THỰC HÀNH TOÁN CAO CẤP

*TÀI LIỆU PHỤC VỤ SINH VIÊN NGÀNH KHOA HỌC DỮ LIỆU*

Nhóm biên soạn: TS. Hoàng Lê Minh – Khưu Minh Cảnh – Hoàng Thị Kiều Anh – Lê Thị Ngọc Huyên – …

TP.HCM – Năm 2019

**MỤC LỤC**

[CHƯƠNG 6: HÀM SỐ, DÃY VÀ MỘT SỐ ỨNG DỤNG 3](#_Toc21050)

[1. Giới thiệu lập trình đệ quy trong Python 3](#_Toc21051)

[2. Dãy số (sequence) đệ quy 4](#_Toc21052)

[3. Revisit: Phương pháp Gradient Ascent/Descent 11](#_Toc21053)

[3.1. Chương trình tổng quát về Gradient Ascent 11](#_Toc21054)

[3.2. Các lưu ý về giá trị khởi tạo ban đầu (initial value) 14](#_Toc21055)

[3.3. Vai trò của kích thước bước nhảy (step\_size) và Epsilon 16](#_Toc21056)

[3.4. Lời bàn 20](#_Toc21057)

[BÀI TẬP CHƯƠNG 6 21](#_Toc21058)

# CHƯƠNG 6: HÀM SỐ, DÃY VÀ MỘT SỐ ỨNG DỤNG

**Mục tiêu:**

* *Lập trình đệ quy trong Python;*
* *Dãy số/chuỗi số;*
* *Revisit phương pháp gradient ascent/descent.*

**Nội dung chính:**

# Giới thiệu lập trình đệ quy trong Python

Trong toán học, nhiều công thức có thể được thể hiện một phần bằng chính nó. Ví dụ: ! = 1 × 2 ×. .× , góc độ khác, điều này có nghĩa là ! = × − 1 ! Do đó, để tính được ! thì chúng ta phải tính được − 1 !,… và cuối cùng người ta định nghĩa 0! = 1. Từ đó, đệ quy là một kỹ thuật lập trình để viết một hàm mà trong hàm lại có lệnh gọi lại chính nó để đáp ứng những định nghĩa toán học.

**Thực hành 1: Viết hàm đệ quy tính n!**

**>>> import math**

**>>> def giaithua(n):**

**if (n==0):**

**return 1**

**else:**

**return n\*giaithua(n-1)**

>>> giaithua(3)

………………………………6……………………….  sinh viên điền kết quả

>>> giaithua(4)

……………………………24………………………….  sinh viên điền kết quả

**Thực hành 2: Viết hàm đệ quy tính dãy số Fibonacci**

Dãy số Fibonacci là dãy số có 2 phần tử ban đầu là 0 và 1, sau đó, các phần tử tiếp theo sẽ là tổng của hai phần tử gần nó nhất. Cụ thể:

= − 1 + − 2 , ≥ 2, 0 = 0, 1 = 1

Khi đó, chúng ta có thể viết chương trình đệ quy như sau:

>>> def fibo(n):

if n == 0:

return 0

if n == 1:

return 1

return fibo(n-1) + fibo(n-2)

Sinh viên có thể thử nghiệm các giá trị:

>>> fibo(4)

……………………3…………………………..

>>> fibo(5)

……………………5…………………………..

>>> fibo(6)

……………………8…………………………..

# Dãy số (sequence) đệ quy

Tổng quát, một dãy số đệ quy là một dãy được cho dưới dạng toán học sau:

= , = , , , , …(

= ộ á !ị #$% !ướ#

Giá trị của )\* + được tính theo giá trị )\*. Do đó, dãy số hội tụ khi hiệu )\* + − )\* dần về 0 khi → ∞. Điều này suy ra, ít nhất phương trình . / = / (hàm . tương ứng với dãy )\*) có nghiệm. Ta gọi các nghiệm đó là điểm bất động (fixed points).

