TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC
School of Applied Mathematics and Informatics

Báo cáo Giải tích số

Chủ đề Phương pháp lặp đơn giải f(x)=0

Giảng viên hướng dẫn: Hà Thị Ngọc Yến
Nhóm sinh viên thực hiện: Nhóm 5

Vũ Thị Tâm	20185403
Nguyễn Thành Đạt	20185334
Nguyễn Thị Nga	20185387
Nguyễn Thị Thủy	20185412
Hoàng Đức Minh Triều	20185415
Đoàn Quốc Vĩnh	20185427

Mục lục

1 Bài toán đặt ra

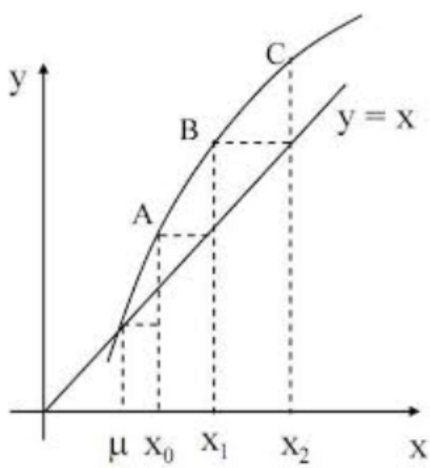
Giải phương trình $f(x) = 0$. (1)
Giả sử (a, b) là khoảng phân ly nghiệm α của phương trình $f(x) = 0$ (tức là $f(x) = 0$ có nghiệm duy nhất $\alpha \in (a, b)$), đồng thời $f(x)$ liên tục và có đạo hàm liên tục trên đoạn $[a, b]$. Cần một phương pháp để tìm ra nghiệm đúng hoặc gần đúng của phương trình.

2 Ý tưởng phương pháp

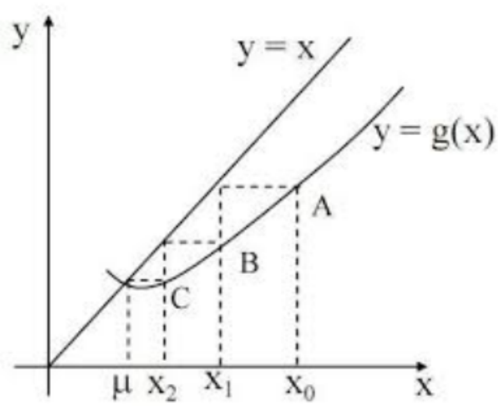
MỤC TIÊU : Tìm nghiệm gần đúng với sai số cho phép ε.
XÂY DỰNG CÔNG THỨC

- Biến đổi tương đương $f(x) = 0 \Leftrightarrow x = g(x)$, với $g(x)$ là hàm liên tục trên khoảng (a, b)
- Chọn giá trị $x_0 \in (a, b)$
- Tính $x_1 = g(x_0), x_2 = g(x_1), \ldots .x_k = g(x_{k-1}) k \to \infty \Rightarrow$Thu được dãy x_n Nếu x_n hội tụ thì $\exists \lim\limits_{n\to\infty} x_n$.
- Từ đó ta thu được công thức lặp $x_n = g(x_{n-1})$

Ý NGHĨA HÌNH HỌC: Từ phương trình $x = g(x)$, sử dụng công thức lặp trên, tìm được nghiệm của phương trình hoành độ giao điểm của 2 đồ thị $y = x$ và $y = g(x)$.



Hình a
Không hội tụ đến nghiệm α



Hình b
Hội tụ đến nghiệm α

3 Sự hội tụ của phương pháp

- Quá trình lặp lại được gọi là hội tụ, nếu dãy x_n tính theo công thức trên hội tụ, tức là $\exists \lim\limits_{n\to\infty} x_n = x^*$.

- Nếu quá trình lặp hội tụ, nghĩa là $\lim\limits_{n\to\infty} x_n = x^*$.

