівняння, що містять тільки алгебраїчні функції (цілі, раціональні, ірраціональні). Зокрема, многочлен є алгебраїчною функцією.

42y^4+21xy-14x^3+42xy^2-42y^2+6=0.

Рівняння, в яких невідомі входять під знаком трансцендентних функцій (тригонометричні, показникові, логарифмічні та ін.), називаються ***трансцендентними***.

Методи рішення рівнянь взагалі і зокрема нелінійних рівнянь діляться на *прямі* і *ітераційні*.

*Прямі* методи дозволяють записати корені у вигляді деякого кінцевого співвідношення (формули). З шкільного курсу алгебри читачеві відомі такі методи для вирішення тригонометричних, логарифмічних, показникових, а також простих алгебраїчних рівнянь.

Проте рівняння, що зустрічаються на практиці, не вдається вирішити такими простими методами. Для їх вирішення використовуються числові методи, основною властивістю яких є *ітеративність*, тобто це – *методи послідовних наближень*.

Процедура знаходження кореня (або декількох коренів) нелінійного рівняння за допомогою числових методів складається з наступних етапів:

1) Визначення кількості, характеру та розміщення коренів (відшукування інтервалів, що містять один корінь).

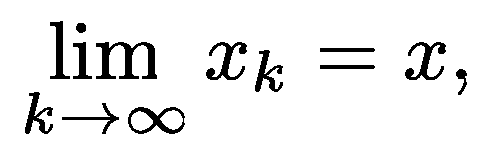
2) Знаходження наближеного значення кореня (початкового наближення).

3) Уточнення наближеного значення до деякої заданої міри точності.

***Кількість коренів та відрізки***, що містять лише один корінь, відшукуються різними способами: з фізичних міркувань, з розв’язків аналогічної задачі при інших початкових даних, за допомогою графічних методів. У деяких випадках відразу знаходять початкове наближення.

Якщо апріорні оцінки початкового наближення провести не вдається, то для ***відокремлення дійсних коренів*** можна застосувати наступний алгоритм. Інтервал області визначення функції *F*(*x*), на якому шукаються корені рівняння (1), розбивають на підінтервали (*xi*– 1, *xi*), *i* = 1, 2, …, *N*. Зміна знаку у двох сусідніх точках засвідчує наявність кореня на підінтервалі (*xi*– 1, *xi*). Таким чином знаходять дві близько розташовані точки *а* і *b*, в яких безперервна функція *F*(*x*) набуває значень різних знаків, тобто *F*(*a*) *F*(*b*) < 0. В цьому випадку між точками *а* і *b* є принаймні одна точка, в якій *F*(*x*) = 0. В якості початкового наближення *x*0 можна прийняти середину відрізку [*а*, *b*], тобто *х*0 = (*а* + *b*)/2.

***Послідовне уточнення*** початкового наближення *x*0 є ітеративним процесом. Кожний крок уточнення початкового значення називається ітерацією (англ. iteration — *повторення*). В результаті ітерацій знаходиться послідовність наближених значень кореня *x*1, *x*2, …, *xk*, … Якщо ці значення із зростанням *k* прямують до істинного значення кореня



то говорять, що ітераційний *процес сходиться*.

Далі розглянемо деякі ітеративні методи розв’язання нелінійних рівнянь з однією невідомою.

## Метод половинного ділення (дихотомії, бісекції)

Це один з найпростіших методів знаходження коренів нелінійних рівнянь. Він полягає в наступному. Припустимо, що нам вдалося знайти відрізок [*а*, *b*], на якому розташовано шукане значення кореня *х* = *x*, тобто *x* ϵ [*а*, *b*]. В якості початкового наближення кореня *x*0 приймаємо середину цього відрізку: *x*0 = (*а* + *b*)/2. Далі досліджуємо значення функції *F*(*x*) на кінцях відрізків [*а*, *x*0] і [*x*0, *b*], тобто. у точках *а*, *x*0, *b*. Той з відрізків, на кінцях якого *F*(*x*) набуває значень різних знаків, містить шуканий корінь; тому його приймаємо як новий відрізок [*a*1, *b*1]. Другу половину відрізку [*a*, *b*], на якій знак *F*(*x*) не міняється, відкидаємо. В якості першого наближення кореня приймаємо середину нового відрізку *x*1 = (*а*1 + *b*1)/2 і т. д. Таким чином, *k*-е наближення обчислюється як

|  |  |
| --- | --- |
|  | (2) |

після кожної ітерації відрізок, на якому розташований корінь, зменшується удвічі, а після *k* ітерацій він скорочується в 2*k* разів:

|  |  |
| --- | --- |
|  | (3) |

Нехай наближене рішення *x*\* необхідно знайти з точністю до деякого заданого малого числа ε > 0:

|  |  |
| --- | --- |
|  | (4) |

Взявши як наближений розв’язок *k*-е наближення кореня: *x*\* = *xk*, запишемо (4) у виді

|  |  |
| --- | --- |
|  | (5) |

Як легко бачити, з (2) витікає, що (5) виконано, якщо

|  |  |
| --- | --- |
|  | (6) |

Таким чином, ітеративний процес треба продовжувати до тих пір, поки не буде виконано умову (6).

