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 8: HÀM NHIỀU BIẾN VÀ ỨNG DỤNG (tiếp theo) 3](#_Toc10698)

[0. Một số lưu ý khi sử dụng sympy (phần 2) 3](#_Toc10699)

[1. Giới thiệu Solveset trong Sympy 5](#_Toc10700)

[2. Giới thiệu các biến đổi với hàm nhiều biến 8](#_Toc10701)

[3. Bài toán tối ưu về nhân tử Lagrange 9](#_Toc10702)

[BÀI TẬP CHƯƠNG 8 13](#_Toc10703)

# CHƯƠNG 8: HÀM NHIỀU BIẾN VÀ ỨNG DỤNG (tiếp theo)

**Mục tiêu:**

* *Bổ sung về lưu ý sử dụng sympy, hàm solveset*
* *Hàm nhiều biến*
* *Bài toán tối ưu hóa hàm nhiều biến với bằng nhân tử Lagrange*

**Nội dung chính:**

# Một số lưu ý khi sử dụng sympy (phần 2)

**Dưới đây là một số lưu ý khi sử dụng sympy mà người sử dụng cần biết:**

* Để tạo danh sách (list) giá trị, gán giá trị vào trong các dấu [ ] brackets (ví dụ: x = [1,2,3,4,5]). Để lấy giá trị thứ i trong list x thì chúng ta sử dụng **x[i - 1]**. Lưu ý rằng điều này có nghĩa là truy cập đến phần tử đầu tiên (thứ **1**) là **x[0]**,…
* Các dữ liệu dạng **tuples** giống với kiểu lists nhưng khó sử dụng hơn. Về cách tạo nó giống, chỉ cần đưa các phần tử vào trong dấu ngoặc đơn, ví dụ: x = (2,3,4). Tuy nhiên, về cách sử dụng, kiểu tuple khác với list ở hai điểm. Điểm khác biệt chính là chúng ta không thể cập nhật/thay đổi được giá trị tại sau khi tạo chúng. Điều này hữu ích khi chúng ta muốn tạo một bộ dữ liệu mà ngăn sự thay đổi (cố ý hoặc vô tình nào đó, ví dụ: khi ghi nhận thông tin tại một thời điểm trong hộp đen hành trình).

**Thực hành 1a: Minh họa về sử dụng Tuple**

**>>> y = (3, 5, 7)**

**>>> y**

**(3, 5, 7)**

**>>> y[1]**

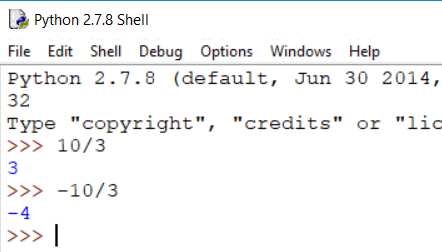
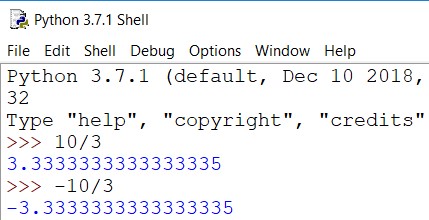
**5**

**>>> y[1]= 4**

……………………Text

Description automatically generated………………………..  sinh viên ghi nhận lỗi

* Nếu chúng ta muốn chia các số nguyên, ý tưởng tốt nhất là nên sử dụng gói from \_\_future\_\_ import division. Vì sử dụng gói này sẽ ngăn cản Python cắt bớt trị trả về bằng cách sử dụng lệnh floor(). [tuy nhiên, ở các phiên bản 3.x, dường như điều này không cần thiết]



**Thực hành 1b:**

>>> from \_\_future\_\_ import **division**

>>> 10/3

……………………… 3.3333333333333335 …………………………..  sinh viên ghi nhận giá trị

* Khi có phương trình và một giá trị (một biến), chúng ta có thể sử dụng phương thức thay thế subs để thay thế giá trị. Ví dụ, nếu ta có: **x + 14** và chúng ta biết **x = 1**, chúng ta có thể thực hiện lệnh thay thế như sau **(x + 14).subs(x, 1)** để nhận được giá trị 15.

