SỰ LIÊN KẾT "HOÀN HẢO" CỦA PHƯƠNG PHÁP LẬP TRÌNH VÀ CÁC PHƯƠNG PHÁP GIẢI TOÁN CAO CẤP Ở TRƯỜNG ĐẠI HỌC VĂN LANG

THE "PERFECT" CONNECTION OF PROGRAMMING METHODS AND ADVANCED MATH SOLVING METHODS AT VAN LANG UNIVERSITY

NGUYỄN VĂN LỘC(*)

TÓM TẮT: *Bài viết trình bày sự liên kết "hoàn hảo" của phương pháp lập trình và các phương pháp giải Toán cao cấp nhằm hình thành tri thức khám phá và phương pháp giải toán đa chiều, đa biến cho sinh viên Trường Đại học Văn Lang.*
Từ khóa: *phương pháp lập trình; phương pháp giải Toán cao cấp.*

ABSTRACT: *The paper presents the "perfect" connection of programming methods and advanced math solving methods in order to form discovery knowledge and multi-dimensional and multi-variable math solving methods for students of Van Lang University.*
Key words: *programming method; advanced math solving method.*

1. ĐẶT VẤN ĐỀ

Thực tiễn sản xuất và khoa học kỹ thuật đòi hỏi, toán học hiện đại cuối thế kỷ XX, đầu thế kỷ XXI gắn chặt với điều khiển học và phát triển mạnh theo ba hướng lớn: toán học rời rạc (khắc phục sự phức tạp), toán học ngẫu nhiên (khắc phục tính bất định) và các lý thuyết tối ưu hóa (giải quyết điều khiển tốt nhất). Trong sự phát triển, toán học đã bộc lộ "khiếm khuyết", đó là, với các phương pháp theo quan điểm liên tục của Toán học cổ điển, việc tìm ra kết quả phải sử dụng lượng thời gian "không hề nhỏ", điều đó tạo nên bất lợi trong hoạt động kinh tế - kinh doanh - kỹ thuật. "Khiếm khuyết" đó sẽ được khắc phục nếu "liên kết" Toán cao cấp và lập trình. Công trình đầu tiên thể hiện sự liên kết đó là của giáo sư Wassily Leontief, Đại học Harvard. Mùa hè năm 1949, ông đã dùng máy tính Mark II để xử lý các thông tin về nền kinh tế Mỹ với 500 ngành, mỗi ngành, dùng một phương trình tuyến tính mô tả sự phụ thuộc đầu ra của ngành này đối với các ngành kinh tế khác. Công trình của W. Leontief đã mở ra một kỷ nguyên mới cho việc lập các mô hình toán học trong kinh tế [1, tr.4] và khắc phục "khiếm khuyết" của việc sử dụng các phương pháp "cổ điển" trong giải toán cao cấp. Chương trình Toán của Trường Đại học Văn Lang đã phản ánh được các hướng phát triển cơ bản của Toán học hiện đại, tuy nhiên cũng bộc lộ "khiếm khuyết" nêu trên. Do đó, để khắc phục "khiếm khuyết", việc dạy Toán cao cấp cần phải "liên kết" với thực hành lập trình, như là yêu cầu, nhằm chuẩn bị cho sinh viên kỹ năng giải các bài toán phức tạp, đa chiều, đa biến. Trong bài viết này, chúng tôi "tường minh hóa" sự liên kết "hoàn hảo" của các phương pháp giải Toán cao cấp, thông qua môn Đại số tuyến tính và phương pháp lập trình ở Trường Đại học Văn Lang.

(*) PGS.TS. Trường Đại học Văn Lang, *loc.nv@vlu.edu.vn, Mã số: TCKH25-01-2021*

2. NỘI DUNG

Python đã bắt đầu được thực hiện vào tháng 12-1989 bởi Guido van Rossum tại Centrum Wiskunde & informatica (CWI) ở Hà Lan. *"Python là một ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng và đa mục đích với rất nhiều ưu điểm: Hình thức và cấu trúc rõ ràng"* [3, tr.1]. Python đã thể hiện tính hiệu quả và ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Vì vậy, chúng tôi sử dụng ngôn ngữ lập trình Python vừa là phương pháp giải toán lập trình kết hợp với các phương pháp giải toán cao cấp của đại số tuyến tính nhằm hình thành tri thức khám phá các phương pháp khác nhau trong giải toán, một trong các loại kiến thức - kỹ năng chiến lược quan trọng phải hình thành và rèn luyện cho sinh viên thông qua hoạt động dạy học các môn học ở trường đại học, vừa là công cụ giải toán đa biến, đa chiều, giải quyết vấn đề "tốc độ" tạo ra kết quả bài toán. Sau đây, chúng tôi trình bày quy trình liên kết giữa lập trình và đại số tuyến tính thông qua giải một số dạng toán điển hình của đại số tuyến tính.