Метод ділення відрізку навпіл проілюстрований на рис. 1. Нехай для визначеності *F*(*a*) < 0, *F*(*b*) > 0. В якості початкового наближення кореня приймемо *x*0 = (*а* +*b*)/2. Оскільки в даному випадку *F*(*x*0) < 0, то *x* ϵ [*x*0, *b*], і розглядаємо тільки відрізок [*x*0, *b*], тобто *a*1 = *x*0, *b*1 = *b*. Наступне наближення: *x*1 = (*a*1 + *b*1)/2. При цьому відрізок [*x*1, *b*1] відкидаємо, оскільки *F*(*x*1) > 0 і F(*b*1) > 0. Таким чином, *x* ϵ [*x*0, *x*1], *a*2 = *x*0, *b*2 = *x*2. Аналогічно знаходимо наступні наближення: *x*2 = (*x*0 + *x*1)/2 і т. д. до виконання умови (6).

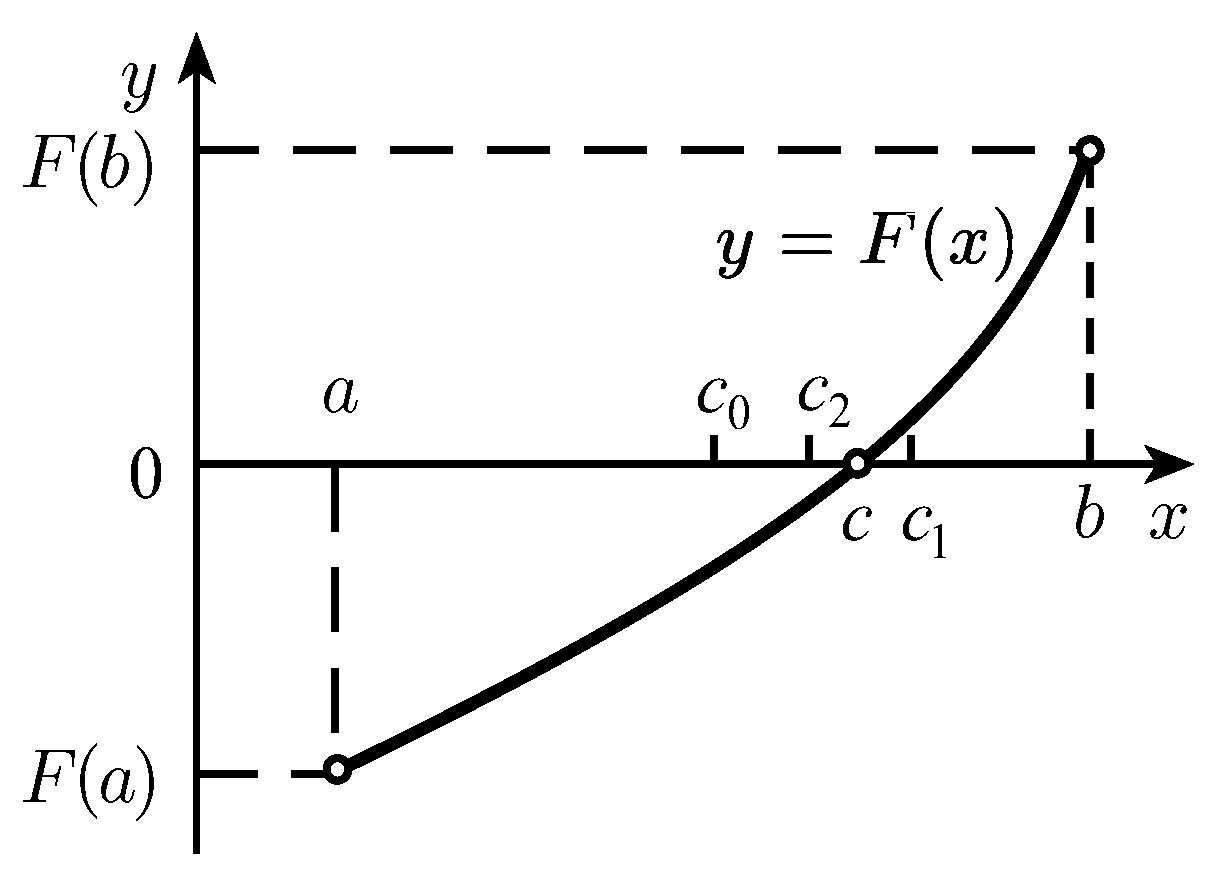


Рис. 1. Метод ділення відрізку навпіл

На відміну від більшості інших ітераційних методів метод ділення відрізку навпіл завжди сходиться, причому можна гарантувати, що отримане рішення матиме будь-яку наперед задану точність (зрозуміло, у рамках розрядності комп'ютера).

При застосуванні цього методу для визначення моменту досягнення необхідної точності і зупинки ітерацій застосовується співвідношення (6), що гарантує виконання (4).