**Quy trình tìm giới hạn của các dãy dạng đệ quy như sau:**

***Bước* 1: Giải phương trình tìm nghiệm là các điểm bất động** 0 = 0**.**

***Bước* 2: Nếu dãy số không phải là dãy hằng thì xác định khoảng** , 1 **thỏa và** 1 **đều là điểm bất động và vùng có điểm cuối (end point) (có thể là** ±∞**) thỏa không tồn tại điểm bất động trong** , 1 **và giá trị ban đầu của dãy nằm trong khoảng** , 1**.**

***Bước* 3: Kiểm tra khi nào hàm sẽ ánh xạ vào trong khoảng** , 1 **ở bước 2.**

***Bước* 4: Nếu ở bước 3 kiểm tra nằm trong** , 1 **thì:**

**Nếu dãy** { } **là đơn điệu thì hoặc** 567→8 = 1 **hoặc** 567→8 = **.**

**Nếu không nằm trong** , 1 **thì kiểm tra nằm trong** , **, nếu** < ) < **hoặc kiểm tra nằm trong** , 1 **, nếu** < < :**.**

Sử dụng đệ quy hãy viết chương trình hỗ trợ dự đoán giới hạn của một số dãy số sau:

**Thực hành 3: Chứng minh dãy có giới hạn và tìm giới hạn của dãy**

)\* + = )\*<=

; 1 (

)+ = 2

**Giải:**

Xây dựng chương trình đệ quy:

>>> def an\_exp\_an(n):

if n == 1: return 1.0/2

else:

return an\_exp\_an(n-1)\*\*an\_exp\_an(n-1)

Sinh viên thực hiện một số thử nghiệm:

>>> an\_exp\_an(1)

……………… 0.5………………..  sinh viên ghi kết quả

>>> an\_exp\_an(2)

………………… 0.7071067811865476……………..  sinh viên ghi kết quả

>>> an\_exp\_an(3)

……………… 0.7826540273556803………………..  sinh viên ghi kết quả

>>> an\_exp\_an(4)

……………… 0.8254713306050621………………..  sinh viên ghi kết quả

>>> an\_exp\_an(5)

……………… 0.8535714578150946………………..  sinh viên ghi kết quả

>>> an\_exp\_an(10)

………………… 0.9176411981636667……………..  sinh viên ghi kết quả

Do việc tính toán cần nhiều thời gian nên chúng ta không thể tính toán đến giá trị lớn như n=50,... Tuy vậy, chúng ta có thể tận dụng các kết quả trên để nhận định đây là **dãy tăng hay giảm**? ………………tăng…………  sinh viên trả lời.

Xét hàm . / = />, / > 0 tương ứng với dãy )\* + = . )\*

Bước 1: Tìm các điểm bất động với phương trình . / = /:

/> = / ⟺ /A / = ln / ⟺ ln / / − 1 = 0 ⟺ / = 1

Bước 2: Nhận định vị trí liên quan đến điểm bất động và vùng biến thiên của hàm . là: 0,1 Bước 3: Xét hàm . trong khoảng 0,1 , ta có:

0 < . / = /> = D>E\*> < DF = 1 GớH 0 < / < 1

Lưu ý: do ln / < 0 khi 0 < / < 1 nên /A / < 0 khi 0 < / < 1

Bước 4: Từ kết quả trên, chúng ta có thể kết luận dãy {)\*} là dãy tăng và nằm trong khoảng 0,1 . Do đó, ta có thể kết luận giới hạn của dãy là: 567→8 =

**Thực hành 4: Chứng minh dãy có giới hạn và tìm giới hạn của dãy**

5

I)\* + = . )\* = 6 − )\* , = 1,2,3, …(

)+ = 4

Giải:

Xây dựng chương trình đệ quy và xét một số giá trị:

>>> def bai4(n):

if n==1:

return 4

else:

return 5.0/(6-bai4(n-1))

>>> bai4(5)

………………… 1.019108280254777……………..  sinh viên ghi kết quả

>>> bai4(6)

………………… 1.0038363171355498……………..  sinh viên ghi kết quả

>>> bai4(7)

………………… 1.000767852572306……………..  sinh viên ghi kết quả

>>> bai4(10)

………………… 1.0000061439905628……………..  sinh viên ghi kết quả

>>> bai4(100)

……………… 1.0………………..  sinh viên ghi kết quả

Các kết quả trên để nhận định đây là **dãy tăng hay giảm**? ……giảm………  sinh viên trả lời.

Xét hàm . / = OP>N tương ứng với dãy )\* + = . )\*

Bước 1: Tìm các điểm bất động với phương trình . / = /:

5 = / ⟺ 6/ − /Q = 5 ⟺ xQ − 6x + 5 = 0 ⟺ / = 1 ∨ / = 5 6 − /

Bước 2: Nhận định vị trí liên quan đến điểm bất động và vùng biến thiên của hàm . là: 1, 5

Bước 3: Xét hàm . trong khoảng 1,5 , ta có:

.

T / = 6 −5/ Q > 0

Nên dễ dàng thấy rằng . là hàm tăng.

