ĐẠI HỌC HOA SEN

Tạp chí Khoa học Trường Đại học Hoa Sen
https://www.hoasen.edu.vn/qlkh/ tapchidhhs.vn

LẬP TRÌNH PYTHON GIẢI TOÁN "ĐƯA DẠNG TOÀN PHƯƠNG VỀ DẠNG CHÍNH TẮC" VÀ ỨNG DỤNG

PGS.TS. Kỹ sư Tin học Nguyễn Văn Lộc*
Khoa Công Nghệ Thông Tin, Trường Đại học Hoa Sen

Thông tin bài báo

Nhận bài: 07/2025
Chấp nhận: 10/2025
Xuất bản online: 12/2025

TÓM TẮT

Mục tiêu nghiên cứu: Phát triển một công cụ hỗ trợ giải toán bằng Python để tự động hóa quá trình đưa dạng toàn phương về dạng chính tắc. Kết nối lý thuyết đại số tuyến tính với thực hành lập trình nhằm phục vụ giảng dạy, nghiên cứu và ứng dụng thực tiễn.

Phương pháp nghiên cứu: Phương pháp lập trình và thực nghiệm. Sử dụng ngôn ngữ Python với các thư viện NumPy: tính toán ma trận, giá trị riêng và vectơ riêng SymPy: xử lý biểu thức đại số và hiển thị kết quả dưới dạng ký hiệu Thử nghiệm với nhiều ví dụ trong toán học, tối ưu hóa, cơ học và machine learning. Xây dựng thuật toán giải cho mỗi dạng toán thành phần

Kết quả và kết luận: Góp phần làm rõ mối quan hệ giữa lý thuyết dạng toàn phương, ma trận đối xứng và bài toán chéo hóa trực giao. Cung cấp tài liệu tham khảo tích hợp Toán học – Lập trình cho sinh viên, giảng viên và nhà nghiên cứu.

Ý nghĩa của kết quả nghiên cứu: Xây dựng thành công chương trình Python có khả năng tự động: Nhập dạng toàn phương từ biểu thức hoặc từ ma trận đối xứng. Thực hiện phân tích giá trị riêng để tìm hệ trục chính tắc. Xuất ra kết quả dạng chính tắc cùng với ma trận chuyển đổi. Hiển thị các bước trung gian để minh họa rõ ràng quá trình biến đổi.

ABSTRACT

The goal of the research is to create a Python problem-solving application that will automatically transform quadratic forms into canonical forms. To support instruction, research, and real-world applications, integrate programming practice with linear algebra theory.

Programming and experimentation are the research methods. Using libraries and the Python language: NumPy: compute eigenvalues, eigenvectors, and matrices. SymPy is a program that processes algebraic expressions and presents the output in a symbolic format. Try out a variety of machine learning, optimisation, mechanics, and mathematics examples. Create problem-solving algorithms for every kind of challenge.

Findings and conclusions: Help elucidate the connection between orthogonal diagonalisation problems, symmetric matrices, and quadratic form theory. Give students, instructors, and researchers comprehensive reference materials on mathematics and programming.

Importance of the study findings: created a Python software that is capable of automatically Enter the quadratic form from a symmetric matrix or an equation. To determine the canonical coordinate system, use an eigenvalue analysis. Output the transformation matrix and the canonical result. To make the transformation process easy to understand, show intermediate steps.

Keywords: *Dạng toàn phương; Dạng chính tắc; Dữ liệu; Lập trình Python; Toán học.*

* Tác giả liên hệ
Email: *loc.nguyenvan@hoasen.edu.vn*

1. GIỚI THIỆU

Sử dụng lập trình Python không chỉ giải được bài toán "Đưa dạng toàn phương về dạng chính tắc", mà còn giải được các ứng dụng phong phú và đa dạng của bài toán này, cung cấp cho sinh viên các kiến thức đa chiều về ứng dụng Python trong giải toán thực tế, kỹ thuật, toán học và dữ liệu.

2. TỔNG QUAN NGHIÊN CỨU

2.1. Khái niệm

Dạng toàn phương bài toán chuẩn tắc hóa: Thông qua phép biến đổi trực giao, dạng toàn phương có thể được đưa về dạng chính tắc (diagonal form), tức chỉ còn các số hạng bình phương không chứa tích chéo.

2.2. Các nghiên cứu và ứng dụng của dạng toàn phương:

Trong Toán học và Cơ học; Trong Tối ưu hóa; Trong Khoa học dữ liệu và Machine Learning: Trong kỹ thuật và xử lý ảnh.

2.3. Xu hướng lập trình hỗ trợ giải toán. Truyền thống:

Các bước đưa dạng toàn phương về dạng chính tắc thường được tính thủ công, gây khó khăn khi số chiều tăng. Hiện đại: MATLAB và Maple có công cụ hỗ trợ chéo hóa ma trận. Python ngày càng phổ biến với thư viện mạnh mẽ (NumPy, SymPy, SciPy), cho phép vừa tính toán ký hiệu, vừa tính số trị, các nghiên cứu gần đây tập trung vào tự động hóa quy trình tính toán, tích hợp mô phỏng và trực quan hóa để phục vụ dạy học.