**Thực hành 1c:**

>>> import sympy

>>> from sympy import Symbol

>>> x = Symbol('x')

>>> y = x + 1

>>> y.subs(x, 14)

……………………………15……………………………..  sinh viên ghi nhận giá trị

* Ngoài ra, chúng ta cũng có thể sử dụng phương thức thay thế biến cho biểu thức. Ví dụ: khi chúng ta muốn thay thế biến x thành biến y trong biểu thức pi - 17\*x, khi đó, chúng ta có thể thực hiện câu lệnh như sau: **(pi - 17\*x).subs(x, y)** để tạo ra pi - 17\*y. Nếu y là một hàm theo x thì biểu thức sẽ tính toán giá trị (và cuối cùng vẫn thể hiện bằng x). Tuy nhiên, nếu thay thế x bằng một biến mới (được khái báo) thì biểu thức sẽ biểu diễn theo biến mới (như ví dụ y bên trên). Minh họa:

**Thực hành 1d:**

>>> import sympy

>>> from sympy import Symbol

>>> x = Symbol('x')

>>> from sympy import pi

>>> z = pi - 18\*x

>>> z.subs(x, y)

…………………………… -18\*x - 18 + pi…………………………..  sinh viên ghi nhận giá trị

>>> t = Symbol('t')

>>> z = pi - 18\*x

>>> z.subs(x, t)

………………………… pi - 18\*t………………………………..  sinh viên ghi nhận giá trị

# Giới thiệu Solveset trong Sympy

Để giải phương trình, mặc dù Sympy hỗ trợ lệnh Solve rất hay và ấn tượng. Tuy nhiên, lệnh Solve cũng có một số giới hạn như sau:

1. Lệnh Solve không nhất quán về output của nhiều loại phương trình toán học. Xét các ví dụ thực hành như sau:

**Thực hành 2:**

>>> from sympy import solve, sin

>>> x = Symbol('x')

>>> y = Symbol('y')

>>> solve(x-1,x) # phuong trinh = 1

…………………………[1] ………………………………  sinh viên điền vào

>>> solve(x\*\*2-1,x) # phuong trinh = 1

…………………………[-1, 1] ………………………………  sinh viên điền vào

>>> solve(x\*\*2+1,x) # phuong trinh − 1 = 0

…………………………[-I, I] ………………………………  sinh viên điền vào

>>> solve(sin(x),x) # phuong trinh sin = 0

……………………………[0, pi] ………………………………  sinh viên điền vào

1. Nhiều tham số đầu vào có nhiều. Nhiều tham số không cần làm cho người sử dụng tương đối khó để giải quyết.
2. Trong một số trường hợp, việc tìm kiếm giá trị max hoặc min của hàm sử dụng các điểm cực trị (critical points) thì không thể đảm bảo bằng hàm solve.

Từ đó, sympy hỗ trợ thêm lệnh Solveset để khắc phục một số vấn đề của lệnh Solve. Cụ thể như sau:

* Rõ ràng về các giao diện đầu vào và ra! Theo đó, chúng ta phải khai báo thêm ConditionSet để hỗ trợ đầu ra. Ở đầu vào, hàm Solveset chỉ có phương trình, kể cả các biến và các tham số trong domain cũng sẽ được giải.
* Solveset có thể trả về nghiệm vô tận. Ví dụ: khi giải sin = 0, nó sẽ trả về nghiệm là

2nπ|n ∈ Z ∪ 2nπ + π|n ∈ Z .

**Thực hành 3: Thử nghiệm solveset**

>>> from sympy import solveset

>>> **solveset**(sin(x),x)

…………… Union(ImageSet(Lambda(\_n, 2\*\_n\*pi + pi), Integers), ImageSet(Lambda(\_n, 2\*\_n\*pi), Integers)) …………………………………  sinh viên điền vào

…………………………………………………………………

* Khai báo rõ **domain** của nghiệm. Domain của nghiệm là miền nghiệm. Trong Sympy, có 3 miền nghiệm, mặc định là nghiệm phức (domain = S.Complexes). Ví dụ: khi chúng ta giải phương trình sau:
* **Phương trình:** + =

**Thực hành 4:**

>>> from **sympy** import **solveset**, **S**

>>> from **sympy**.**abc** import **x, y, z**

>>> **solveset**(x\*\*2 + 1, x) # mac dinh la mien so phuc: **domain = S.Complexes**

………………… {-I, I} ……………………………………  sinh viên điền vào

>>> **solveset**(x\*\*2 + 1, x, **domain = S.Reals**) # mien so thuc

…………………… EmptySet ………………………………  sinh viên điền vào.