2.1. Dạng toán 1 - Tính định thức của ma trận

1) Công dụng của định thức: a) Tìm ma trận nghịch đảo; b) Tìm hạng của ma trận bằng phương pháp định thức bao quanh; c) Giải hệ phương trình tuyến tính Cramer.

2) Các phương pháp tính định thức của ma trận: Khi tính định thức của ma trận bậc 3 có thể sử dụng phương pháp Xarus hoặc phương pháp tam giác. Tuy nhiên, với các ma trận bậc 4 trở lên, các phương pháp nêu trên không sử dụng được. Khi đó, để tính định thức của ma trận bậc cao ta thường sử dụng các phương pháp sau: 1) tính định thức bằng cách sử dụng các phép biến đổi định thức; 2) tính định thức bằng cách sử dụng khai triển Laplace; 3) tính định thức bằng cách sử dụng phương pháp lập trình Python.

3) Các ví dụ:

Ví dụ 1: Tính định thức cấp 4:

$$D=\begin{vmatrix}1&3&0&0\\2&5&0&0\\-1&4&2&1\\1&-2&3&2\end{vmatrix}$$

Giải: Cách 1 (Phương pháp Laplace)

$$D=\begin{vmatrix}1&3&0&0\\2&5&0&0\\-1&4&2&1\\1&-2&3&2\end{vmatrix}=(-1)^{1+2+1+2}\begin{vmatrix}1&3\\2&5\end{vmatrix}.\begin{vmatrix}2&1\\3&2\end{vmatrix}$$
$$+(-1)^{1+2+1+3}\begin{vmatrix}1&0\\2&0\end{vmatrix}.\begin{vmatrix}4&1\\-2&2\end{vmatrix}$$
$$+(-1)^{1+2+1+4}\begin{vmatrix}1&0\\2&0\end{vmatrix}.\begin{vmatrix}4&2\\-2&3\end{vmatrix}$$
$$+(-1)^{1+2+2+3}\begin{vmatrix}3&0\\5&0\end{vmatrix}.\begin{vmatrix}-1&1\\1&2\end{vmatrix}$$
$$+(-1)^{1+2+2+4}\begin{vmatrix}3&0\\5&0\end{vmatrix}.\begin{vmatrix}-1&2\\1&3\end{vmatrix}$$
$$+(-1)^{1+2+3+4}\begin{vmatrix}0&0\\0&0\end{vmatrix}.\begin{vmatrix}-1&4\\1&-2\end{vmatrix}=-1$$

Cách 2: (Phương pháp lập trình)

Entrée[ ]:

```
import numpy as np
from numpy.linalg import det
A=np.array([[1, 3, 0, 0], [2, 5, 0, 0], [-1, 4, 2, 1], [1,-2,3,2]])
A
B= det(A)
B
```

Out [ ]:

-1.0

Nhận xét: Với các định thức lớn hơn 3, việc sử dụng hai phương pháp nêu trên để tính toán ra kết quả là không đơn giản. Tuy nhiên, nếu sử dụng phương pháp lập trình Python, sẽ thu được kết quả "tức thì" sau khi nhập liệu.

Ví dụ 2: Tính định thức của ma trận cấp 10.

$$A=\begin{bmatrix}4&3&0&0&0&0&0&0&0&0\\1&4&3&0&0&0&0&0&0&0\\0&1&4&3&0&0&0&0&0&0\\0&0&1&4&3&0&0&0&0&0\\0&0&0&1&4&3&0&0&0&0\\0&0&0&0&1&4&3&0&0&0\\0&0&0&0&0&1&4&3&0&0\\0&0&0&0&0&0&1&4&3&0\\0&0&0&0&0&0&0&1&4&3\\0&0&0&0&0&0&0&0&1&4\end{bmatrix}$$

Giải: (Phương pháp lập trình Python)

Entrée[ ]:

```
import numpy as np
from numpy.linalg import det
A=np.array([[4, 3, 0, 0, 0, 0, 0, 0, 0, 0],  [1, 4, 3, 0, 0, 0, 0, 0, 0, 0],  [0, 1, 4, 3, 0, 0, 0, 0, 0, 0],
```