- Nhưng do $g(x)$ là hàm số liên tục trên $[a, b]$ nên

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} g(x_{n-1}) = g[\lim_{n\to\infty} x_{n-1}]$$

  Từ đó suy ra $x^* = g(x^*)$

- Vậy khi đó $x^* = \alpha$ là nghiệm của phương trình $x = g(x)$ hay của phương trình $f(x) = 0$.

3.1 Định lý 1

Định lý 1: Định lý ánh xạ co

Cho không gian mêtric đầy đủ $\mathbb{X}$. Cho $f : \mathbb{X} \to \mathbb{X}$.
Nếu $\exists\ 0 \le \alpha < 1$ sao cho $\forall x, y$ ta có $d(f(x), f(y) \le \alpha.d(x, y)$. Khi đó $\exists\ x_0$ duy nhất thỏa mãn $f(x_0) = x_0$, và nếu ta xét dãy x_n như sau $x_2 = f(x_1), x_3 = f(x_2), \ldots$ thì x_n hội tụ về x.

3.2 Định lý 2

Định lý 1: Định lý ánh xạ co

Cho không gian mêtric đầy đủ $\mathbb{X}$. Cho $f : \mathbb{X} \to \mathbb{X}$.
Nếu $\exists\ 0 \le \alpha < 1$ sao cho $\forall x, y$ ta có $d(f(x), f(y) \le \alpha.d(x, y)$. Khi đó $\exists\ x_0$ duy nhất thỏa mãn $f(x_0) = x_0$, và nếu ta xét dãy x_n như sau $x_2 = f(x_1), x_3 = f(x_2), \ldots$ thì x_n hội tụ về x.

Định lý 2

Giả sử (a, b) là một khoảng phân ly nghiệm α của phương trình $f(x) = 0$ hay $x = g(x)$, $g(x)$ là hàm liên tục và có đạo hàm trên $[a, b]$, $g(x) \in [a, b]$. Với bất kỳ $x_n \in [a, b]$, nếu $\exists\ q \ge 0$ không phụ thuộc vào x sao cho:

$$|g'(x)| \le q < 1,\ \exists x \in [a, b] \qquad (1)$$

Thì quá trình lặp hội tụ tới nghiệm đúng α của phương trình.

α là nghiệm của phương trình $f(x) = 0$ nên ta có $\alpha = g(x)$
Theo công thức lặp $x_n = g(x_{n-1})(2)$ ta có $x_1 = g(x_0)$. Theo định lý Lagrange:
$x_1 - \alpha = g'(c_1)(x_0 - \alpha)$
trong đó c_1 là trị trung gian giữa x_0 và α.Từ đó theo điều kiện (1):
$|x_1 - \alpha| \leq q|x_0 - \alpha|$.
Tương tự ta cũng có:
$|x_2 - \alpha| \leq q|x_1 - \alpha|$;
....
$|x_n - \alpha| \leq q|x_{n-1} - \alpha$;
Nhân các bắt đẳng thức trên vế với vế:
$|x_n - \alpha| \leq q^n|x_0 - \alpha|$;
Do $0 \leq q \leq 1$ nên khi $n \to \infty$ thì $x_n \to \alpha$
hay $\lim\limits_{n\to\infty} x_n = \alpha$
Vậy với các điều kiện của định lý 2 thì quá trình lặp (2) hội tụ và hội tụ tới nghiệm đúng α của phương trình $f(x) = 0$.

4 Đánh giá sai số

4.1 Công thức sai số

Giả sử α là nghiệm đúng của phương trình $f(x) = 0$ và $x_n$là nghiệm gần đúng tính theo công thức $x_n = g(x_{n-1})$. Khi $|x_n - \alpha|$ được đánh giá theo công thức:

$$|x_n - \alpha| \leq \frac{q}{1-q}|x_n - x_{n-1}|$$

hoặc

$$|x_n - \alpha| \leq \frac{q^n}{1-q}|x_1 - x_0|$$

Công thức 1 thuận lợi khi tính trên máy tính
Muốn nghiệm gần đúng x_n đạt sai số ε, ta kiểm tra điều kiện dừng:
$|x_n - x_{n-1}| \leq \delta = \dfrac{\varepsilon(1-q)}{q}$
Công thức 2 thuận lợi trong việc tìm nghiệm gần đúng x_n đạt sai số ε cho trước.
Cụ thể, từ đánh giá $|x_n - \alpha| \leq \dfrac{q^n}{1-q}|x_1 - x_0| \leq \varepsilon$
Ta suy ra số n nhỏ nhất thỏa mãn:
$n \leq \log_q \dfrac{\varepsilon(1-q)}{q}$