Метод ділення відрізку навпіл досить повільний. Обчислимо кількість ітерацій *N*, необхідне для досягнення точності ε. Для цього з'ясуємо, користуючись (3), для яких *k* виконана умова (6), і візьмемо як *N* найменше з таких *k*. Остаточно отримаємо

|  |  |
| --- | --- |
|  | (7) |

де *Е*(*х*) – ціла частина числа *х*. Зазвичай для методу ділення відрізку навпіл *N* більше, ніж для деяких інших методів, що не є перешкодою до застосування цього методу, якщо кожне обчислення значення функції *F*(*x*) нескладне.

В загальному випадку існує необхідність приблизно визначати момент досягнення заданої точності. При цьому використовують такі умовами близькості двох послідовних наближень:

|  |  |
| --- | --- |
|  | (8) |

або

|  |  |
| --- | --- |
|  | (9) |

Ітеративний процес можна завершувати і тоді, коли значення функції *F*(*x*) після *k*-ї ітерації стане меншим по модулю ε, тобто:

|  |  |
| --- | --- |
|  | (10) |

Значення функції *F*(*x*) при знаходженні коренів рівняння (1) часто називають *нев’язкою* (похибкою розв’язку).

нет

да

да

нет

початок

ввод

*a*, *b*, 

Обчислення

*F(a)*



Обчислення

*F*(*хk*)



вивод

*хk*



Зсув лівої границі

*а*=*хk*

Зсув правої границі

*b* = *xk*

кінець

Рис. 2. Метод бісекції. Блок-схема алгоритму

Розглянемо приклад написання програми MATLAB, яка дозволяє отримати наближений розв’язок нелінійного рівняння *F*(*x*) = 0 на інтервалі [*a*, *b*] методом ділення навпіл.

Використаємо наведений на рис. 2 алгоритм. Оскільки в MATLAB немає циклу do...while, який присутній в даному алгоритмі, його необхідно перетворити перед написанням програми так, щоб використовувався цикл while...do (рис. 3).

нет

да

да

нет

початок

ввод

*a*, *b*,

*Fk=Fa*



Обчислення

*Fk=F*(*хk*)



вывод

*хk*



*а* = *хk*

*b* = *xk*

конец

Обчислення

*Fa=F(a)*

Рис. 3. Метод бісекції. Блок-схема алгоритму модифікована

Нижче наведений приклад реалізації даного метода в MATLAB.

|  |
| --- |
| Файл-функція dichotomy.m |
| function x = dichotomy(f, a, b, e)    Fa = feval(f, a);  Fx = Fa;  while abs(Fx) > e  xk = (a + b) / 2;  Fx = feval(f, xk);  if Fa\*Fx > 0  a = xk;  else  b = xk;  end  end  x = xk;  end |

Приклад розв’язання рівняння

x + 0.1 \* (ln(1 + x) - x2 + 9)=0.

|  |
| --- |
| Файл-функція my\_func.m |
| function y = my\_func(x)  y = x + 0.1 \* (log(1 + x) - x^2 + 9); |

|  |
| --- |
| Command Window |
| **>> dichotomy('my\_func',-0.8,0.0,0.001)**  ans = -0.7203 |

Приклад реалізації метода дихотомії в лабораторній роботі (дещо складніший):

|  |
| --- |
| function [c,err,yc]=bisect(f,a,b,delta)  %Ввод - f - функція, вводиться як рядок 'f'  % а і b - ліва и права крайні точки  % delta - допустиме відхилення  %Вихід - с - розв'язок  % yc=f(c)  % err - похибка с    ya=feval(f,a);  yb=feval(f,b);  if (ya\*yb>0)  return  end  maxI=1+round((log(b-a)-log(delta))/log(2));  for k=1:maxI  c=(a+b)/2;  yc=feval(f,c);  if (yc==0)  a=c;  b=c;  elseif yb\*yc>0  b=c;  yb=yc;  else  a=c;  ya=yc;  end  if (b-a < delta)  break;  end  end  c=(a+b)/2;  err=abs(b-a);  yc=feval(f,c); |

## Метод хорд (помилкової точки, помилкового положення, regula falsi)

Метод хорд за своєю процедурою і необхідними вихідними даними подібний до методу ділення навпіл але в ряді випадків збігається швидше.