[0, 0, 1, 4, 3, 0, 0, 0, 0, 0], [0, 0, 0, 1, 4, 3, 0, 0, 0, 0], [0, 0, 0, 0, 1, 4, 3, 0, 0, 0], [0, 0, 0, 0, 0, 1, 4, 3, 0, 0], [0, 0, 0, 0, 0, 0, 1, 4, 3, 0], [0, 0, 0, 0, 0, 0, 0, 1, 4, 3], [0, 0, 0, 0, 0, 0, 0, 0, 1, 4]])

B= det(A)

B

Out [ ]:

88573.00000000004

2.2. Dạng toán 2 - Tìm hạng của ma trận

1) Công dụng của hạng ma trận: a) Tìm điều kiện để một ma trận tồn tại ma trận nghịch đảo; b) Tìm điều kiện để hệ phương trình tuyến tính có nghiệm.

2) Các phương pháp tìm hạng của ma trận: 1) phương pháp ma trận bậc thang; 2) tìm hạng của ma trận bằng phương pháp định thức bao quanh; 3) sử dụng phương pháp lập trình Python trong tìm hạng ma trận.

3) Các ví dụ:

Ví dụ 1: Tìm hạng của ma trận A:

$$A=\begin{bmatrix}2 & 1 & -1 & 3\\ 0 & -1 & 0 & 0\\ 0 & 1 & 2 & 0\\ 0 & -1 & 1 & -4\end{bmatrix}$$

Giải: Cách 1: Phương pháp định thức bao quanh

Ta có định thức cấp hai, tạo thành từ dòng 1, dòng 2 và cột 1, cột 2 khác 0. Ta có:

$D_{12}^{12}=\begin{vmatrix}2 & 1\\ 0 & -1\end{vmatrix}=-2\neq 0$. Các định thức bao quanh: $D_{123}^{123}=\begin{vmatrix}2 & 1 & -1\\ 0 & -1 & 0\\ 0 & 1 & 2\end{vmatrix}=-4\neq 0$. Định thức cấp 4 là:

$$D_{123}^{123}=|A|=\begin{vmatrix}2 & 1 & -1 & 3\\ 0 & -1 & 0 & 0\\ 0 & 1 & 2 & 0\\ 0 & -1 & 1 & -4\end{vmatrix}=2\times\begin{vmatrix}-1 & 0 & 0\\ 1 & 2 & 0\\ -1 & 1 & -4\end{vmatrix}=2\times 8=16\neq 0.$$

Do đó r(A) = 4.

Cách 2: Phương pháp lập trình Python

Entrée[ ]:

import numpy as np

from numpy.linalg import matrix_rank

A=np.array([[2, 1, -1, 3], [0, -1, 0, 0], [0, 1, 2, 0], [0, -1, 1, -4]])

mrA=matrix_rank(A)

print(mrA)

Out [ ]:

4

Nhận xét: Với các ma trận chữ nhật bậc cao, việc tìm hạng của ma trận bằng phương pháp ma trận bậc thang và phương pháp định thức bao quanh hết rất nhiều thời gian. Tuy nhiên, sử dụng phương pháp lập trình sẽ cho kết quả "tức thì".

Ví dụ 2: Tìm hạng của ma trận:

$$A=\begin{bmatrix}4 & 3 & -5 & 2 & 3\\ 8 & 6 & -7 & 4 & 2\\ 4 & 3 & -8 & 2 & 7\\ 4 & 3 & 1 & 2 & -5\\ 8 & 6 & -1 & 4 & -6\end{bmatrix}$$

Giải: (Phương pháp lập trình Python)

Entrée[]:

(import numpy as np;from numpy.linalg import matrix_rank;A=np.array([[4, 3, -5, 2, 3], [8, 6 ,-7, 4, 2], [4, 3, -8, 2, 7], [4, 3, 1, 2, -5], [8, 6, -1, 4,-6]);mrA=matrix_rank(A);print(mrA))

Out [ ]:

2

2.3. Dạng toán 3 - Tìm ma trận nghịch đảo

1) Công dụng của ma trận nghịch đảo: a) Tính định thức; b). Giải phương trình ma trận; c). Giải hệ phương trình tuyến tính.

2) Các phương pháp tìm ma trận nghịch đảo: 1) phương pháp ma trận phụ hợp; 2) phương pháp Gauss-Jordan; 3) phương pháp lập trình.