2.4. Khoảng trống nghiên cứu.

Các tài liệu hiện có về Python thường chỉ dừng ở minh họa rời rạc (ví dụ tính eigenvalues/eigenvectors) mà chưa hệ thống hóa thành quy trình đầy đủ để đưa dạng toàn phương về dạng chính tắc. Ít nghiên cứu kết hợp toán học thuần túy – lập trình – ứng dụng thực tiễn trong cùng một công cụ. Còn thiếu các công cụ trực quan hóa để người học dễ hiểu mối quan hệ giữa biến đổi ma trận và hình học của dạng toàn phương.

3. PHƯƠNG PHÁP NGHIÊN CỨU VÀ DỮ LIỆU

3.1. Phương pháp nghiên cứu

3.1.1. Nghiên cứu lý thuyết.

Hệ thống hóa cơ sở toán học về dạng toàn phương, ma trận đối xứng, giá trị riêng và vectơ riêng. Tìm hiểu các thuật toán đưa dạng toàn phương về dạng chính tắc: Phương pháp biến đổi cơ sở trực giao; Phân tích giá trị riêng (Eigen decomposition); Sử dụng chuẩn trực Gram–Schmidt trong một số trường hợp.

3.1.2. Phương pháp mô hình hóa và thuật toán:

Biểu diễn dạng toàn phương dưới dạng ma trận: Các bước giải thuật:

- Bước 1: Lập ma trận đối xứng từ biểu thức dạng toàn phương.
- Bước 2: Tính các giá trị riêng và vectơ riêng.
- Bước 3: Chuẩn hóa hệ vectơ riêng để xây dựng ma trận trực giao.
- Bước 4: Thực hiện biến đổi để đưa về dạng chính tắc.

3.1.3. Phương pháp lập trình và kiểm chứng. Sử dụng Python và các thư viện:

- NumPy: xử lý ma trận, tính giá trị riêng/vectơ riêng.
- SymPy: biểu diễn và rút gọn dạng toàn phương dưới dạng ký hiệu.
- Matplotlib: trực quan hóa hình học (2D, 3D).
- Xây dựng chương trình minh họa quy trình tự động đưa dạng toàn phương về dạng chính tắc.
- Kiểm chứng kết quả bằng cách so sánh giữa lý thuyết và kết quả tính toán từ Python.

3.2. Dữ liệu nghiên cứu

3.2.1. Dữ liệu toán học:

Các dạng toàn phương tiêu biểu được chọn làm ví dụ minh họa; Các dạng toàn phương trong giáo trình Đại số tuyến tính và bài tập thực hành.

3.2.2. Dữ liệu ứng dụng thực tiễn.

Tối ưu hóa: Hàm mục tiêu bậc hai trong bài toán tối ưu có ràng buộc; Cơ học – Vật lý: Ma trận độ cứng hoặc năng lượng dao động trong hệ nhiều bậc tự do; Machine Learning: Ma trận hiệp phương sai trong PCA; Kinh tế lượng: Hàm hồi quy bình phương tối thiểu (OLS) và dạng chuẩn tắc liên quan.

3.2.3. Nguồn dữ liệu:

Giáo trình Đại số tuyến tính, Tối ưu hóa bậc hai, Cơ học ứng dụng; Các bài báo, nghiên cứu khoa học về dạng toàn phương và ứng dụng; Dữ liệu mô phỏng do người nghiên cứu tự xây dựng để minh họa chương trình Python.

4. KẾT QUẢ NGHIÊN CỨU

4.1. Hàm Python đưa dạng toàn phương về dạng chính tắc

4.1.1. Thuật toán

- **Bước 1:** Biểu diễn dạng toàn phương Q(x) dưới dạng ma trận đối xứng
- **Bước 2:** Tìm trị riêng (eigenvalues) và vectơ riêng (eigenvectors)
- **Bước 3:** Đổi biến về hệ tọa độ mới
- **Bước 4:** Viết lại Q(x) dưới dạng chính tắc

4.1.2. Ví dụ.

Đưa dạng toàn phương sau về dạng chính tắc:

$$Q(x) = -x_2^2 + 4x_3^2 + 2x_1x_2 + 4x_1x_3$$ (Lê sĩ Đồng ,2010)

Giải: Cho biểu thức toàn phương:

$$Q(x) = -x_2^2 + 4x_3^2 + 2x_1x_2 + 4x_1x_3$$

Ta viết lại: $Q(x) = o.x_1^2 + (-1).x_2^2 + 4x_3^2 + 2x_1x_2 + 4x_1x_3$

- **Bước 1:** Viết ma trận đối xứng A của Q(x).