4.2 Chứng minh công thức sai số

Công thức sai số hậu nghiệm
Ta xét :$|x_n - \alpha| \leq q|x_{n-1} - \alpha| = q|(x_{n-1} - x_n) + (x_n - \alpha)|$
$\Rightarrow |x_n - \alpha| \leq q|x_{n-1} - x_n| + q|x_n - \alpha|$
$\Rightarrow (1-q)|x_n - \alpha| \leq q|x_{n-1} - x_n|$
$\Rightarrow |x_n - \alpha| \leq \dfrac{q}{1-q}|x_n - x_{n-1}|$
Trong trường hợp $0 \leq q \leq \dfrac{1}{2}$ ta được:
$|x_n - \alpha| \leq |x_n - x_{n-1}|$
Công thức sai số tiên nghiệm
Ta có $x_n = g(x_{n-1}); x_{n-1} = g(x_{n-2})$.

$$x_n - x_{n-1} = g(x_{n-1}) - g(x_{n-2})$$

Theo định lý Lagrange:

$$x_n - x_{n-1} = g'(c)(x_{n-1} - g(x_{n-2})$$

Với c là giá trị nằm giữa x_{n-1} và x_{n-2}, hay:

$$|x_n - x_{n-1}| = g'(c)(x_{n-1} - x_{n-2})| \leq q|(x_{n-1} - x_n)|$$

Thay vào công thức 1 và truy hồi ta có:

$$|x_n - \alpha| \leq \frac{q}{1-q}|x_n - x_{n-1}| \leq \frac{q^2}{1-q}|x_{n-1} - x_{n-2}| \leq ... \leq \frac{q^n}{1-q}|x_1 - x_0|$$

5 Thuật toán

5.1 Gradient Descent

Giới thiệu:

- Điểm local minimum x^* của hàm số là điểm có đạo hàm $f'(x^*)$ bằng 0. Hơn thế nữa, trong lân cận của nó, đạo hàm của các điểm phía bên trái là không dương, đạo hàm của các điểm phía bên phải là không âm.
- Đường tiếp tuyến với đồ thị hàm số đó tại 1 điểm bất kỳ có hệ số góc chính bằng đạo hàm của hàm số tại điểm đó.

Giả sử x_t là điểm ta tìm được sau vòng lặp thứ t. Ta cần tìm một thuật toán để đưa x_t về càng gần x càng tốt:

- Nếu đạo hàm của hàm số tại $x_t : f'(x_t) > 0$ thì x_t nằm về phía bên phải so với x(và ngược lại). Để điểm tiếp theo x_{t+1} gần với x hơn, chúng ta cần di chuyển x_t về phía bên trái, tức về phía âm. Nói các khác, chúng ta cần di chuyển ngược dấu với đạo hàm:

  $$x_{t+1} = x_t + \Delta$$

  Trong đó Δ là một đại lượng ngược dấu với đạo hàm $f'(x_t)$
- x_t càng xa x về phía bên phải thì $f'(x_t)$ càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển Δ, một cách trực quan nhất, là tỉ lệ thuận với $f'(x_t)$

5.2 Thuật toán lặp đơn ngôn ngữ tự nhiên

Bước 1: Tìm khoảng phân ly nghiệm (a, b) bằng phương pháp đồ thị

Bước 2: Biến đổi hàm $f(x) = 0$ về $x = g(x)$. Đặt $q = \max\limits_{x \in [a,b]} |g'(x)|$

Bước 3: Nếu $q \leq 1$ thì chuyển tới bước 4, ngược lại quay về bước 2.