3) Các ví dụ

Ví dụ 1: Tìm ma trận nghịch đảo của ma trận : $A=\begin{bmatrix}1 & 1 & 0\\ 2 & 2 & 1\\ 1 & 0 & 1\end{bmatrix}$

Giải: Cách 1: (Phương pháp ma trận phụ hợp)

$|A|=1\neq 0\Rightarrow\exists A^{-1};A_{11}=2;A_{21}=-1;A_{31}=1;$

$A_{12}=-1;A_{22}=1;A_{32}=-1;A_{13}=-2;A_{23}=-1;A_{33}=0$

$$P_A=\begin{bmatrix}2 & -1 & 1\\ -1 & 1 & -1\\ -2 & 1 & 0\end{bmatrix}$$

$$\Rightarrow A^{-1}=\frac{1}{|A|}.P_A=\begin{bmatrix}2 & -1 & 1\\ -1 & 1 & -1\\ -2 & 1 & 0\end{bmatrix}$$

Cách 2: (Phương pháp lập trình)

Entrée [ ]:

```
import numpy as np
from numpy.linalg import inv
A = np.array([[1, 1, 0], [2, 2, 1], [1, 0, 1]])
B = inv(A)
B
```

Out [ ]:

```
array([[ 2., -1.,  1.],
       [-1.,  1., -1.],
       [-2.,  1., -0.]])
```

Nhận xét: Với các ma trận bậc cao, việc tìm ma trận nghịch đảo của ma trận bằng các phương pháp truyền thống hết rất nhiều thời gian. Tuy nhiên, sử dụng phương pháp lập trình sẽ cho kết quả "tức thì".

Ví dụ 2: Tìm ma trận nghịch đảo của ma trận sau: $A=\begin{bmatrix}1 & -2 & 5 & 0\\ 2 & 0 & 4 & -1\\ 3 & 1 & 0 & 7\\ 0 & 4 & -2 & 0\end{bmatrix}$

Giải: (phương pháp lập trình)

Entrée [ ]:

(import numpy as np; from numpy.linalg import inv; A = np.array([[1, -2, 5, 0], [2, 0, 4, -1], [3, 1, 0, 7], [0, 4, -2, 0]]); B = inv(A); B)

Out[ ]:

array([[-0.72151899, 0.70886076, 0.10126582, -0.38607595], [0.21518987, -0.08860759, -0.01265823, 0.36075949], [0.43037975, -0.17721519, -0.02531646, 0.22151899], [0.27848101, -0.29113924, 0.10126582, 0.11392405]]).

2.4. Dạng toán 4 - Giải hệ phương trình tuyến tính Cramer

1) Công dụng của hệ phương trình tuyến tính: a) Mô hình hóa các bài toán kinh tế; b) Tìm vectơ riêng, giá trị riêng của ánh xạ tuyến tính.

2) Các phương pháp giải hệ phương trình tuyến tính Cramer: 1) phương pháp định thức (Phương pháp Cramer); 2) phương pháp Gauss; 3) phương pháp ma trận nghịch đảo [2, tr.65-67]; 4) phương pháp lập trình.

3) Các ví dụ:

Ví dụ 1: Giải hệ phương trình:

$$\begin{cases}2x_1 - x_2 - 2x_3 = 5\\ 4x_1 + x_2 + 2x_3 = 1\\ 8x_1 - x_2 + x_3 = 5\end{cases}$$

Giải: Cách 1. Ta có:

$$\det(A)=\begin{vmatrix}2 & -1 & -2\\ 4 & 1 & 2\\ 8 & -1 & 1\end{vmatrix}=18\neq 0$$

Vậy, hệ đã cho là hệ Cramer nên có nghiệm duy nhất là:

$$x_1=\frac{D_1}{D}=\frac{1}{18}\begin{vmatrix}5 & -1 & -2\\ 1 & 1 & 2\\ 5 & -1 & 1\end{vmatrix}=1, x_2=\frac{D_2}{D}=\frac{1}{18}\begin{vmatrix}2 & 5 & -2\\ 4 & 1 & 2\\ 8 & 5 & 1\end{vmatrix}=1, x_3=\frac{D_3}{D}=\frac{1}{18}\begin{vmatrix}2 & -1 & 5\\ 4 & 1 & 1\\ 8 & -1 & 5\end{vmatrix}=-2$$

Vậy, nghiệm của hệ là: (1, 1, -2)

Cách 2: (Phương pháp lập trình).