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

Với $Q(x) = x^T Ax$ ta cần xác định ma trận đối xứng A:

- **Bước 2:** Tìm trị riêng (eigenvalues) và vector riêng (eigenvector)

Mã Python minh họa bằng NumPy.

```
import numpy as np
# Ma trận đối xứng của Q(x)
A = np.array([ [0, 1, 2], [1, -1, 0], [2, 0, 4] ])
# Tính trị riêng và vector riêng
eigenvalues, eigenvectors = np.linalg.eigh(A)
# In kết quả
print("Eigenvalues (trị riêng):")
print(eigenvalues)
print("\nEigenvectors (vector riêng, cột là vector riêng):")
print(eigenvectors)
```

	Eigenvalues (trị riêng): [-1.85410197e+00 1.39807870e-16 4.85410197e+00] Eigenvectors (vector riêng, cột là vector riêng): [[0.63403768 0.66666667 0.39185683] [-0.74234424 0.66666667 0.06693714] [-0.21661313 -0.33333333 0.91758795]]

- **Bước 3:** Đổi biến về hệ tọa độ mới

Đặt: $z = P^T x \Rightarrow Q(x) = x^T Ax = z^T Dz$

Trong đó: $D = P^T AP$ là ma trận đường chéo chứa các trị riêng

- **Bước 4: Viết lại Q(x) ở dạng chính tắc**

Gọi P là ma trận trực chuẩn chứa các vector riêng và D là ma trận đường chéo chứa các trị riêng

$$Q(x) = x^T Ax = z^T Dz = -1{,}854z_1^2 + 1{,}398z_2^2 + 0{,}392z_3^2; voi..z = p^T x$$

Sau khi chạy code, ta nhận được:
- Các **trị riêng (eigenvalues)**: chính là hệ số trong dạng chính tắc.
- Các **vector riêng (eigenvectors)**: là ma trận biến đổi tuyến tính từ hệ cũ sang hệ tọa độ mới.
- Dạng chính tắc là tổ hợp của các bình phương z_i^2 theo trị riêng.

Luyện tập: Đưa dạng toàn phương sau về dạng chính tắc

$$Q(x) = 2x_1^2 + x_2^2 + x_3^2 + 3x_1x_2 + 4x_1x_3$$ (David C.Lay-Steven R.Lay-Judi J.McDonald, 2016).

Đáp số: Dạng chính tắc.

$$Q(x) = -1{,}0495z_1^2 + 1{,}0000.z_2^2 + 4{,}0495z_3^2$$

4.2. Ứng dụng thực tiễn trong khoa học kỹ thuật, kinh tế, toán học và xử lý dữ liệu của bài toán "đưa dạng toàn phương về dạng chính tắc"

4.2.1. Ứng dụng trong Cơ học và vật lý (đặc biệt trong dao động và động lực học).

Trong Cơ học và Vật lý: Dạng toàn phương là công cụ trung tâm để mô hình hóa và chuyển đổi các hệ dao động, hệ chuyển động quay, và các hệ động lực phức tạp. Dạng toàn phương (hay còn gọi là biểu thức bậc hai) xuất hiện rất tự nhiên và đóng vai trò trung tâm trong việc mô tả:
- Năng lượng trong dao động điều hòa (Cơ học);
- Năng lượng toàn phần của hệ dao động:
- Phân tích dao động riêng (Normal modes);
- Động năng và thế năng trong cơ học cổ điển và Lagrange;
- Ứng dụng trong động lực học quay;
- Dao động và điều khiển hệ thống.

Công dụng:
- **Phân tích dao động của hệ vật lý:** Nhiều bài toán dao động cơ học có thể được mô tả bởi một dạng toàn phương năng lượng.
- **Xác định trục chính của mô men quán tính** trong cơ học vật rắn.

Ví dụ: Tìm trục chính của mô men quán tính

Xét một vật phẳng quay quanh gốc tọa độ, mô men quán tính của nó được cho bởi dạng toàn phương: $I = Ax^2 + 2Bxy + Cy^2$

Yêu cầu: Đưa dạng toàn phương về dạng chính tắc: $I = \lambda_1 x'^2 + \lambda_2 x'^2$

bằng cách quay hệ trục tọa độ để loại bỏ tích xy. Từ đó: +Xác định trục chính quán tính; +Giúp thiết kế cơ cấu quay ổn định.

Giải:

Bước 1: Viết dạng toàn phương dưới dạng ma trận:

Dạng toàn phương có thể viết thành: $I = [x \quad y]\begin{bmatrix} A & B \\ B & C \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$

Ký hiệu ma trận mô men quán tính là: $M = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$

Bước 2: Tìm trị riêng (eigenvalues) và vector riêng (eigenvectors) của ma trận M. Ví dụ cụ thể:

Giả sử: A =4; B =-3; C=9. Khi đó: $M = \begin{bmatrix} 4 & -3 \\ -3 & 9 \end{bmatrix}$

Tìm trị riêng λ từ phương trình đặc trưng:

$$det(M - \lambda I) = 0 \Rightarrow \begin{vmatrix} 4-\lambda & -3 \\ -3 & 9-\lambda \end{vmatrix} = 0$$

$$\Leftrightarrow (4-\lambda)(9-\lambda) - (-3)^2 = 0 \Leftrightarrow \lambda^2 - 13\lambda + 27 = 0 \Leftrightarrow \lambda_{1,2} = \frac{13 \pm \sqrt{(-13)^2 - 4.1.27}}{2} = \frac{13 \pm \sqrt{61}}{2}$$

$$\Leftrightarrow \lambda_1 = \frac{13 - \sqrt{61}}{2}, \lambda_2 = \frac{13 + \sqrt{61}}{2}$$