Припустимо, що ми знайшли відрізок [*а*, *b*], на якому функція *F*(*x*) міняє знак і містить шукане значення кореня *х* = *x*, тобто *x* ϵ [*а*, *b*]. В якості початкового наближення кореня *x*0 приймаємо точку, в якій хорда, що з’єднує точки (*a*, *F*(*a*)) і (*b*, *F*(*b*)), перетинається з віссю *x* (рис. 3). Вираз для знаходження *x*0 має наступний вид:

|  |  |
| --- | --- |
|  | (11) |

Далі, як і в методі дихотомії, досліджуємо значення функції *F*(*x*) на кінцях відрізків [*а*, *x*0] і [*x*0, *b*], тобто. у точках *а*, *x*0, *b*. Той з відрізків, на кінцях якого *F*(*x*) набуває значень різних знаків, містить шуканий корінь. Тому його приймаємо як новий відрізок [*a*1, *b*1]. Для прикладу рис. 3 відкидаємо другу половину відрізку [*a*, *b*], на якій знак *F*(*x*) не міняється і границі наступного інтервалу будуть *a*1 = *a*, *b*1 = *x*0. Наступне наближення кореня знову обчислюється за (11). Таким чином, *k*-е наближення обчислюється як:

|  |  |
| --- | --- |
|  | (12) |

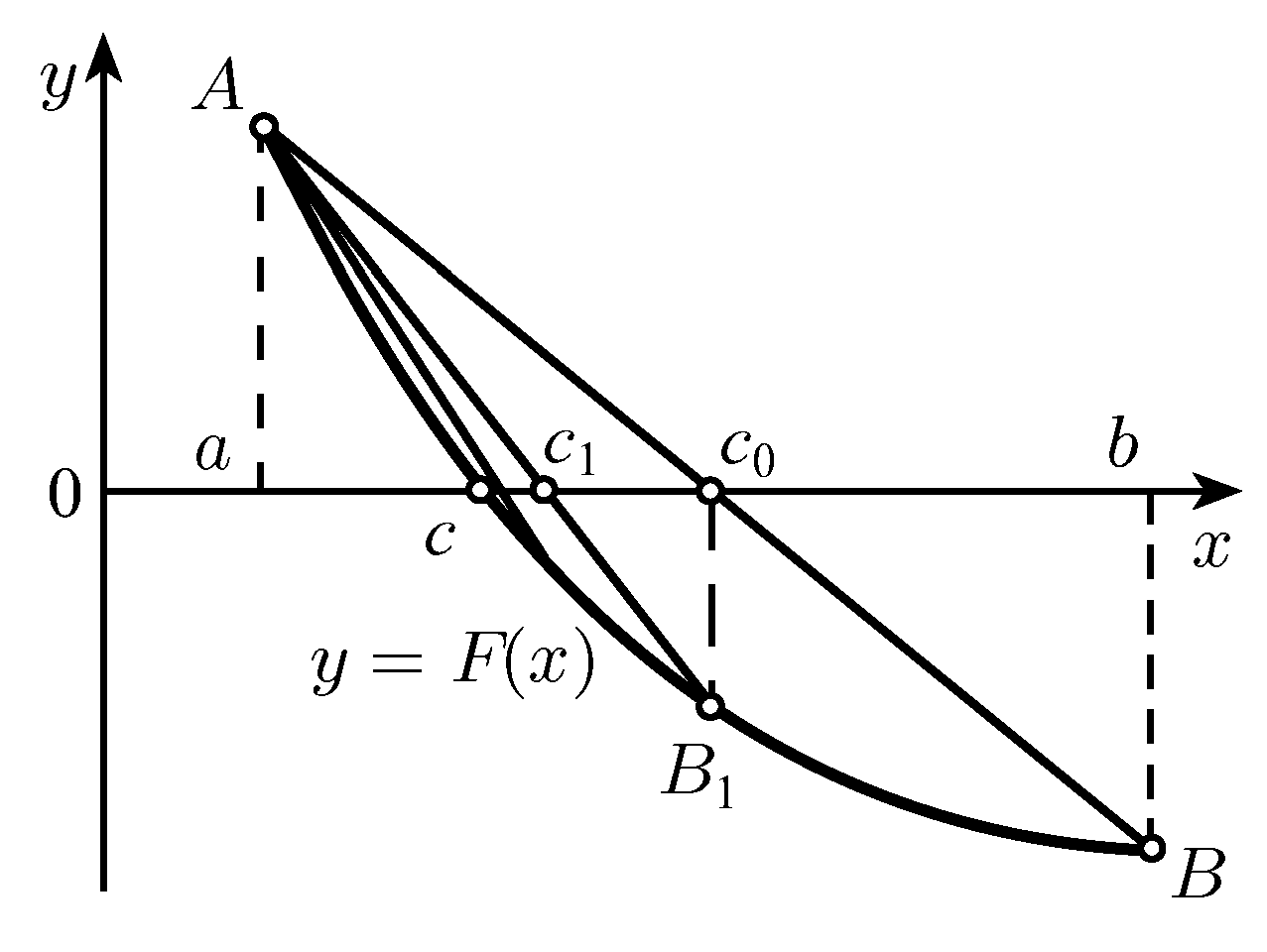


Рис. 4. Метод хорд

На відміну від методу ділення навпіл в методі хорд умову завершення ітерацій типу (6) застосувати неможливо. На рис. 4 видно, що довжина відрізку [*a*, *xk*] ніколи не стане менше довжини відрізку [*a*, *x*]. В даному методі використовують одну з умов (8)–(10).