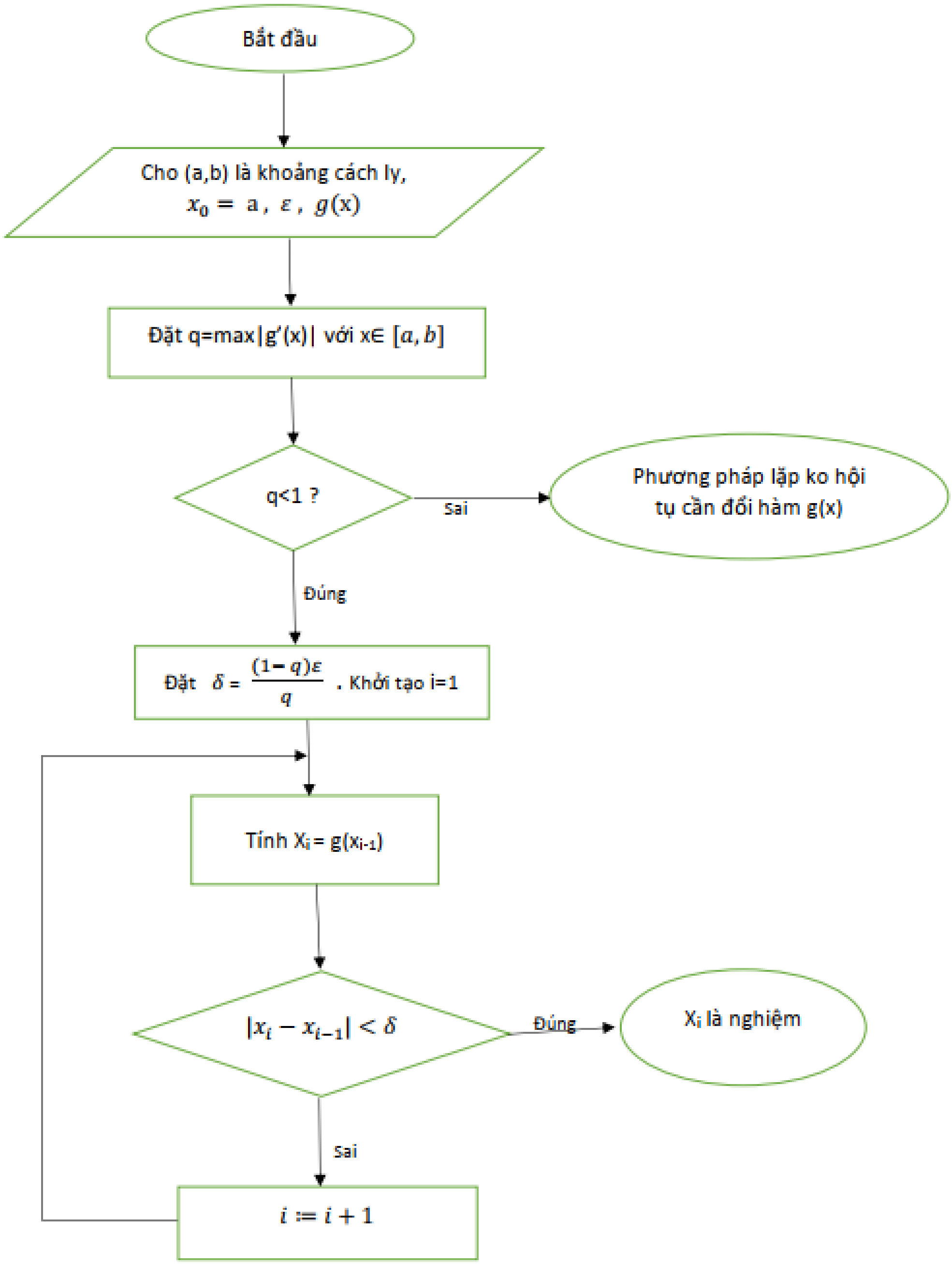
Bước 4: Đặt $\delta = \dfrac{(1-q)\varepsilon}{q}$. Khởi tạo $i = 0$.