Bước 3: Tìm vector riêng tương ứng

Giải hệ: $(M - \lambda I)\vec{v} = 0$

Ví dụ: Với $\lambda_1 = \frac{13-\sqrt{61}}{2}$, thay vào: $\begin{bmatrix} 4-\lambda_1 & -3 \\ -3 & 9-\lambda_1 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$

Giải ra một vector riêng $\overrightarrow{v_1}$, làm tương tự với λ_2 được $\overrightarrow{v_2}$

Nhận xét: Do M là ma trận đối xứng nên hai vector riêng trực giao, do đó có thể trực chuẩn hóa để làm cơ sở trực chuẩn mới.

Bước 4: Chuyển sang hệ trục chính (dạng chính tắc)

Sử dụng biến đổi tọa độ: $\begin{bmatrix} x \\ y \end{bmatrix} = P\begin{bmatrix} x' \\ y' \end{bmatrix}$, trong đó P là ma trận cột gồm các vector riêng chuẩn hóa.

Khi đó, phương trình trở thành: $I = \lambda_1 x'^2 + \lambda_2 y'^2$, đây là dạng chính tắc của dạng Toàn phương.

Kết luận vật lý

- Các trục x', y' là trục chính mô men quán tính;
- Giá trị λ_1, λ_2 là mô men quán tính chính tương ứng với các trục đó

Tổng kết thuật toán đưa dạng toàn phương về dạng chính tắc

Bước	Nội dung
1	Viết dạng toàn phương thành ma trận đối xứng
2	Tìm trị riêng bằng phương trình đặc trưng
3	Tìm vector riêng ứng với mỗi trị riêng
4	Chuẩn hóa và xây dựng ma trận chuyển hệ tọa độ
5	Viết lại dạng toàn phương theo tọa độ mới để có dạng chính tắc

Sử dụng hàm NumPy và SymPy để:

- Biểu diễn mô men quán tính dưới dạng toàn phương;
- Tìm trị riêng và vector riêng;
- Chuyển dạng toàn phương sang dạng chính tắc

Mã Python minh họa bằng NumPy và SymPy

Int[1]:	import numpy as np
	from sympy import Matrix, symbols, simplify, pprint
	# Bước 1: Khai báo ma trận mô men quán tính
	A, B, C = 4, -3, 9
	M = np.array([[A, B], [B, C]])
	print("🔷 Ma trận mô men quán tính:")
	print(M)
	# Bước 2: Tìm trị riêng và vector riêng (principal axes)
	eigvals, eigvecs = np.linalg.eig(M)
	print("\n🔶 Trị riêng (principal moments of inertia):")
	print(eigvals)
	print("\n🔶 Vector riêng (principal axes) – cột là eigenvectors:")
	print(eigvecs)
	# Bước 3: Kiểm tra trực chuẩn và chuyển sang dạng chính tắc
	# Gọi P là ma trận chuyển đổi gồm các vector riêng
	P = eigvecs
	D = np.dot(P.T @ M @ P)
	print("\n🔷 Dạng chính tắc (ma trận đường chéo sau đổi biến):")
	print(D)

Out[1]:	◈ Ma trận mô men quán tính: [[4 -3] [-3 9]] ◈ Trị riêng (principal moments of inertia): [2.59487516 10.40512484] ◈ Vector riêng: [[0.89..., -0.45...], [0.45..., 0.89...]] ◈ Dạng chính tắc: [[2.45... 0.0] [0.0 10.54...]]

Ý nghĩa

- **np.array → tạo ma trận ban đầu.**
- **np.linalg.eig → tìm trị riêng và vectơ riêng (moment chính và trục chính).**
- **P.T @ M @ P → biến đổi ma trận về** dạng chính tắc **(đường chéo hóa).**

4.2.2. Ứng dụng tối ưu hóa bậc hai (Quadratic Optimization)

Trong tối ưu hóa bậc hai, dạng toàn phương đóng vai trò trung tâm, đặc biệt trong các bài toán tối ưu hóa không ràng buộc và có ràng buộc tuyến tính. Dưới đây là tóm tắt chi tiết về vai trò và ứng dụng của dạng toàn phương trong tối ưu hóa bậc hai:

- Khái niệm dạng toàn phương trong tối ưu hóa;
- Bài toán tối ưu hóa bậc hai tổng quát;
- Vai trò của dạng toàn phương:
- Xác định tính lồi/lõm. Đơn giản hóa bài toán, giúp đơn giản hóa tính toán, ứng dụng trong học máy, giải bài toán tối ưu hóa hàm mục tiêu dạng bậc hai trong kinh tế, tài chính, logistic, xác định điểm cực trị, xác định đặc tính lõm/lồi của hàm;
- Ứng dụng thực tiễn: Machine Learning; Kinh tế lượng; Kỹ thuật; Trí tuệ nhân tạo; Game và Đồ họa.

Công dụng

- Giải bài toán tối ưu hóa hàm mục tiêu dạng bậc hai trong kinh tế, tài chính, logistic.
- Xác định điểm cực trị, xác định đặc tính lõm/lồi của hàm.