Як видно методи хорд та ділення навпіл досить схожі. Однією з характерних особливостей розглянутих методів є те, що обидва методи не вимагають знання додаткової інформації про функцію *F*(*x*). Наприклад, не вимагається, щоб функція була диференційована. Неперервність *F*(*x*) гарантує успіх застосування цих методів. Складніші методи розв’язання нелінійних рівнянь використовують додаткову інформацію про функцію *F*(*x*), передусім властивість диференційованості функції. Як результат вони зазвичай мають швидшу збіжність, але в той же час, можуть бути застосованими для вужчого класу функцій, і їх збіжність не завжди гарантована. Прикладом такого методу служить метод Ньютона.

Приклад

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Уточнити корінь рівняння , що лежить на проміжку [–1, 0], методом хорд з точністю до 0,001.  Розв’язання  Маємо функцію .  Для обчислень використаємо формулу  , де .  Результати обчислень розміщуємо в таблиці.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | *k* | *xk* | *f*(*xk*) | *f*(*xk*) – *f*(*a*) | *xk*– *a* |  | | 0  1  2  3  4 | 0  –0,882  –0,943  –0,946  –0,946 | 1,5  0,2173  0,0121  0,0014 | 1,7  0,4173  0,2121  0,2014 | 1  0,118  0,057  0,054 | –0,118  –0,057  –0,054  –0,054 |   Відповідь.*x* ≈ –0.946 |

## Метод Ньютона (дотичних)

Якщо *F*(*x*), *F*'(*x*) і *F*"(*x*) *неперервні* *в околі кореня* *x*, цю додаткову інформацію про властивості функції *F*(*x*) можна використати для побудови алгоритмів, які породжують послідовності {*x*k}, що *сходяться* до *x* *швидше*, ніж при методі ділення навпіл або методі хорд. Метод Ньютона (чи Ньютона-Рафсона) є одним з найкорисніших і найвідомішим алгоритмом, в якому використовується неперервність *F*'(*x*) і *F*"(*x*).

Відмінність методу Ньютона від попереднього методу полягає в тому, що на *k*-й ітерації замість хорди *проводиться дотична* до кривої *у* = *F*(*x*) при *х* = *xk*–1 і шукається точка перетину дотичної з віссю абсцис. При цьому не обов'язково задавати відрізок [*а*, *b*], що містить корінь рівняння (1), а досить лише знайти деяке початкове наближення кореня *х* = *x*0 (рис. 5).