Bước 4: Từ kết quả trên, chúng ta có thể kết luận dãy {)\*} là dãy đơn điệu và nằm trong khoảng 1,5 . Do đó, ta có thể kết luận giới hạn của dãy là: 567→8 =

**Thực hành 5: Xét giới hạn của dãy** = U V **biết dãy Fibonacci được định nghĩa như sau:** U

* + += 0= ,\* ++= 0\*P+ (

F

Với bài này, trước tiên, ta phải thiết lập được “hàm” của dãy, nghĩa là biểu diễn được dạng

)\* + = . )\* . Ta có:

)\* = \* + = \* + \*P+ = 1 + \*P+ = 1 + )\*P+1

* + \* \*

Và lưu ý rằng: )+ = WWXY = WYWWY Z = 1

Do vậy, ta xây dựng được dãy như sau:

1

I)\* + = 1 + )\* , = 1,2,3, …(

)+ = 1

Và hàm được xét tương ứng là [ / = 1 + 

Chương trình đệ quy để tính các giá trị:

>>> **def** **tisoFibo**(n):

if n==1:

**return** 1

else:

**return** 1.0 + 1.0/**tisoFibo**(n-1)

>>> for i in range(1, 11): **print** (i, tisoFibo(i))

………………………… Text

Description automatically generated………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

…………………………………………………………..

Nhận xét gì về **10** số hạng đầu tiên? Dãy đơn điệu hay không đơn điệu?........đơn điệu.......................

Tuy nhiên, chúng ta xét các số hạng lẻ (1,3,5,7,9) và các số hạng chẵn (2,4,6,8,10):

* Dãy số )+, )\, )N, )], )^ có đơn điệu tăng không?...........có....................
* Dãy số )Q, )\_, )O, )‘, )+F có đơn điệu tăng không?.............không...................

Từ đó, ta xét 2 dãy a\* là những )Eẻ và c\* là những )deẵ\*, cụ thể như sau:

+ Với dãy a\*:

a+ = )+

aQ = )\ = [ )Q = [°[ a+ a\ = )N = [ )\_ = [°[ aQ a\_ = )] = [ )O = [°[ a\

Tóm lại:

a\* + a=+ = )[°[+ a\* (

+ Với dãy c\*:

c+ = )Q

cQ = )\_ = [ )\ = [°[ c+ c\ = )O = [ )N = [°[ cQ c\_ = )‘ = [ )] = [°[ c\

Tóm lại:

c\* + = [°[ c\* ( c+ = )Q

Như vậy, 2 dãy {a\*} và {c\*} đều cùng hàm đệ quy là [°[ và chỉ khác nhau về giá trị đầu tiên.

Thật ra, 2 dãy cũng là phản ánh dãy {)\*}. Từ đó, chúng ta xét hàm:

1 2/ + 1

. / = [ ° [ / = 1 + 1 + 1/ = / + 1

Dễ thấy, khi đó 2 dãy {a\*} và {c\*} được thể hiện như sau:

2a\* + 1

Ia\* + = . a\* = a\* + 1 ( a+ = )+ = 1



Ic\* + = . c\* = c\* + 1 ( c+ = )Q = 2

Đến đây, chúng ta thực hiện các bước theo quy trình:

* Bước 1: Tìm điểm bất động của hàm . / = Q> +

> +

2/ + 1 = / ⟺ 2/ + 1 = /Q + / ⟺ /Q − / − 1 = 0 ⟺ / = 1 ± √5

/ + 1 2

* Bước 2: Xét trên trục số, ta thấy: < c+

−1<

+

P

√

N

Q

<

a

+

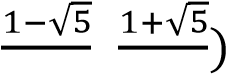
<

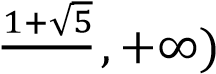
+

√

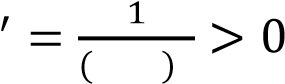
N

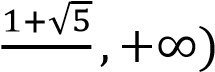
Q

Từ đó, ta xét sự hội tụ của dãy {a\*} trong khoảng Q , Q  và dãy {c\*} trong khoảng



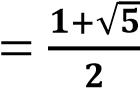
Q

* Bước 3: Khảo sát hàm số ., tính . + > X  thì ta thấy rằng hàm số . đều tăng ở 2

khoảng +P√N , + √N và , tóm lại là tăng chỉ trừ điểm / = −1 không xác định.