Ví dụ: Tối ưu hàm mục tiêu bậc hai không ràng buộc:

$$f(x,y) = 3x^2 + 4xy + 6y^2$$

Ta cần:

- Chuyển hàm này về dạng chính tắc không còn tích chéo xy;
- Phân tích ma trận Hessian (ma trận đối xứng);
- Xác định loại điểm cực trị (tối tiểu, tối đa, yên ngựa)

Giải:

- **Bước 1:** Viết dưới dạng toàn phương ma trận

  Ta viết lại hàm dưới dạng: $f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

  (Ma trận đối xứng nên hệ số 4xy chia đều vào 2xy + 2yx)

  Ma trận hệ số (Hessian): $Q = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$

- **Bước 2:** Tìm trị riêng và vector riêng của ma trận

Giải phương trình đặc trưng:

$$det(Q - \lambda I) = 0 \Rightarrow \begin{vmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{vmatrix} = 0 \Leftrightarrow (3-\lambda)(6-\lambda) - 4 = 0$$
$$\Leftrightarrow \lambda^2 - 9\lambda + 14 = 0 \Rightarrow \lambda_1 = 7, \lambda_2 = 2$$

Tìm vector riêng ứng với từng trị riêng:

$$\lambda_1 = 7 \Rightarrow (Q - 7I)\vec{v} = 0 \Rightarrow \overrightarrow{v_1} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$
$$\lambda_2 = 2 \Rightarrow (Q - 2I)\vec{v} = 0 \Rightarrow \overrightarrow{v_2} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

- **Bước 3:** Chuyển đổi `tọa độ về trục chính

  Giả sử đặt biến mới (x',y') theo hệ tọa độ mới dựa trên các vector riêng đã chuẩn hóa, khi đó:

  $$\begin{bmatrix} x \\ y \end{bmatrix} = P \begin{bmatrix} x' \\ y' \end{bmatrix} .. voi .. P = \left[\frac{\overrightarrow{v_1}}{\|\overrightarrow{v_1}\|}, \frac{\overrightarrow{v_2}}{\|\overrightarrow{v_2}\|} \right]$$

  Thay vào, ta có dạng chính tắc:

  $$f(x,y) = \lambda_1 (x')^2 + \lambda_2 (y')^2 = 7(x')^2 + 2(y')^2$$

- **Bước 4:** Phân tích bài toán tối ưu hóa
- Ma trận Hessian có các trị riêng đều dương nên ma trận xác định dương
- Hàm f(x,y) là hàm lồi nghiêm ngặt
- Điểm cực tiểu toàn cục tại nghiệm của $\nabla f = 0$

Tổng kết thuật toán đưa dạng toàn phương về dạng chính tắc

Bước	Nội dung
1	Viết hàm mục tiêu dạng ma trận toàn phương
2	Tìm trị riêng và vector riêng của Hessian
3	Chuyển đổi hệ trục (tọa độ) về trục chính
4	Viết lại hàm dưới dạng chính tắc
5	Phân tích cực trị dựa vào dấu trị riêng

Mã Python minh họa bằng NumPyv

Int[1]:	import numpy as np
	from numpy.linalg import eig
	# Bước 1: Ma trận Hessian
	Q = np.array([[3, 2],
	[2, 6]])
	# Bước 2: Trị riêng và vector riêng
	eigvals, eigvecs = eig(Q)
	print("🔷 Trị riêng (eigenvalues):", eigvals)
	print("🔷 Vector riêng (eigenvectors – theo cột):\n", eigvecs)
	# Bước 3: Dạng chính tắc
	D = eigvecs.T @ Q @ eigvecs
	print("🔶 Dạng chính tắc (Q' = P^T Q P):\n", D)

Out[1]:	🔷 Trị riêng (eigenvalues): [2. 7.] 🔷 Vector riêng (eigenvectors – theo cột): [[-0.89442719 -0.4472136] [0.4472136 -0.89442719]] 🔶 Dạng chính tắc (Q' = P^T Q P): [[2. 0.] [0. 7.]]

Ý nghĩa

- np.array →định nghĩa ma trận Hessian/dạng toàn phương.
- eig(Q) →tìm trị riêng và vectơ riêng.
- eigvecs.T@ Q @ eigvecs →biến đổi ma trận về **dạng chính tắc**.

Dạng chính tắc: $f(x,y) = \lambda_1 (x')^2 + \lambda_2 (y')^2 = 7(x')^2 + 2(y')^2$

4.2.3. Ứng dụng trong Machine Learning và trí tuệ nhân tạo

Trong Machine Learning và Trí tuệ nhân tạo, dạng toàn phương thường xuất hiện trong:

- Hàm mất mát bậc hai (Quadratic Loss Function);
- Hàm mục tiêu trong Ridge Regression, Linear Discriminant Analysis (LDA);
- Phân tích phương sai (PCA), tối ưu hóa convex/quadratic.

Công dụng:

Trong Principal Component Analysis (PCA): Biến đổi ma trận hiệp phương sai (dạng toàn phương) về dạng chéo để giảm chiều dữ liệu.