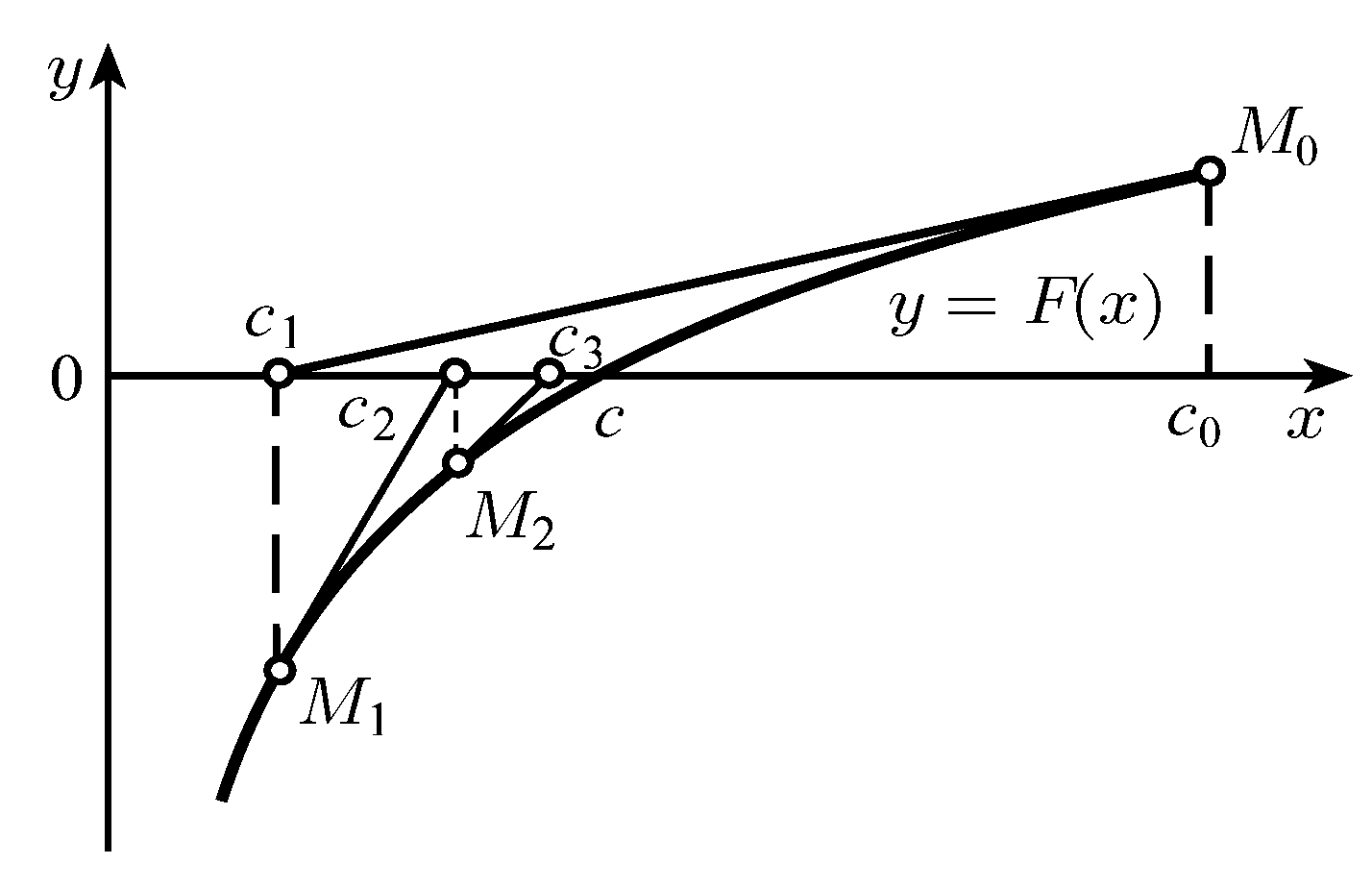


Рис. 5. Метод дотичних

З рівняння дотичної до кривої *у* = *F*(*x*) в точці (*x*0, *F*(*x*0)) можна знайти *перше наближення кореня*:

|  |  |
| --- | --- |
|  |  |

Аналогічно можуть бути знайдені *наступні* *k-ті наближення*:

|  |  |
| --- | --- |
|  | (13) |

при цьому необхідно, щоб *F*'(*xk*) *не дорівнювала нулю*. Для завершення ітеративного процесу можуть бути використані умови (8)–(10).

З (13) витікає, що на кожній ітерації *об'єм обчислень* в методі Ньютона *більший*, ніж в розглянутих раніше методах, оскільки доводиться знаходити значення не лише функції *F*(*x*), але і її похідної. Проте *швидкість* збіжності тут значно *вище*, ніж в інших методах.

На відміну від методів ділення навпіл і хорд, які мають лінійну збіжність, метод Ньютона має *квадратичну збіжність*, тобто:

|  |  |
| --- | --- |
|  | (14) |

*Збіжність* і ефективність методу Ньютона *залежить* від вдалого вибору *початкового* *наближення* *x*0, тому на практиці цей вибір може бути досить складним. Для запобігання розбіжності іноді доцільно використати змішаний алгоритм. Він полягає в тому, що спочатку застосовується метод, що завжди сходиться (наприклад, метод ділення відрізку навпіл), а після деякого числа ітерацій – метод Ньютона, що швидко сходиться.

Приклад

Уточнити один з коренів (*x*0 = –1) методом Ньютона (дотичних) з точністю 0,001:

*x*3 – 0,2*x*2 + 0,5*x* + 1,5 = 0.

Знаходимо похідну:

*f*'(*x*) = 3*x*2 – 0,4*x* + 0,5.

Для обчислень використовуємо таблицю:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *n* | *xn* | *f*(*xn*) | *f*'(*xn*) |  |
| 0 | –1 | –0,2 | 3,9 | –0,051 |
| 1 | –0,949 | –0,0093 | 3,5814 | –0,0026 |
| 2 | –0,9464 | –0,0004 | 3,5657 | –0,00001 |

Відповідь: *x* = –0,946.

Схема алгоритму метода Ньютона (рис. 6).

да

нет

нет

начало

ввод

*a*, *b*,

*xk = a* або *xk = b*

або 

*Fk=F*(*xk*)

**

*dFk =* 0



вивод

*хk*

кінець



да

Рис. 6. Метод дотичних. Блок-схема алгоритму

В лабораторних роботах цей метод реалізовано за дещо іншим алгоритмом (рис. 7).

нет

початок

ввод

*x*0, ε, maxI

*F=F*(*x*0)

**



вивод

*хk*

кінець



да

k=1 до maxI

*x*0 = *x*1

*F*(*x*0)

Рис. 7. Метод дотичних. Блок-схема алгоритму

Приклад реалізації в MATLAB

|  |
| --- |
| newton.m |
| function [p0, err, k, y]=newton(f, df, p0, delta, epsilon, maxI)  % Вхід - f - функція, вводиться як рядок 'f'  % df - похідна f, вводиться як рядок 'df'  % р0 - початкове наближення кореня функции f  % delta - допустиме відхилення для р0  % epsilon - допустимое відхилення для значення функції f(р0)  % maxI - максимальне число ітерацій  %Вихід - р0 - наближення кореня за методом Ньютона  % err - похибка обчислення для р0  % k - число ітерацій  % У - значення функції f(р0)  for k=1:maxI  p1=p0-feval(f,p0)/feval(df,p0);  err=abs(p1-p0);  relerr=2\*err/(abs(p1)+delta);  p0=p1;  y=feval(f,p0);  if (err<delta) || (relerr<delta) || (abs(y)<epsilon)  break;  end  end |

Приклад розв’язання того самого рівняння

x + 0.1 \* (ln(1 + x) - x2 + 9) = 0.

