**Mục lục**

[I. Sơ đồ hoocne 2](#_Toc104528128)

[II. Miền chứa nghiệm của đa thức 2](#_Toc104528129)

[1. Bán kính nghiệm 2](#_Toc104528130)

[2. Cận trên và cận dưới 2](#_Toc104528131)

[III. Ý tưởng phương pháp 3](#_Toc104528132)

[IV. Thuật toán 3](#_Toc104528133)

[1. Thuật toán tổng thể 3](#_Toc104528134)

[a. Bằng chữ: 3](#_Toc104528135)

[2. Thuật toán chi tiết 4](#_Toc104528136)

[a. Hàm tính giá trị f(x) 4](#_Toc104528137)

[b. Hàm tính giá trị đạo hàm của f(x) 5](#_Toc104528138)

[c. Hàm tìm miền chứa nghiệm của đa thức f(x) 5](#_Toc104528139)

[d. Thuật toán chia đôi giải phương trình f(x) = 0 trong kcl (c, d) 7](#_Toc104528140)

[e. Giải phương trình đa thức f(x) = 0 7](#_Toc104528141)

[VI. Đánh giá thuật toán 11](#_Toc104528142)

[1. Ưu điểm 11](#_Toc104528143)

[2­­. Nhược điểm 11](#_Toc104528144)

Phương trình đa thức

# I. Sơ đồ hoocne

# II. Miền chứa nghiệm của đa thức

## 1. Bán kính nghiệm

Cho phương trình đa thức p(x) = a0xn + a1xn – 1 + … + an (a0 0)

Thì ta sẽ có nghiệm của phương trình sẽ nằm trong đường tròn tâm O bán kính

Trong đó:

## 2. Cận trên và cận dưới

Cho phương trình đa thức p(x) = a0xn + a1xn – 1 + … + an (a0 0)

Thì ta có các nghiệm dương của phương trình thỏa mãn

Trong đó: B là max trị tuyệt đối của các hệ số âm và k là vị trí ak âm đầu tiên tính từ

Nếu muốn xác định cận dưới thì đặt x = -x rồi tính như cách tìm cận trên

Nhược điểm: Không phân biệt được khi nào phương trình vô nghiệm

# III. Ý tưởng phương pháp

- Tìm bán kính chứa nghiệm của phương trình đa thức thông qua công thức bán kính nghiệm

- Dùng gói min-max để tìm khoảng cực trị trong miền nghiệm vừa tìm

- Hai điểm cực trị liền nhau sẽ tạo thành một điểm ly nghiệm, sử dụng các phương pháp gần đúng để tìm nghiệm trong mỗi khoảng li nghiệm của phương trình đa thức

# IV. Thuật toán

## 1. Thuật toán tổng thể

### a. Bằng chữ:

input: eps, n, a[]

trong đó: n là bậc của đa thức, a[] là mảng lưu hệ số của đa thức

output: tất cả các nghiệm của phương trình đa thức

Bước 1: nhập input

Bước 2: Nếu n < 0 thì báo n không thỏa mãn và kết thúc chương trình, ngược lại thì chuyển sang bước 3

Bước 3: tính là bán kính nghiệm của hệ phương trình và gán c = -R, d = R

Trong đó:

(c, d) là miền chứa nghiệm của phương trình đa thức

Bước 4: Dùng thuật toán gradien descent để tìm các khoảng cực trị của phương trình đa thức và lưu vào mảng survey[]

Bước 5: nếu khoảng cực trị đang xét thỏa mãn khoảng li nghiệm thì chuyển sang bước 7, ngược lại thì chuyển sang bước 6

Bước 6: Nếu phần tử hiện tại trong survey[] chưa phải là phần tử cuối thì chuyển sang phần tử tiếp theo của mảng và quay lại bước 5, ngược lại thì chuyển sang bước 8

Bước 7: Dùng phương pháp chia đôi để giải nghiệm gần đúng của phương trình đa thức rồi quay lại bước 6.