Tăng hiệu quả huấn luyện mô hình ML (John Paul Mueller, 2016)

Ví dụ: Demo Python: PCA và dạng chính tắc

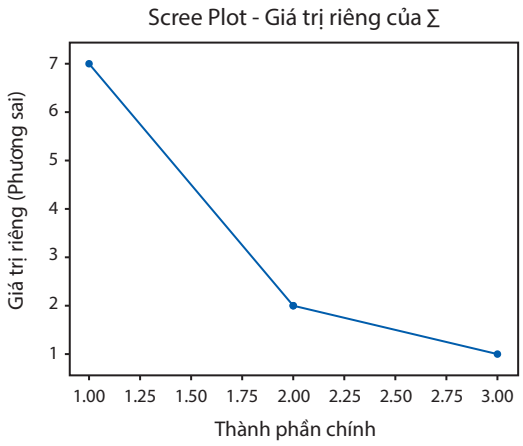
Int[1]:	import numpy as np import pandas as pd import matplotlib.pyplot as plt from numpy.linalg import eigh # 1. Sinh dữ liệu giả (3 chiều, có tương quan) np.random.seed(0) mu = np.array([0, 0, 0]) Sigma = np.array([[4.0, 2.4, 1.2], [2.4, 3.0, 1.5], [1.2, 1.5, 2.5]]) X = np.random.multivariate_normal(mu, Sigma, size=200) df = pd.DataFrame(X, columns=["X1", "X2", "X3"]) # 2. Tính ma trận hiệp phương sai X_centered = df - df.mean() cov_matrix = np.cov(X_centered.T) print("Ma trận hiệp phương sai Σ:") print(cov_matrix) # 3. Phân tích giá trị riêng (Eigen decomposition) eigvals, eigvecs = eigh(cov_matrix) idx = np.argsort(eigvals)[::-1] # Sắp xếp giảm dần eigvals, eigvecs = eigvals[idx], eigvecs[:, idx] print("\nGiá trị riêng (Λ):", eigvals) print("\nVectơ riêng (P):\n", eigvecs) # 4. Đổi biến: Z = X_centered * P Z = X_centered @ eigvecs print("\n5 dòng đầu của tọa độ thành phần chính (Z):\n", Z.head()) # 5. Kiểm chứng dạng toàn phương: x^T Σ x = z^T Λ z x = X_centered.iloc[0].values.reshape(-1,1) lhs = float(x.T @ cov_matrix @ x) z = eigvecs.T @ x rhs = float(z.T @ np.diag(eigvals) @ z) print("\nKiểm chứng dạng toàn phương:") print("x^T Σ x =", lhs, "≈ z^T Λ z =", rhs) # 6. Vẽ Scree Plot plt.plot(range(1, len(eigvals)+1), eigvals, marker='o') plt.title("Scree Plot - Giá trị riêng của Σ") plt.xlabel("Thành phần chính") plt.ylabel("Giá trị riêng (Phương sai)") plt.show()

Out[1]:

Ma trận hiệp phương sai Σ:
[[4.11349979 2.65846641 1.10647026]
[2.65846641 3.26214033 1.51293652]
[1.10647026 1.51293652 2.43108787]]
Giá trị riêng (Λ): [7.09751065 1.8974878 0.81172954]
Vectơ riêng (P):
[[-0.69103189 -0.52578015 0.49601428]
[-0.62334159 0.0860563 -0.77719983]
[-0.36595109 0.8462562 0.38720828]]
5 dòng đầu của tọa độ thành phần chính (Z):

	0	1	2
0	4.525721	-0.687882	0.934532
1	5.645740	-2.762265	-0.881438
2	2.395294	0.099918	-0.022169
3	1.030277	-0.224455	1.487882
4	1.910341	-0.243995	0.513999

Kiểm chứng dạng toàn phương:
$x^T \Sigma x = 146.97909277060606 \approx z^T \Lambda z = 146.97909277060$



Giải thích kết quả

Đoạn code là một ví dụ PCA (Phân tích thành phần chính) minh họa trực tiếp việc đưa dạng toàn phương về dạng chính tắc.

Bước 1: Sinh dữ liệu giả.

- Dữ liệu được sinh ngẫu nhiên từ phân phối chuẩn đa biến 3 chiều;
- Ma trận hiệp phương sai gốc Σ được chọn có nhiều phần tử ngoài đường chéo $\neq 0 \Rightarrow$ các biến có tương quan;
- Bộ dữ liệu này chính là tình huống PCA thường được dùng để xử lý: giảm tương quan và đưa về hệ tọa độ mới.

Bước 2: Tính ma trận hiệp phương sai mẫu

Bước 3: Phân tích giá trị riêng. Mỗi giá trị riêng chính là phương sai của dữ liệu trên một trục chính (principal component); Vectơ riêng xác định hướng của các trục đó; Do $\lambda 1$ lớn nhất, PC1 giữ phần lớn thông tin (phương sai lớn nhất của dữ liệu).