|  |
| --- |
| Файл-функція my\_func.m |
| function y = my\_func(x)  y = x + 0.1 \* (log(1 + x) - x^2 + 9); |

Функція розрахунку похідної:

|  |
| --- |
| Файл-функція my\_func\_diff.m |
| function y = my\_func\_diff(x)  y = 0.1/(x + 1) - 0.2\*x + 1.0; |

Виконання команд

|  |
| --- |
| Command Window |
| **>> newton('my\_func', 'my\_func\_diff', -0.8, 0.001, 0.001, 100)**  ans = -0.7206 |

## Метод січних

Застосовуючи формулу (13) методу Ньютона, необхідно точно обчислювати похідні від розглянутих функцій, бажано аналітично. На практиці під час обчислень похідні апроксимують розділеними першими різницями:

|  |  |
| --- | --- |
|  | (15) |

Використовуючи апроксимацію (15) у формулі Ньютона (13), можна перейти до методу січних:

|  |  |
| --- | --- |
|  | (16) |

Щоб реалізувати цей метод, на відміну від методу Ньютона необхідні два початкових значення, за допомогою яких здійснюється локальна апроксимація функції січними (рис. 4)

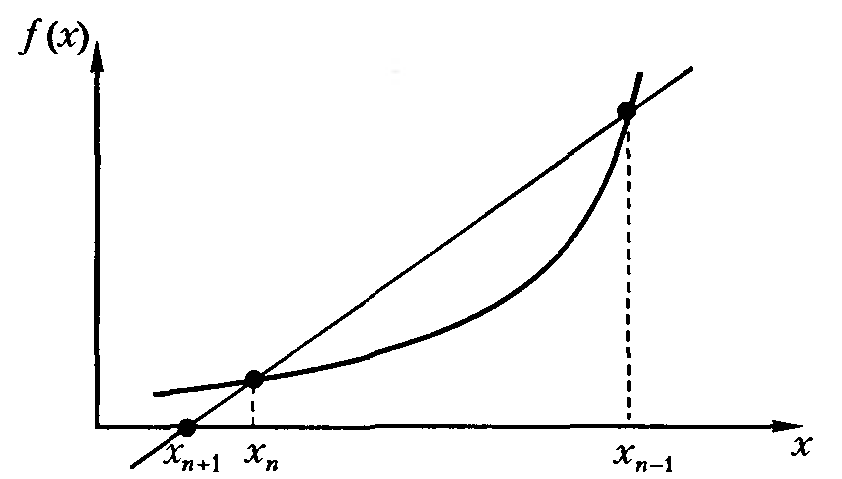


Рис. 4. Метод січних

Збіжність методу хорд дещо повільніша, ніж методу Ньютона, але набагато швидша, ніж для методу дихотомії. Можна показати, що збіжність методу Ньютона має показник :

|  |  |
| --- | --- |
|  | (17) |

## Метод Стефенсона

Природно, що були запропоновані методи, в яких не потрібно обчислювати похідну (як у методі хорд), але збіжність яких квадратична (як у методі Ньютона). Наприклад, таким є метод Стефенсона, відповідно до якого

|  |  |
| --- | --- |
|  | (18) |

і

|  |  |
| --- | --- |
|  | (19) |

## Метод простої ітерації

Для використання цього методу вихідне рівняння (1) записується в канонічному виді:

|  |  |
| --- | --- |
| *x* = *f*(*x*). | (20) |

Нехай відоме початкове наближення кореня *x* = *x*0. Підставляючи це значення в праву частину рівняння (20), отримаємо наступне наближення:

|  |  |
| --- | --- |
| *x*1 = *f*(*x*0). |  |

Підставляючи кожний раз нове значення кореня в (20), отримаємо послідовність значень

|  |  |
| --- | --- |
| *xk* = *f*(*xk*–1), *k* = 1, 2, … | (21) |

Ітераційний процес завершується, якщо результати двох послідовних ітерацій близькі, тобто виконується умова (8).

Даний метод не є абсолютно збіжним, як, наприклад, метод дихотомії. Достатньою умовою збіжності методу простої ітерації, за якої |*εk*+1| буде менше |*εk*|, є умова

|  |  |
| --- | --- |
| |*f*'(*x*)| < 1 | (22) |

І чим меншим буде значення |*f*'(*x*)|, тим швидше буде збігатись ітераційний процес.