* **Phương trình:** = **:** (miền thực là 1 nghiệm {0} nhưng miền phức là một họ nghiệm khác). Chi tiết như sau:

**Thực hành 5:**

>>> **from sympy import solveset, S**

>>> **from sympy.abc import x, y, z**

>>> from sympy import **E**

>>> solveset(e\*\*x-1, x) # mac dinh la mien so phuc: domain = S.Complexes

……………………… Text

Description automatically generated ………………………  sinh viên điền **lỗi** vào

>>> solveset(E\*\*x-1, x) # mac dinh la mien so phuc: domain = S.Complexes

……………… ImageSet(Lambda(\_n, 2\*\_n\*I\*pi), Integers) ……………  sinh viên điền vào

>>> solveset(E\*\*x-1, x, domain = **S.Reals**)

………………………… {0} ………………………………  sinh viên điền vào

* **Phương trình phức hợp. Ví dụ:** sin + − 9 = 0

Phương trình trên sẽ có nghiệm là hợp của 2 tập: −3, 3 ∪ x|x ∈ ℝ ⋀ x + sin x = 0

**Thực hành 6:**

>>> solveset((sin(x)+x)\*(x\*\*2-9), x, domain = S.Reals)

Union(ConditionSet(x, Eq(x + sin(x), 0), Reals), {-3, 3}) ……  sinh viên điền vào

**Thực hành 7: Một số thực hành khác với Solveset:**

>>> from sympy import exp, S

>>> solveset(exp(x) > 1, x, domain = S.Reals)

……………………………… Interval.open(0, oo) …………………  sinh viên điền vào

>>> solveset(E\*\*x > 1, x, domain = S.Reals)

………………………………… Interval.open(0, oo) ………………………  sinh viên điền vào

>>> solveset(E\*\*x > E, x, domain = S.Reals)

……………………………… Interval.open(1, oo) ………………………  sinh viên điền vào

Ghi chú: domain = S.Reals chúng ta có thể sử dụng phương pháp viết tắt như sau:

**>>> R = S.Reals**

>>> solveset(E\*\*x > E, x, **R**)

……………………… Interval.open(1, oo) …………………………  sinh viên điền vào

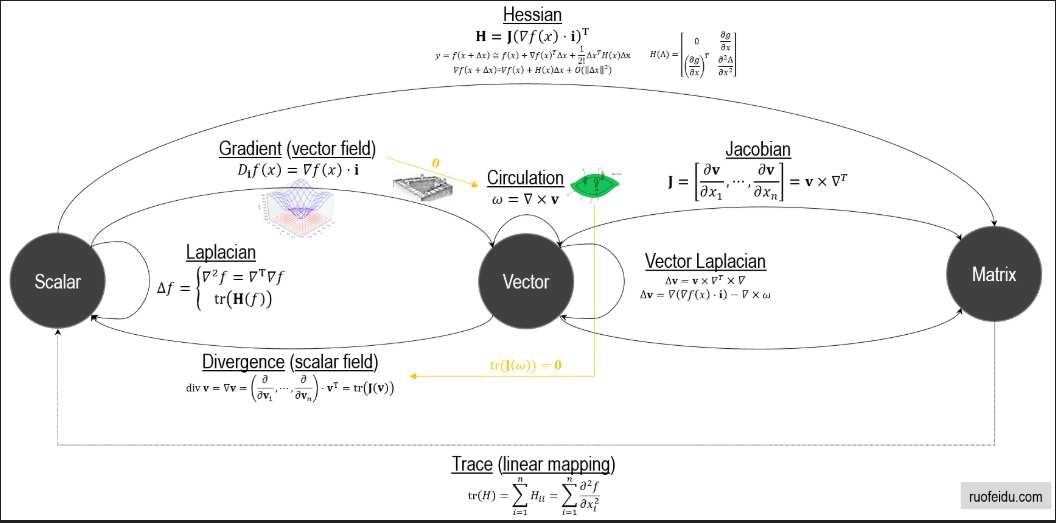
Tài liệu tham khảo thêm: https://docs.sympy.org/latest/modules/solvers/solveset.html

# Giới thiệu các biến đổi với hàm nhiều biến

Phần này giới thiệu tóm tắt để sinh viên biết tên các loại đạo hàm trong hàm nhiều biến. Mục tiêu chính là nhận diện loại giá trị đầu vào và đầu ra cho các phép tính như:

* Gradient: từ giá trị tính ra vector
* Divergence: từ vector tính ra một giá trị
* Circulation
* Vector Laplacian
* Trace
* Jacobian của một vector ra kết quả một ma trận.
* Hessian của một giá trị ra kết quả một ma trận