Bước 4: Đổi biến. Đây là phép biến đổi Z=XP; Dữ liệu gốc được chiếu lên hệ trục mới P; Bảng kết quả (5 dòng đầu) là tọa độ trong hệ PC; Các tọa độ này không còn tương quan (covariance giữa các PC ≈ 0).

Bước 5: Kiểm chứng dạng toàn phương. Đây chính là đưa dạng toàn phương về dạng chính tắc.

Bước 6: Scree Plot giúp quyết định giữ lại bao nhiêu thành phần chính khi giảm chiều dữ liệu.

Tóm lại: Code đã minh họa toàn bộ quy trình PCA = chéo hóa ma trận hiệp phương sai.

Kết quả chứng tỏ PCA chính là việc đưa dạng toàn phương về dạng chính tắc: dữ liệu ban đầu có tương quan → biến đổi trực giao → hệ trục mới độc lập, chỉ còn phương sai trên đường chéo Λ

4.2.4. Ứng dụng trong Thị giác máy tính và xử lý ảnh

Trong thị giác máy tính (computer vision) và xử lý ảnh (image processing), dạng toàn phương thường xuất hiện trong:

- Phát hiện cạnh, góc (Corner detection – ví dụ Harris Corner);
- PCA trong trích chọn đặc trưng;
- Elliptical blob detection (Laplacian of Gaussian);
- Moments hình học: chuyển hệ tọa độ để nhận dạng, khử quay/scaling.

Công dụng:
- Trong phân tích ảnh, nhận diện hình dạng: dùng ma trận **moment quán tính** (dạng toàn phương) để xác định **trục đối xứng chính** của hình dạng.
- Trong trích xuất đặc trưng ảnh như **ellipse fitting, corner detection.**

Ví dụ: Tìm trục chính của một blob ảnh (covariance ellipse)

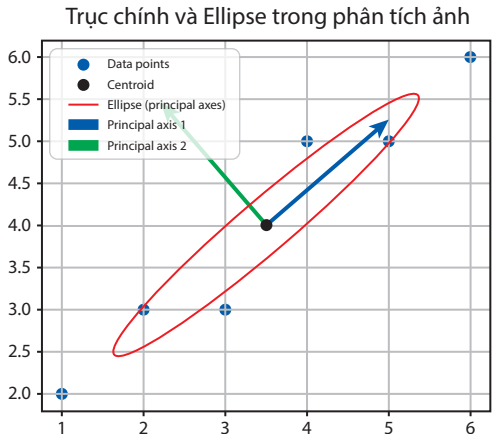
Int[1]:	import numpy as np
	# Giả sử có 6 điểm ảnh sáng thuộc 1 blob
	X = np.array([[1, 2],[2, 3],[3, 3],[4, 5],[5, 5],[6, 6]])

Int[2]:	centroid = np.mean(X, axis=0) X_centered = X - centroid Int[3]: cov_matrix = np.cov(X_centered.T) print("◈ Ma trận hiệp phương sai:\n", cov_matrix)
Out[3]:	◈Ma trận hiệp phương sai: [[3.5 2.8] [2.8 2.4]]
Int[4]:	eigvals, eigvecs = np.linalg.eig(cov_matrix) print("◈ Trị riêng (Eigenvalues – phương sai theo trục chính):\n", eigvals) print("◈ Vector riêng (Eigenvectors – trục chính):\n", eigvecs)
Out[4]:	◈ Trị riêng (Eigenvalues – phương sai theo trục chính): [5.80350661 0.09649339] ◈ Vector riêng (Eigenvectors – trục chính): [[0.77225168 -0.63531673] [0.63531673 0.77225168]]

Int[5]:	import matplotlib.pyplot as plt
	def draw_ellipse(mean, eigvals, eigvecs):
	t = np.linspace(0, 2*np.pi, 100)
	circle = np.array([np.cos(t), np.sin(t)]) # đơn vị tròn
	ellipse = eigvecs @ np.diag(np.sqrt(eigvals)) @ circle
	plt.plot(ellipse[0] + mean[0], ellipse[1] + mean[1], color='red', label='Ellipse (principal axes)')
	plt.figure(figsize=(6,6))

```
plt.scatter(centroid[0], centroid[1], color='black', label='Centroid')
draw ellipse(centroid, eigvals, eigvecs)
# Vẽ trục chính
for i in range(2):
vec = eigvecs[:, i]
plt.quiver(centroid[0], centroid[1], vec[0], vec[1],
angles='xy', scale_units='xy', scale=0.5, color=['blue','green'][i], label=f'Principal axis {i+1}')
plt.gca().set_aspect('equal')
plt.legend()
plt.title("Trục chính và Ellipse trong phân tích hình ảnh")
plt.grid(True)
plt.show()
```

Out[5]:



Giải thích kết quả

- Ma trận hiệp phương sai Σ: biểu diễn tương quan giữa các biến gốc.
- Giá trị riêng Λ: chính là phương sai theo từng trục chính (PC1, PC2, …).
- Vectơ riêng P: xác định hướng của các trục chính (biến đổi trực giao).
- Đẳng thức dạng toàn phương: $x^T \sum x = z^T \Lambda z$ được kiểm chứng bằng tính toán trong code.
- Scree plot: giúp chọn số thành phần chính cần giữ lại.