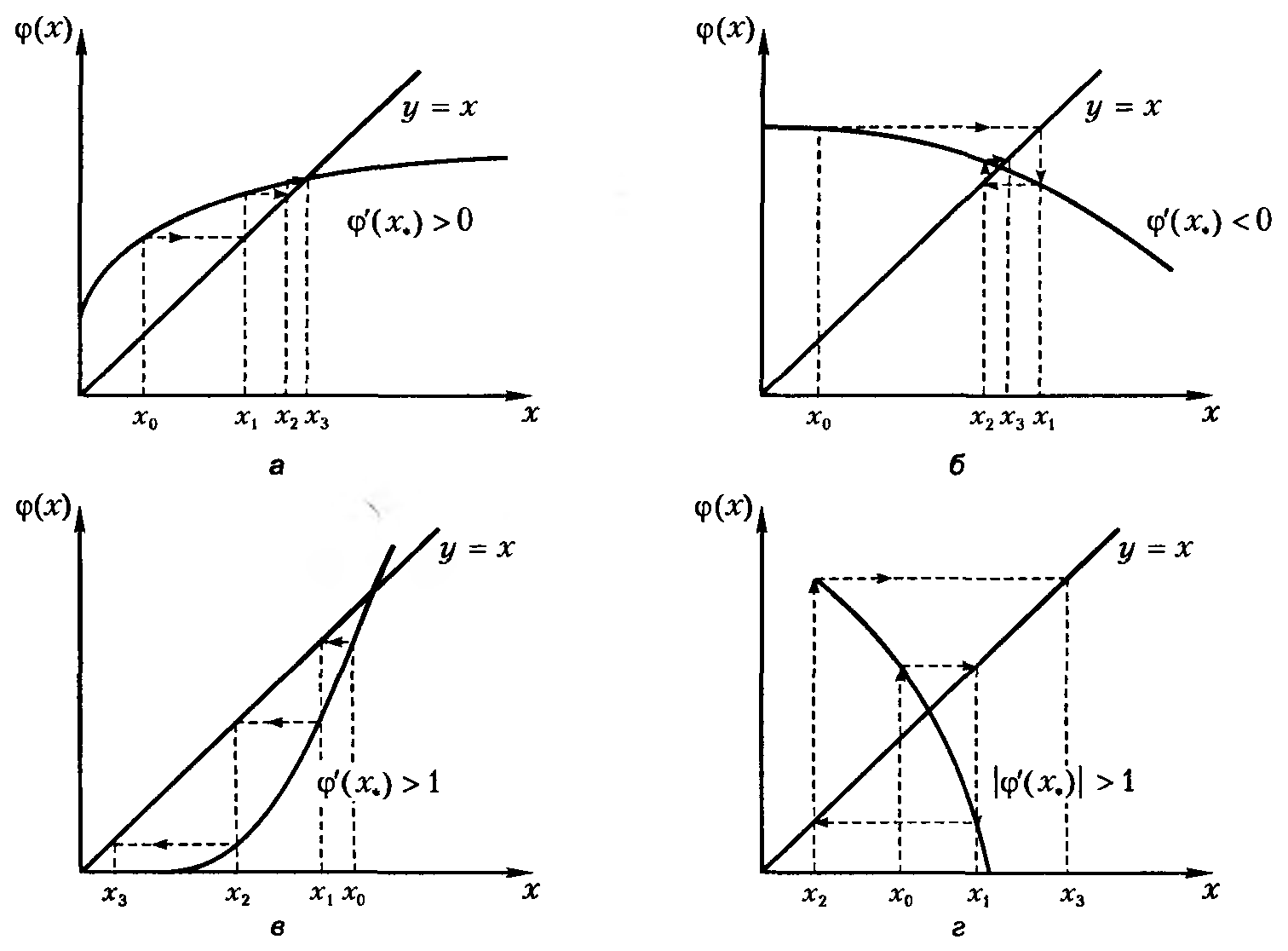


Рис. 5. Можливі варіанти збіжності ітерацій: *а* – монотонна збіжність, *б* – коливальна збіжність,

*в* – монотонна розбіжність, *г* – коливальна розбіжність

Метод простої ітерації розглянутий для рівняння виду (20). До такого виду можна привести і загальне рівняння (1):

|  |  |
| --- | --- |
| *F*(*x*) = 0, *τF*(*x*) = 0,  *x* = *x* – *τF*(*x*). | (23) |

Тут *τ*≠ 0 – деяке число. Рівняння (23) еквівалентне рівнянню (20) з функцією *f*(*x*) = *x* – *τF*(*x*). За рахунок вибору значення параметра *τ* можна добиватись збіжності метода простої ітерації і підвищення швидкості збіжності. Наприклад, якщо на деякому відрізку, що містить корінь рівняння, похідна *F*'(*x*) обмежена константами *m* і *M* для забезпечення умови збіжності (22) параметр можна взяти *τ* = 2/(*M* + *m*).

Параметр *τ* можна взяти і змінним, залежним від номеру ітерації. Так, якщо покласти *τk* = 1/*F*'(*xk*), то метод простої ітерації для рівняння (23) прийме вид, що співпадає з рівнянням методу Ньютона. Отже метод Ньютона можна трактувати як частинний випадок методу простої ітерації зі змінним *τ*.

Приклад

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Знайдемо корінь рівняння      Початкове рівняння можна замінити еквівалентним        *x*0 = 0.1   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | *i* | 1 | 2 | 3 | 4 | 5 | 6 | | *xi* | 0.9751 | 0.6784 | 0.7780 | 0.7438 | 0.7555 | 0.7515 | |

|  |  |
| --- | --- |
|  | () |

Приклад

|  |
| --- |
|  |