Q Q Q

* Bước 4: Nhận xét 2 dãy: ta thấy dãy {a\*} tăng và dãy {c\*} giảm. Từ điều đó, ta có thể kết

luận giới hạn của 2 dãy {a\*} và {c\*} là + Q√N. Tóm lại: 567→8 

Yêu cầu sinh viên: Vẽ đồ thị của hàm số . / trong khoảng +P√N , 10

Q

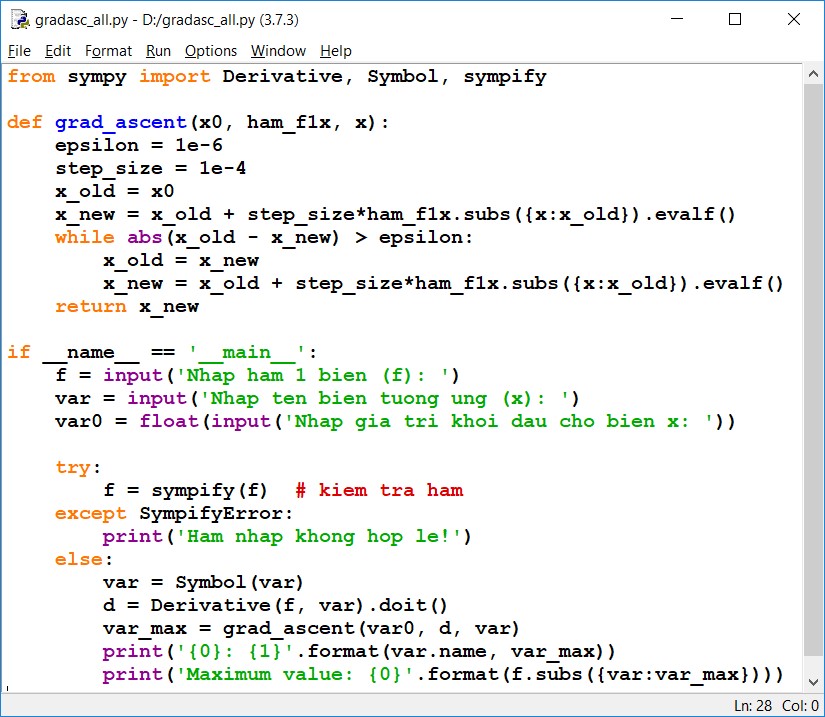
>>> ……………………………………………………….  sinh viên viết lệnh vẽ đồ thị

# Revisit: Phương pháp Gradient Ascent/Descent

Trong chương 2, chúng ta đã có khái niệm về phương pháp Gradient Ascent. Theo đó, chúng ta biết được phương pháp để tính toán ra các giá trị cực đại cho hàm số. Trong bài này, chúng ta sẽ nghiên cứu rõ hơn phương pháp bằng chương trình tổng quát và giải thích rõ hơn các tham số trong phương pháp:

## Chương trình tổng quát về Gradient Ascent

Chúng ta sẽ sửa đổi chương trình một ít để chương trình thành chương trình tổng quát về tính toán Gradient Ascent.



from sympy import Derivative, Symbol, sympify **def grad\_ascent(x0, ham\_f1x, x):**

epsilon = 1e-6 step\_size = 1e-4 x\_old = x0 x\_new = x\_old + step\_size\*ham\_f1x.subs({x:x\_old}).evalf() while abs(x\_old - x\_new) > epsilon:

x\_old = x\_new x\_new = x\_old + step\_size\*ham\_f1x.subs({x:x\_old}).evalf() return x\_new

**if \_\_name\_\_ == '\_\_main\_\_':**

f = input('Nhap ham 1 bien (f): ') var = input('Nhap ten bien tuong ung (x): ') var0 = float(input('Nhap gia tri khoi dau cho bien x: '))

try:

f = sympify(f) # kiem tra ham except SympifyError:

print('Ham nhap khong hop le!')

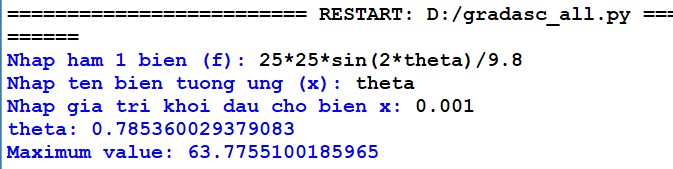
else:

var = Symbol(var) d = Derivative(f, var).doit() var\_max = grad\_ascent(var0, d, var) print('{0}: {1}'.format(var.name, var\_max)) print('Maximum value: {0}'.format(f.subs({var:var\_max})))

Lưu tập tin và thực hiện thử nghiệm:

* Nhập hàm: 25\*25\*sin(2\*theta)/9.8
* Nhập biến có tên là: theta - Giá trị khởi đầu là: 0.001

Khi đó, chương trình sẽ tính toán và chạy ra giá trị theta cũng như giá trị cực đại.