Như vậy, PCA thực chất là một ứng dụng đưa dạng toàn phương về dạng chính tắc bằng lập trình Python.

Ta có các ứng dụng khác như sau:

4.2.5. Ứng dụng trong đại số tuyến tính và hình học giải tích

Trong đại số tuyến tính và hình học giải tích, đưa một dạng toàn phương về dạng chính tắc là bước nền tảng để:

- Phân loại đường conic, mặt bậc hai;
- Đơn giản hóa phương trình bằng phép quay tọa độ; Phân tích hình học không gian: elip, hypebol, parabol (Stephen Boyd, 2018)

Công dụng:

- Phân loại các loại mặt bậc hai trong không gian: mặt cầu, mặt ellipsoid, mặt hyperbole, mặt parabol...
- Đưa phương trình tổng quát về dạng chính tắc giúp xác định loại mặt và vẽ hình (Howard Anton, 2010)

4.2.6. Ứng dụng trong kinh tế lượng (Econometrics)

Trong kinh tế lượng, dạng toàn phương xuất hiện tự nhiên trong:

- Hàm mất mát của hồi quy tuyến tính;
- Ước lượng bình phương tối thiểu (OLS);
- Phân tích phương sai-covariance;
- Kiểm định Wald, kiểm định tương quan tuyến tính

Công dụng:

- Trong phân tích phương sai – hiệp phương sai, việc chéo hóa ma trận (dạng toàn phương) giúp xác định các thành phần độc lập ảnh hưởng đến biến phụ thuộc (Kevin Sheppard, 2019)
- Phân tích nhân tố.

4.2.7. Ứng dụng trong kỹ thuật và thiết kế

Trong kỹ thuật và thiết kế, dạng toàn phương thường xuất hiện trong:

- Phân tích ứng suất - biến dạng;
- Tính mô men quán tính của tiết diện;
- Thiết kế ổn định kết cấu;
- Tối ưu hóa hình dạng

Công dụng:

- Trong cơ học kết cấu và thiết kế máy: xác định phương chính của vật liệu trong ứng suất – biến dạng.
- Thiết kế các bộ cảm biến chịu lực, biến dạng theo các hướng ưu tiên. (Michael Baron, 2014)

5. KẾT LUẬN

Kết quả nghiên cứu đề tài chứng minh tính khả thi trong cả giảng dạy, nghiên cứu và ứng dụng thực tiễn. Đề tài đã kết hợp thành công lý thuyết đại số tuyến tính và công cụ lập trình Python để tự động hóa bài toán đưa dạng toàn phương về dạng chính tắc. Giúp đơn giản hóa bài toán, hiểu rõ cấu trúc và bản chất của hệ thống. Xây dựng chương trình Python có khả năng tự động nhận dạng và đưa dạng toàn phương về dạng chính tắc. Hiển thị các bước trung gian (ma trận, giá trị riêng, vectơ riêng, ma trận biến đổi, dạng chính tắc). Hỗ trợ giảng dạy đại số tuyến tính, tối ưu hóa, cơ học và trí tuệ nhân tạo. Gợi mở khả năng mở rộng thành thư viện Python nhỏ hoặc tích hợp vào các hệ thống học tập trực tuyến.

Công cụ lập trình có ưu điểm: Tính toán nhanh và chính xác. Hiển thị đầy đủ bước giải, giúp người học dễ hiểu bản chất toán học, có khả năng mở rộng thành thư viện hỗ trợ học tập và nghiên cứu.

Hạn chế: Chưa tối ưu cho các dạng toàn phương có kích thước rất lớn (ma trận nhiều chiều).

Hướng phát triển: Mở rộng ứng dụng sang các lĩnh vực khoa học dữ liệu, trí tuệ nhân tạo. Xây dựng giao diện trực quan (web/app) để người dùng dễ thao tác. Kết quả nghiên cứu còn cung cấp cho sinh viên hệ thống kiến thức sử dụng lập trình Python trong xử lý các bài toán không chỉ trong toán học, dữ liệu mà trong các lĩnh vực khoa học kỹ thuật, kinh tế lượng và trong giải các bài toán thực tế, phát triển ở sinh viên tư duy sáng tạo và tư duy thuật toán, những loại hình tư duy rất cần cho người lao động trong cách mạng công nghiệp 4.0.

TÀI LIỆU THAM KHẢO

[1] Anton, H. (2010). *Elementary linear algebra: Applications version*. Wiley.

[2] Baron, M. (2014). *Probability and statistics for computer scientists*. CRC Press.

[3] Boyd, S. (2018). *Introduction to applied linear algebra*. Cambridge University Press.

[4] Đồng, L. S. (2010). *Toán cao cấp: Đại số tuyến tính*. NXB Giáo dục Việt Nam.

[5] Lay, D. C., Lay, S. R., & McDonald, J. J. (2016). Linear algebra and its applications. Pearson.

[6] Mueller, J. P. (2016). *Machine learning for dummies*. Dummies – A Wiley Brand.

[7] Sheppard, K. (2019). *Introduction to Python for econometrics, statistics and data analysis*. University of Oxford.