Bước 5: Tăng $i = i + 1$, tính $x_i = g(x_{i-1})$.

Bước 6: Nếu $|x_i - x_{i-1}| \leq \delta$ thì chuyển đến bước 7, ngược lại quay lại bước 5.

Bước 7: Kết luận x_i là nghiệm cần tìm.

5.3 Sơ đồ khối biểu diễn phương pháp lặp đơn



5.4 Giả mã

Input: $x_0, \varepsilon, g(x), q = \max\limits_{[a,b]} |g(x)'|$

Output: x_1

if $q \geq 1$ then return NaN
else
begin
$\delta = \dfrac{1-q}{q}\varepsilon;$
loop = 1
while (true)
begin
$x_1 = f(x_0)$
if ($abs(x_1 - x_0) \leq \delta$ then break;
$x_0 = x_1$
loop = loop+1
end
return x_1, loop
end

6 Chương trình và ví dụ

6.1 Xây dựng hàm $f(x)$, $g(x)$ và đạo hàm gradient

```python
import math
import numpy as np
import matplotlib.pyplot as plt
# Hàm f(x) = x^3 + x - 100
def f(x):
    return x**3 + x - 1000
    #return x**5 - x - 1

# Hàm g(x) = (1000-x)^(1/3)
def g(x):
    return (1000-x)**(1/3)
    #return (x+1)**(1/5)

# Hàm gDeri = (-1)^n x g'(x)
def gDeri(x,n):
    return ((-1)**n)*(-1/3*(1000-x)**(-2/3))
    #return 1/5*((x+1)**(-4/5))

# Đạo hàm cho hàm gDeri
def deri(x,n):
    dy = gDeri(x+10e-9,n)-gDeri(x-10e-9,n)
    dx = 2*10e-9
    return dy/dx
```

6.2 Vẽ đồ thị minh họa

```python
# Vẽ đồ thị
x = np.linspace(5,12,100)
y = f(x)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

plt.plot(x,y, 'r')
plt.show()

# INIT
epsilon = 1e-6 # Sai số của bài toán
distance = [9,10] # Chọn khoảng cách ly nghiệm cho bài toán '''
#distance = [1,2]
start = distance[0]
end = distance[1]

# Thuật toán Gradient dencents
def gd(distance, n):
    x = [distance[0]]
    for i in range(100):
        x_new = x[-1] - 0.1*deri(x[-1],n)
        if abs(deri(x_new,n)) < 1e-3:
            break
        x.append(x_new)
    return x[-1]
```

6.3 Hàm tìm cực tiểu, cực đại

```python
# Tìm cực tiểu
def minima(distance, value):
    n = 0
    a = gd(distance, n)
    distance[0] = a + 10e-3
    b = gDeri(a, n)
    if a >= start and a <= end:
        value.append(b)
    return distance, value

# Tìm cực đại
def maxima(distance, value):
    n = 1
    a = gd(distance, n)
    distance[0] = a + 10e-3
    b = gDeri(a, 0)
    if a >= start and a <= end:
        value.append(b)
    return distance, value

# Tìm tất cả cực trị trong khoảng phân ly nghiệm
def ext(distance, value):
    if distance[0] >= distance[1]:
        return value
    else:
        if deri(distance[0], 0) < 0:
            distance, value = minima(distance, value)
        else:
            distance, value = maxima(distance, value)
        return ext(distance, value)
```