Các thử nghiệm khác:

* Thử nghiệm 1:

Nhap ham 1 bien (f): cos(t)

Nhap ten bien tuong ung (x): t

Nhap gia tri khoi dau cho bien x: 0.01

t: ……………… 0.00999900001666658………………………  Sinh viên điền giá trị vào

Maximum value: ………… 0.999950010415832……………..  Sinh viên điền giá trị vào

* Thử nghiệm 2:

Nhap ham 1 bien (f): cos(t)+p

Nhap ten bien tuong ung (x): t

Nhap gia tri khoi dau cho bien x: 0.01

t: …………… 0.00999900001666658 ………………………  Sinh viên điền giá trị vào

Maximum value: ………… p + 0.999950010415832 ……………..  Sinh viên điền giá trị vào

Tuy nhiên, lưu ý rằng hàm **cos(ky)** sẽ không hoạt động đạo hàm bậc nhất của nó là vẫn còn chứa giá trị k trong hàm mà Sympy không biết rõ giá trị k trong đó. Do đó, Sympy không thể tính toán được thuật toán như đoạn code bên trên nếu khi lấy đạo hàm còn hàm bên trong nó. Cụ thể hơn, so sánh abs(x\_old – x\_new) > epsilon không thể thực hiện.

## Các lưu ý về giá trị khởi tạo ban đầu (initial value)

Giá trị khởi tạo của biến để chúng ta bắt đầu lặp thực hiện chương trình đóng vai trò quan trọng trong thuật toán. Xét hàm /N − 30/\ + 50/. Chúng ta bắt đầu việc tìm giá trị lớn nhất sử dụng chương trình:

* Giá trị bắt đầu là -2:

Nhap ham 1 bien (f): x\*\*5-30\*x\*\*3+50\*x

Nhap ten bien tuong ung (x): x

Nhap gia tri khoi dau cho bien x: -2

t: ……………… -4.17445116397103…………………………  Sinh viên điền giá trị vào

Maximum value: ………… 705.959460322318……………..  Sinh viên điền giá trị vào

Chương trình dừng khi tìm ra vị trí đỉnh gần nhất (closest peak) nhưng không phải lúc nào cũng là giá trị cực đại toàn cục. Trong trường hợp này, chúng ta chọn giá trị khởi đầu là -2 và nó dừng lại ở đỉnh tương ứng với giá trị cực đại toàn cục (khoảng 706). Xét ví trường hợp tiếp theo với giá trị khởi đầu khác:

* Giá trị bắt đầu là 0.5:

Nhap ham 1 bien (f): x\*\*5-30\*x\*\*3+50\*x

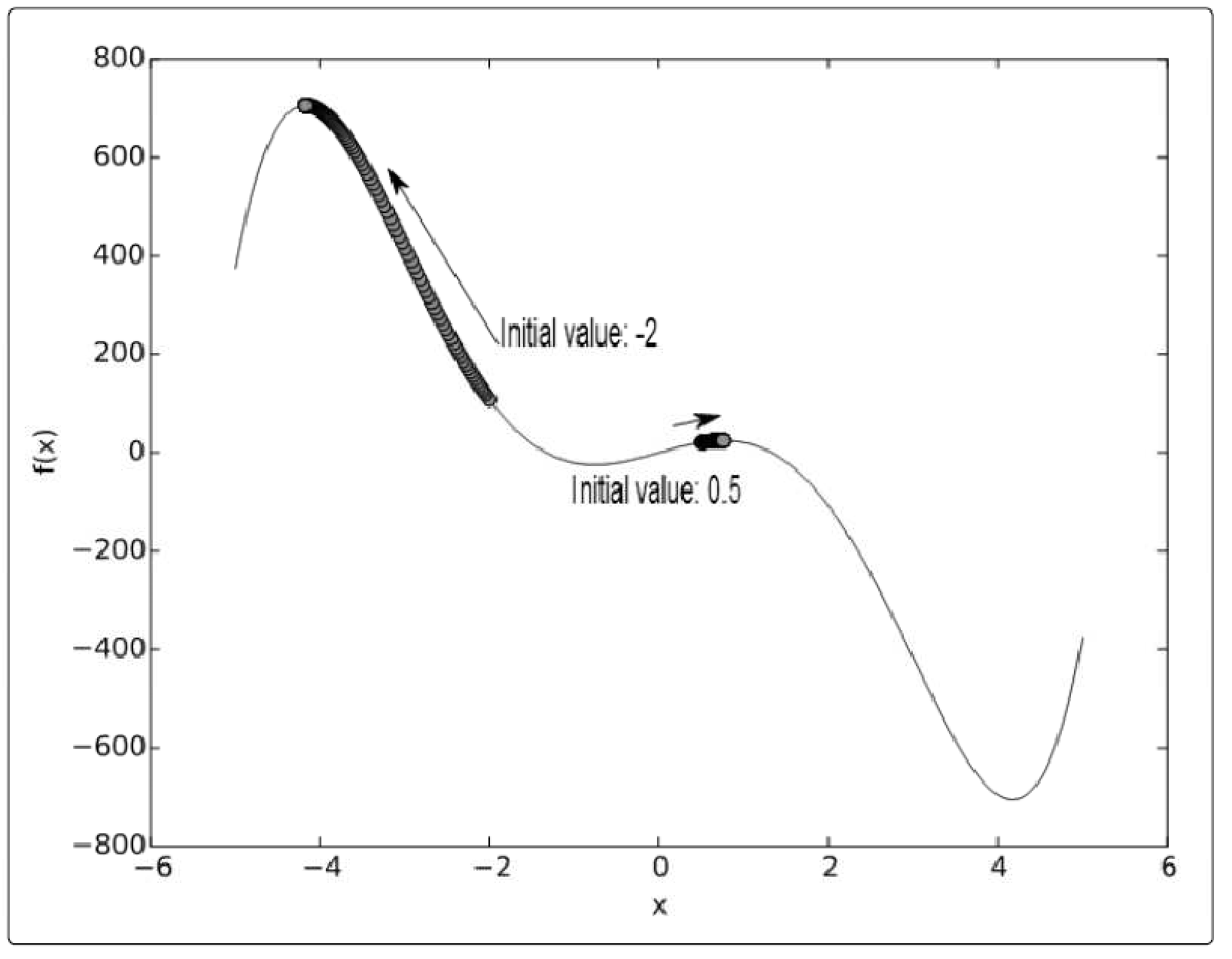
Nhap ten bien tuong ung (x): x

Nhap gia tri khoi dau cho bien x: 0.5

t: …………………… 0.757452532565767 ……………………  Sinh viên điền giá trị vào

Maximum value: …………… 25.0846622605419 ……………..  Sinh viên điền giá trị vào

Trong trường hợp này, giá trị cực đại thuật toán gradient ascent tìm được không phải là giá trị lớn nhất của hàm số. Hình dưới đây sẽ cho thấy kết quả của thuật toán gradient ascent cho cả hai trường hợp bên trên:



Do vậy, khi sử dụng phương pháp, giá trị ban đầu cần được lựa chọn kỹ lưỡng. Sau này, một số dạng thay đổi của thuật toán nỗ lực đề cập đến giới hạn này (để cải tiến).

## Vai trò của kích thước bước nhảy (step\_size) và Epsilon

Trong thuật toán **gradient ascent**, giá trị j (hay /) tiếp theo của biến được tính toán bằng phương trình:

co

jkớl = jdũ + n cj

Trong đó, n là bước nhảy (step\_size). Bước nhảy quyết định khoảng cách của bước tiếp theo. Nên nó phải đủ nhỏ để tránh chuyện vượt qua đỉnh. Do đó, nếu giá trị / gần với giá trị làm cho hàm cực đại và bước nhảy tương đối lớn thì giá trị hàm ở bước tiếp theo có thể sẽ nhỏ cực đại. Và thuật toán sẽ gọi là thất bại! Ngược lại, nếu bước nhảy quá nhỏ thì việc tính toán sẽ nhiều hơn. Trong chương trình tính toán này, chúng ta sử dụng bước nhảy là 10P\, hiển nhiên giá trị này sẽ không phù hợp với mọi hàm.

Giá trị epsilon (p) quyết định khi nào chúng ta nên dừng việc lặp trong thuật toán vì việc thay đổi của / là không đáng kể. Điều này do chúng ta mong đợi đạo hàm cấp 1 của . / sẽ triệt tiêu (bằng 0) tại điểm cực đại và lý tưởng hơn là trị tuyệt đối giữa hai giá trị / (hoặc j) bằng 0, nghĩa là |/kớl − /dũ| = 0. Tuy nhiên, do có sai số nên hiệu này sẽ không bao giờ bằng 0. Vì vậy, giá tị epsilon được chọn là giá trị gần với 0 để xử lý trường hợp này trong tính toán số thực tế, với sự ngầm hiểu là / không đổi. Trong chương trình trên, ta sử dụng p = 10PO cho tất cả các hàm. Mặc dù giá trị này đủ nhỏ và phù hợp cho hàm có nghiệm .T / = 0 như sin / nhưng đối với các hàm khác, nếu cần, chúng ta phải điều chỉnh lại giá trị epsilon.