Entrée [ ]:

```
import numpy as np
from numpy.linalg import inv
A = np.array([[2, -1, -2], [4, 1, 2], [8, -1, 1]])
B = inv(A)
B
```

Out [ ]: array([[1.66666667e-01, 1.66666667e-01, 1.38777878e-17], [6.66666667e-01, 1.00000000e+00, -6.66666667e-01], [-6.66666667e-01, -3.33333333e-01, 3.33333333e-01]])

Entrée [ ]:

```
C = np.array([[5], [1], [5]])
C
```

Out [ ]:

(array([[5], [1], [5]])

Entrée [ ]:

```
D = B.dot(C)
D
```

Out [ ]:

(array([[1.], [1.], [-2.]]).

Nhận xét: Với các hệ phương trình bậc cao, việc giải hệ phương trình bằng các phương pháp truyền thống phải sử dụng nhiều thời gian. Tuy nhiên, sử dụng phương pháp lập trình sẽ cho kết quả "tức thì".

Ví dụ 2: Giải hệ phương trình:

$$\begin{cases} x_1 + 2x_2 + 3x_3 - 2x_4 = 6 \\ 2x_1 - x_2 - 2x_3 - 3x_4 = 8 \\ 3x_1 + 2x_2 - x_3 + 2x_4 = 4 \\ 2x_1 - 3x_2 + 2x_3 + x_4 = -8 \end{cases}$$

Giải: (Phương pháp lập trình)

Entrée [ ]: (import numpy as np; from numpy.linalg import inv; A = np.array([[1, 2, 3, -2], [2, -1, -2, -3], [3, 2, -1, 2], [2, -3, 2, 1]]); B = inv(A); B)

Out [ ]: array([[0.05555556, 0.11111111, 0.16666667, 0.11111111], [0.11111111, -0.05555556, 0.11111111, -0.16666667], [0.16666667, -0.11111111, -0.05555556, 0.11111111], [-0.11111111, -0.16666667, 0.11111111, 0.05555556]])

Entrée [ ]: (C = np.array([[6], [8], [4], [-8]]); C)

Out [ ]: (array([[6], [8], [4], [-8]])

Entrée [ ]: (D = B.dot(C) ; D)

Out [ ]: (array([[1.], [2.], [-1.], [-2.]])

3. KẾT LUẬN - KIẾN NGHỊ

Để chuẩn bị tốt năng lực cho người lao động đáp ứng yêu cầu ngày càng cao của thị trường lao động, với tầm nhìn chiến lược, Khoa Công nghệ thông tin, Trường Đại học Văn Lang đã đưa vào chương trình đào tạo các môn thực hành lập trình Python tạo nên sự liên kết "hoàn hảo" với các môn Toán cao cấp, không những giúp cho sự hình thành tri thức chiến lược, tri thức nghề và tri thức khám phá cho sinh viên mà còn phản ánh được đặc trưng của sự phát triển Toán học hiện đại trong thế kỷ XXI: Sự phát triển của máy tính điện tử và điều khiển học đã đưa Toán học phát triển lên tầm cao mới. Do vậy, có thể nói, tất cả các ngành đào tạo được trang bị kiến thức Toán cao cấp mà thiếu kiến thức thực hành lập trình, sẽ là một "khiếm khuyết" của quy trình đào tạo nghề cho sinh viên. Nên chăng, cần "chính thức hóa" việc trang bị kiến thức thực hành lập trình tương ứng với kiến thức Toán cao cấp cho sinh viên không chỉ với tư cách kiến thức nghề mà còn là một phần của "văn hóa" trong thời đại mà sự tác động ngày càng sâu rộng của công nghệ thông tin trong tất cả các lĩnh vực của cuộc Cách mạng công nghiệp 4.0.

TÀI LIỆU THAM KHẢO

[1] Bộ môn Toán - Khoa Khoa học cơ bản, Trường Đại học Văn Lang (2020), *Giáo trình Toán cao cấp*, Tài liệu lưu hành nội bộ.

[2] Lê Sĩ Đồng (2010), *Đại số tuyến tính*, Nxb Giáo dục Việt Nam.

[3] Vũ Hải Quân (2019), *Tự học lập trình Python căn bản*, Nxb Đại học Quốc gia Thành phố Hồ Chí Minh.

Ngày nhận bài: 22-7-2020. Ngày biên tập xong: 02-01-2021. Duyệt đăng: 22-01-2021