**BỨC TRANH TỔNG QUAN TÓM TẮT VỀ ĐẠO HÀM CỦA HÀM SỐ**



(Theo http://blog.ruofeidu.com/gradient-circulation-laplacian-divergence-jacobian-hessian-trace/) Lưu ý:

* Scalar là hàm số thông thường - Vector là hàm số theo vector
* Matrix là ma trận hàm số

# Bài toán tối ưu về nhân tử Lagrange

**3.1.Dẫn nhập**

Trong ngành tối ưu hóa, phương pháp nhân tử Lagrange (đặt theo tên của nhà toán học Joseph Louis Lagrange) là một phương pháp để tìm cực tiểu hoặc cực đại địa phương của một hàm số **theo các điều kiện giới hạn**. Bài toán được phát biểu như sau:

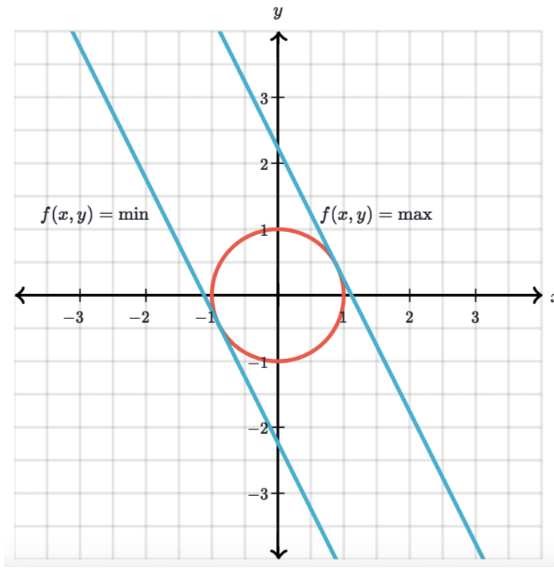
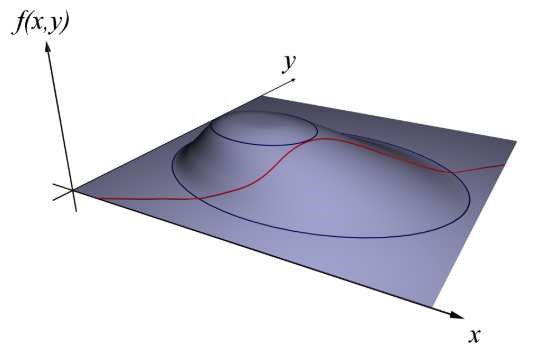
Cho hàm số ! " đa biến, với " là biến vector. Tìm " để hàm ! đạt cực trị theo điều kiện # " = $ với $ là một hằng số.

Biểu diễn toán học:

%& ’(’)\* ℎ,ặ$ %’.’(’)\* ! "

/à.# 12ộ$: # " = $

Minh họa trong dưới đây, ! , 5 = 2 + 5 là đường màu đỏ và # , 5 = + 5 (đường vòng tròn) với giá trị $ = 1 (điều kiện giới hạn được hiểu là # , 5 ≤ $) và chiếu xuống mặt phẳng, ta có các hình bên dưới (hình trên mạng nguồn: khanacademy):



**Kỹ thuật giải các bài toán nhân tử Lagrange:**

Bước 1: Thêm một biến nhân tử Lagrange 7 và định nghĩa một hàm Lagrangian ℒ như sau:

ℒ ", 7 = ! " − 7 # " − $

Với " là vector đại diện do hàm có thể nhiều biến, nên " có thể là bộ , 5, ), … Bước 2: Giải đạo hàm của ℒ ", 7 , nghĩa là ∇ℒ ", 7 = 0. Bản chất của phương trình là:

∇f X = 7∇g x ?

;

# " = $

Vì khi lấy đạo hàm thỏa điều kiện cực trị, nghĩa là phương trình ∇ℒ ", 7 = 0 sẽ là hệ:

Aℒ

A" ∇f X − 7∇g X

∇ℒ ", 7 = @AℒB = C # " − $ D

A7

Bước 3: Dựa vào các nghiệm của phương trình đạo hàm trên, ta thay thế vào hàm ! " để tìm giá trị lớn nhất hoặc nhỏ nhất

**3.2.Ông chủ bán Mì Cay**

Ví dụ: Ông chủ một cửa hàng bán mì cay mỗi tháng chỉ có thể rút từ ngân hàng 50 triệu đồng để trang trãi chi phí tiền lương thuê người làm (biến x), vật liệu sử dụng để làm mì cay (biến y) và tiền thuế/mặt bằng (giả định là một hằng số). Biết rằng lợi nhuận từ hai yếu tố mang lại là một hàm theo 2 biến và 5:

E , 5 = 4 5 − 8 − 5

Ý nghĩa là của hàm E , 5 : Nếu không có nhân viên làm việc (tương ứng = 0) thì lợi nhuận sẽ bằng 0 (vì không tạo ra được sản phẩm). Hiển nhiên, số nhân viên cũng mang lại cho ông chủ một số lợi nhuận khi làm việc. Tuy nhiên, số nhân viên càng nhiều thì lợi nhuận càng thấp (được đánh giá là −8 vì phải trả lương cho nhân viên khá lớn!). Ngược lại, nếu không sử dụng vật liệu (5 = 0) thì lợi nhuận tạo ra sẽ âm khi số nhân viên tăng lên. Việc sử dụng vật liệu càng nhiều (y càng lớn) thì càng có lợi nhuận. Ngoài ra, ông chủ còn phải chi ra 5 triệu tiền thuê mặt bằng, thuế,… để bán hàng. Với ràng buộc ông chủ chỉ vay được ngân hàng 50 triệu, hãy giúp ông chủ tìm phương án chọn x và y sao cho lợi nhuận N là lớn nhất.

Giải:

Xét hàm ! , 5, ) = 4 5 − 8 − 5 + I + J − K **.** Lúc này hàm ) được gọi là nhân tử Lagrange. Hàm ! sẽ có cực trị (cụ thể là cực đại  sinh viên tìm hiểu vì sao không phải là cực tiểu) khi:

A! A! A!

= 0,= 0,= 0

A A5 A)

Sau đó, chúng ta giải hệ trên sẽ được giá trị x, y tương ứng. Cụ thể, trong Python:

+ Lập nhân tử Lagrange:

**Thực hành 8:**

>>> from **sympy** import **Derivative, Symbol, solve**

>>> x = Symbol('x')

>>> y = Symbol('y')

>>> z = Symbol('z')

>>> f = 4\*x\*y - 8\*x - 5 + z\*(x+y-50)

+ Tính các đạo hàm riêng theo các biến x, y và z:

>>> dx = Derivative(f, x).doit()

>>> dy = Derivative(f, y).doit()

>>> dz = Derivative(f, z).doit()

+ Giải **hệ phương trình** (giải cả 3 phương trình), là hệ phương trình ∇ℒ ", 7 :

>>> nghiem = **solve**([dx, dy, dz], (x,y,z))

>>> print (nghiem)

………………… {x: 24, y: 26, z: -96} ………………………  Sinh viên điền kết quả vào

Cuối cùng, thay thế vào công thức tính, ta được số tiền thuê nhân viên (x), số tiền cần để mua nhiên liệu (y) và lợi lợi nhuận (tiền lời) dự kiến của ông chủ mì cay:

>>> xx = nghiem[x]

>>> yy = nghiem[y]

>>> 4\*xx\*yy - 8\*xx – 5

………………… 2299 …………………………  Sinh viên điền kết quả vào

Hoặc:

>>> ketqua = 4\*nghiem[x]\* nghiem[y] – 8\* nghiem[x] - 5

>>> print (ketqua)

…………………… 2299 …………………………  Sinh viên điền kết quả vào

# BÀI TẬP CHƯƠNG 8

**Bài tập 1:** Tìm các điểm trên mặt phẳng + 5 + ) = 5 (cả 3 , 5, ) biến đều dương) làm cho hàm ! , 5, ) = 5 cực đại.

[Đáp án: (x,y,z)=(1,2,2)]

**Bài tập 2:** Hai người bạn mở quán bán phở trong một khu dân cư. Hai loại phở bán ra là bát/tô thường và bát/tô đặc biệt có vốn lần lượt là 40k (nghĩa là 40 nghìn đồng) và 60k mỗi bát/tô. Một luận văn về nghiên cứu thị trường của một chị sinh viên kinh tế tìm thấy quy luật tại khu vực đó như sau: nếu gọi xk là giá bát /tô phở thường và yk là giá bát/tô phở đặc biệt giá là yk thì thị trường sẽ có thể bán được:

* Số lượng bát/tô thường được bán: 500(y-x) tô/năm.
* Số lượng bát/tô đặc biệt được bán: 45000 + 500(x-2y) tô/năm.

Hãy tìm giá bán mỗi loại tô/bát phở để lợi nhuận cao nhất năm đó.

[Đáp án: giá bát/tô thường 65k; bát/tô đặc biệt 75k].