Như vậy, chúng ta có thể cài đặt lại chương trình theo hướng việc tính toán sẽ dừng nếu đạo hàm của hàm số không có nghiệm làm triệt tiêu (.T / = 0 vô nghiệm). Ví dụ trường hợp D> hoặc log / . Với chương trình mới, nếu nhập các hàm trên thì chương trình sẽ không tiếp tục thực thi. Dưới đây là chương trình được cập nhật:

**from** sympy **import** Derivative, Symbol, sympify **def grad\_ascent(x0, ham\_f1x, x): from sympy import solve, E if not solve(ham\_f1x):**

**print('Khong the tiep tuc, phuong trinh {0}=0 vo nghiem'.format(ham\_f1x))**

**return**

# như đoạn mã cũ epsilon = 1e-6 step\_size = 1e-4 x\_old = x0 x\_new = x\_old + step\_size\*ham\_f1x.subs({x:x\_old}).evalf() while abs(x\_old - x\_new) > epsilon:

x\_old = x\_new x\_new = x\_old + step\_size\*ham\_f1x.subs({x:x\_old}).evalf()

return x\_new

**if \_\_name\_\_ == '\_\_main\_\_':**

f = input('Nhap ham 1 bien (f): ') var = input('Nhap ten bien tuong ung (x): ') var0 = float(input('Nhap gia tri khoi dau cho bien x: '))

**try:**

f = sympify(f) # kiem tra ham

**except SympifyError:**

print('Ham nhap khong hop le!')

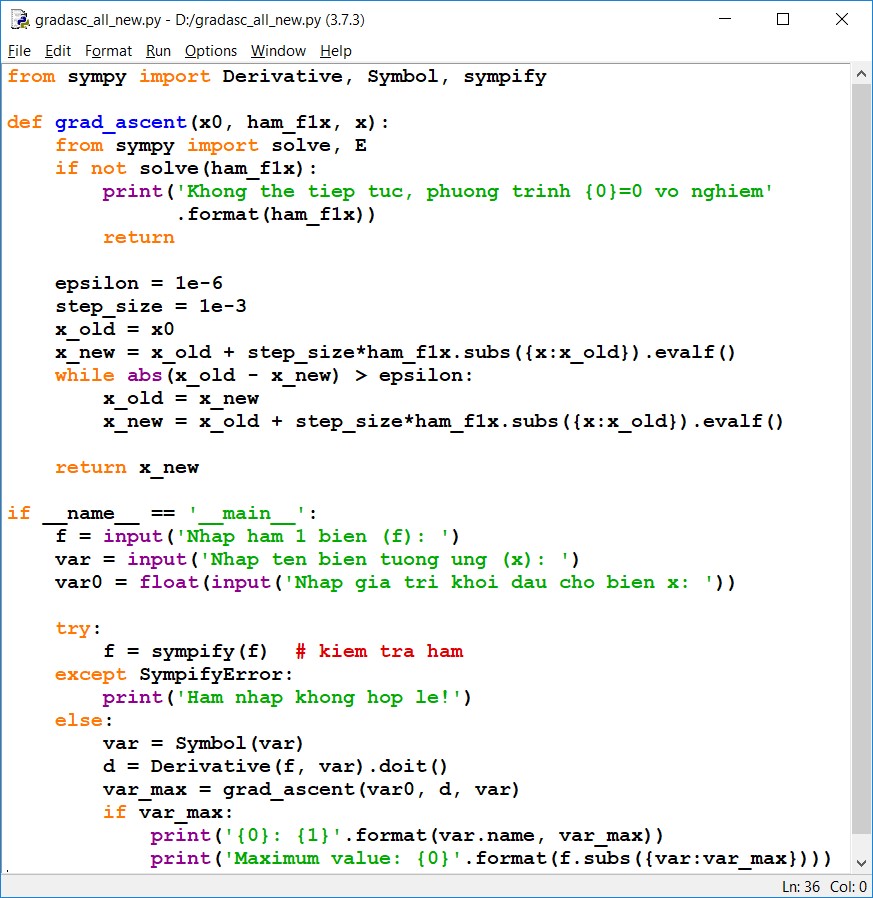
**else:**

var = Symbol(var) d = Derivative(f, var).doit() var\_max = grad\_ascent(var0, d, var)