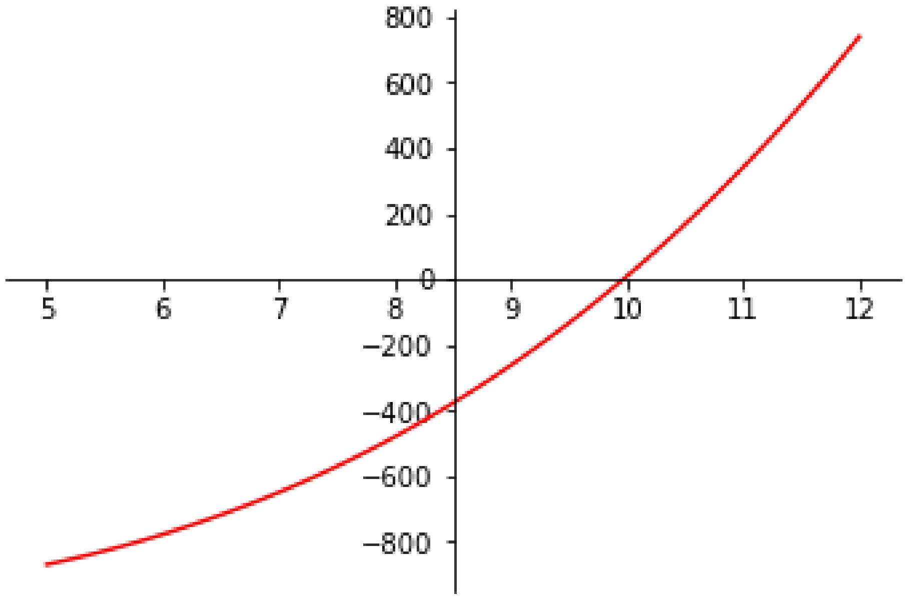
6.4 Hàm tính q hệ số co

```python
# Tìm max của  |g(x)'| trong khoảng phân ly nghiệm
def maxAbsolute():
    value =[]
    value.append(gDeri(distance[0],0))
    value.append(gDeri(distance[1],0))

    value = ext(distance, value)
    value.sort()

    # Công thức nhanh tính giá trị max của |g(x)'|
    value_max = (abs(value[0]+value[-1]) + abs(value[0]-value[-1])) /

    return value_max

q = maxAbsolute()

# Phương pháp lặp đơn với sai số hậu nghiệm
def repeatMethod1(x0, epsilon):
    if q >= 1:
        print("Điều kiện hội tụ không thỏa mãn")
    else:
        epsilon_0 = ((1 - q)*epsilon) / q
        loop = 1
        while(1):
            x1 = g(x0)
            print("loop = {}, x1 = {}, x0 = {}, f(x)= {}".format(loop,
            if abs(x1-x0) < epsilon_0:
                break
            x0 = x1
            loop = loop + 1
        print("Nghiệm gần đúng của phương trình là: {}".format(x1))

# Thực hiện chương trình
repeatMethod1(start, epsilon)
```

6.5 Kết quả chạy

Ví dụ 1: $x^3 + x - 100$

```
 loop = 1, x1 = 9.969909547282054, x0 = 9, f(x)= 0.9699095472819863
loop = 2, x1 = 9.96665590891103, x0 = 9.969909547282054, f(x)= -0.003253638371575107
loop = 3, x1 = 9.96666682705008, x0 = 9.96665590891103, f(x)= 1.0918138968918356e-05
Nghiệm gần đúng của phương trình là: 9.96666682705008
```



Đồ thị

Ví dụ 2: $x^5 - x - 1$

```
loop = 1, x1 = 1.148698354997035, x0 = 1, f(x)= -0.1486983549970342
loop = 2, x1 = 1.1652928729386878, x0 = 1.148698354997035, f(x)= -0.016594517941652898
loop = 3, x1 = 1.167087262645595, x0 = 1.1652928729386878, f(x)= -0.0017943897069074843
loop = 4, x1 = 1.1672806328403909, x0 = 1.167087262645595, f(x)= -0.00019337019479581663
loop = 5, x1 = 1.1673014634946366, x0 = 1.1672806328403909, f(x)= -2.0830654245074243e-05
loop = 6, x1 = 1.1673037073719947, x0 = 1.1673014634946366, f(x)= -2.2438773576993754e-06
Nghiệm gần đúng của phương trình là: 1.1673037073719947
```

Đồ thị