Bước 8, in ra output

## 2. Thuật toán chi tiết

Function nhap\_DT:

for i = 0 to n:

nhập a[i]

### a. Hàm tính giá trị f(x)

input: x, n, a[n]

output: giá trị của f(x) tại x

Function f:

sum = 0

for i = 0 to n:

sum = sum + a[i] \* xn – i

return sum

### b. Hàm tính giá trị đạo hàm của f(x)

input: x, n, a[n]

output: giá trị của đạo hàm f(x) tại x là df(x)

Function df:

sum = 0

for i = 0 to n - 1:

sum = sum + a[i] \* (n – i) \* xn – 1 - i

return sum

### d. Thuật toán chia đôi giải phương trình f(x) = 0 trong kcl (c, d)

input: n, a[n], c, d, epsi

output: nghiệm gần đúng x

function bisection:

do:

mid = (c + d) / 2

if f(a, n, mid) = 0:

return mid

if f(a, n, mid) \* f(a, n ,c) > 0:

c = mid

else:

d = mid

denta = |d – c|

while (denta > epsi)

return mid

### e. Giải phương trình đa thức f(x) = 0

input: n, a[n], epsi

output: các nghiệm của phương trình đa thức f(x) = 0

Function solve:

eta = e-7

k = 0

survey[n] //Mảng để chứa 2 các cực trị và miền chứa nghiệm

if n = 0:

if a[0] = 0:

print “phương trình đa thức vô số nghiệm”

else:

print “phương trình đa thức vô nghiệm”

if n = 1:

print “Nghiệm duy nhất là: -a[1]/a[0]

domain\_solution(a, n, low, up)

if low = up:

if f(a, n, low) = 0:

print “Nghiệm là:” low

else:

print “Phương trình vô nghiệm”

else:

//Thuật toán Gradient Descent tìm cực trị

x1 = low

x0 = x1

k = 1

while (x0 < up):

if df(f, x0) > 0:

sign = 1

else:

sign = -1

eta = 10-7

x1 = x0 + sign \* eta \*df(f, x0)

while (|df(f, x1)| > epsi):

if df(f, x0) \* df(f, x1) > 0:

while eta < 0.008:

eta = eta \* 2

x1 = x0 + sign \* eta \* df(f, x0)

if df(f, x1) \* df(f, x0) < 0

eta = eta / 2

thoát vòng lặp while

else:

while eta > 0:

eta = eta / 2

x1 = x0 + sign \* eta \*df(f, x0)

if df(f, x1) \* df(f, x0) > 0:

thoát vòng lặp while

x1 = x0 + sign \* eta \* df(f, x0)

x0 = x1

eta = 10-7

Thêm x1 vào mảng survey: survey[k] = x1

x1 = x1 + 0.001

k = k + 1

Thêm low và up vào mảng survey[]: survey[0] = low

survey[k] = up

// Bắt đầu tìm nghiệm của phương trình đa thức

for i = 0 to k – 1:

value1 = f(a, n, survey[i])

value2 = f(a, n, survey[i + 1])

if |value1| <= epsi:

print “nghiệm là: “ survey[i]

if |value2| <= epsi:

print “nghiệm là: “ survey[i+1]

else:

if value1 \* value2 < 0:

print “nghiệm là: “

bisection(a, n, survey[i], survey[i+1], epsi)

# VI. Đánh giá thuật toán

1. Ưu điểm

* Độ phức tạp thuật toán thấp hơn
* Tốc độ tìm ra các nghiệm của phương trình đa thức nhanh hơn
* Thuật toán GD có sử dụng eta động giúp việc tìm ra các cực trị nhanh hơn rất nhiều và hiệu quả hơn dùng eta tĩnh

1. Nhược điểm

* Khó cài đặt thuật toán trên máy tính hơn so với phương pháp chia nhỏ miền nghiệm
* Một số đa thức với độ dốc đồ thì quá lớn thì tìm rất lâu và có thể không tìm ra được