# thêm lệnh if (nằm trong khối lệnh else) để kiểm tra giá trị var\_max trước khi kết luận:

**if var\_max:**

**print('{0}: {1}'.format(var.name, var\_max)) print('Maximum value: {0}'.format(f.subs({var:var\_max})))**



Trong phần chỉnh sửa cho hàm grad\_ascent này, chúng ta gọi hàm solve() của Sympy để kiểm tra sự tồn tại nghiệm của phương trình .’ / = 0. Nếu không có nghiệm, chúng ta in ra thông báo và trả về giá trị rỗng, xem đoạn:

**print('Khong the tiep tuc, phuong trinh {0}=0 vo nghiem'.format(ham\_f1x))**

**return**

Từ đó, chúng ta cũng thay đổi hàm \_\_main\_\_. Cụ thể là kiểm tra giá trị trả về của hàm grad\_ascent có phải là rỗng hay không; nếu có giá trị (không rỗng) thì chúng ta sẽ in ra; ngượi lại, nếu hàm không tồn tại đạo hàm thì chúng ta kết thúc chương trình.

**Thực hành: Thử nghiệm với các hàm** w0 **và** x% 0

Nhap ham 1 bien (f): **log(x)**

Nhap ten bien tuong ung (x): x

Nhap gia tri khoi dau cho bien x: 0.1

…………… Khong the tiep tuc, phuong trinh 1/x=0 vo nghiem ………  sinh viên điền vào

Nhap ham 1 bien (f): **E\*\*x**

Nhap ten bien tuong ung (x): x

Nhap gia tri khoi dau cho bien x: 0.1

……… Khong the tiep tuc, phuong trinh exp(x)=0 vo nghiem ………  sinh viên điền vào

## Lời bàn

Thuật toán cùng họ nhưng ngược lại với **gradient ascent** là thuật toán **gradient descent**. Trong thuật toán **gradient descent**, giá trị nhỏ nhất sẽ được tìm kiếm thay vì tìm giá trị lớn nhất (như gradient ascent). Do đó, điểm khác biệt lớn nhất giữa hai hàm là trong khi thuật toán gradient ascent tìm cách *“leo lên”* thì gradient descent tìm cách *“leo xuống”*. Sự khác biệt đó chính là sự hình thành giá trị tiếp theo của j (hay /) như sau:

co

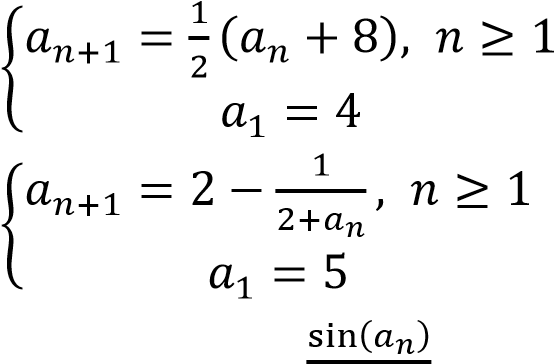
jkớl = jdũ − n cj

# BÀI TẬP CHƯƠNG 6

**Bài tập 1:** Viết chương trình tính toán 50 phần tử bằng đệ quy và tìm giới hạn các dãy sau:

1.(

2.

 3. I)\* + = )\* + ~\ = , ≥ 1 (

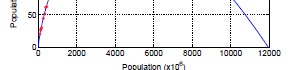
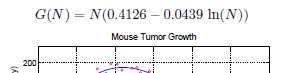
)+ = Q

**Bài tập 2:** [Sự phát triển của tế bào ung thư]

Phương trình Gompertz là một trong số những mô hình được sử dụng để mô tả các sự phát triển trong tự nhiên, sinh học. Phương trình có dạng:

= 1 − x

Với là số lượng tế bào/dân số/… (> 0) , ) và : là những hằng số được tính toán. Ví dụ hình bên dưới là đồ thị của phương trình phát triển của tế bào ung thư trên chuột:



Ở điều kiện cân bằng (equilibrium): = : − )A = 0

Do > 0 nên điều kiện cân bằng xảy ra khi: ~ = D/<

Giả định sự phát triển của tế bào ung thư theo phương trình Gompertz cụ thể như sau:

= 0.5 − 0.05A

Với là trọng lượng (mg). Hãy tìm khối lượng mg khi cân bằng (giải = 0, kết quả là mg) và tìm tỉ lệ tăng nhanh nhất của khối tế bào ung thư (giải ′ = 0, kết quả là mg/ngày).