Advances in Applied Mathematics 应用数学进展, 2024, 13(2), 501-510
Published Online February 2024 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2024.132049

Hans汉斯

Python软件在线性代数课程中的应用探索

齐秀文，徐娇娇，宋　帆

昌吉学院，数学与数据科学学院，新疆　昌吉

收稿日期：2024年1月5日；录用日期：2024年1月31日；发布日期：2024年2月7日

摘　要

针对线性代数课程内容较抽象、计算复杂的特点，引入一款合适的软件尤为重要，而Python简单易学，功能强大，正好是教学辅助工具的最佳选择。在教学过程中引入Python，将抽象枯燥的代数理论与有趣的实践练习联系在一起，实现将抽象问题具体化、复杂问题简单化的目标。重点探讨了Python软件在解线性方程组、二次型和正交化三个方面的应用。将软件与传统教学模式相结合，可增强学习的趣味性，提高学生学习兴趣和教学质量。

关键词

Python，线性方程组，正交化，二次型

The Application of Python Software in the Course of Linear Algebra

Xiuwen Qi, Jiaojiao Xu, Fan Song

School of Mathematics and Data Science of Changji University, Changji Xinjiang

Received: Jan. 5th, 2024; accepted: Jan. 31st, 2024; published: Feb. 7th, 2024

Abstract

In view of the abstract content and complex calculation of linear algebra, it is particularly important to introduce a suitable software. Python is easy to learn and powerful, which is the best choice for teaching aids. Python is introduced in the teaching process to link abstract and boring algebra theory with interesting practical exercises, so as to realize the goal of concretizing abstract problems and simplifying complex problems. The application of Python software in solving linear equations, quadratic form and orthogonalization is discussed. The combination of software and traditional teaching mode can enhance the interest of learning and improve students' learning interest and teaching quality.

文章引用: 齐秀文, 徐娇娇, 宋帆. Python 软件在线性代数课程中的应用探索[J]. 应用数学进展, 2024, 13(2): 501-510.
DOI: 10.12677/aam.2024.132049

Keywords

Python, Linear Equations, Orthogonalization, Quadratic Form

1. 引言

Python 语言诞生于 1990 年，由 Guido van Rossum 设计并领导开发。最近十年 Python 在网络爬虫、数据分析、AI、机器学习、Web 开发、金融、运维及测试等领域都有不俗的表现，它专注于解决问题、自由开发的社区环境以及丰富的第三方库，Python 具有强大的科学及工程计算能力，它不但具有以矩阵计算为基础的强大数学计算能力和分析功能，而且还具有丰富的可视化表现功能和简洁的程序设计能力[1]。

线性代数是大学理、工、经管各专业重要的基础必修课，它在培养具有良好科学素养和创新能力的数学及应用人才方面起着十分重要的作用。它是以讨论有限维空间线性理论为主，具有较强的抽象性与逻辑性，是数学的一个重要分支，其理论与方法已广泛应用于其它科学领域中。主要包括：矩阵、行列式、线性方程组、秩问题、矩阵的特征值和特征向量、二次型等内容。同时，该课程能够为培养工科各专业学生的逻辑推理和抽象思维能力、空间直观和想象能力打下良好的基础。通过对线性代数课程的学习，学生掌握行列式、矩阵、线性方程组、向量组等基本理论，进一步增强学生的数学素养、数学计算、抽象思维与逻辑思维能力，提高学生综合分析、处理问题的能力，为利用矩阵这个数学工具处理专业领域内的复杂工程问题提供理论基础。然而，线性代数中存在着大量以运算为基础的内容，这些内容往往成为许多学生学习线性代数的拦路虎，影响了他们对线性代数的兴趣，制约了他们对线性代数知识的进一步理解及应用，Python 软件强大的计算功能也会使线性代数中庞大的计算变得轻而易举，在线性代数的教学过程中，在适当时候把 Python 软件引入到教学中，引导学生学会使用 Python 解决计算问题，让学生能够把重心放在对理论知识的理解上，能有效提升教学效果，也能提高学生的实践应用能力，从而大大地提高了学生的学习积极性和教学效果[2]。

2. Python 软件在线性代数中的应用举例

2.1. Python 在解线性方程组中的应用

2.1.1. 消元法

对如下所示的线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{1}$$

利用线性方程组的初等变换，得到与原方程组(1)同解的方程组。用矩阵的初等变换，消掉方程组中各方程中的一些未知量，得到方程组解的过程，称为消元法[3]。

令方程组(1)的系数矩阵 $\boldsymbol{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$，未知量矩阵 $\boldsymbol{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$，常数矩阵 $\boldsymbol{b}=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$，则线性方程组(1)可表示为 $\boldsymbol{Ax}=\boldsymbol{b}$，则线性方程组的消元解法可通过对增广矩阵

$$(\boldsymbol{A}\,|\,\boldsymbol{b})=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix}$$

作有限次矩阵的三种行初等变换，有时也需作矩阵的第一种列初等变换，将其化为行最简阶梯型矩阵，即可得到原方程组的解。

2.1.2. 消元法的 Python 程序

依据消元法的一般步骤，分两步给出消元法的 Python 程序[4]。

第 1 步：消元

首先，导入 numpy 库，定义矩阵的三种初等变换，分别记为 Q1，Q2，Q3；然后定义 rowStep 函数，用 while 循环进行消元，直到系数矩阵成阶梯型，while 循环结束，代码如附件 1.1 所示。

第 2 步：回代

回代过程与消元几乎一致，只是从下往上进行，得到最简行阶梯矩阵，代码如附件 1.2 所示。

例 1 用消元法解线性方程组 $\begin{cases} 2x_1+2x_2-x_3=6 \\ x_1-2x_2+4x_3=3 \\ 5x_1+7x_2+x_3=28 \end{cases}$。

解：将上述方程组的系数矩阵输入 A，b，运行程序，即可得到如下结果：

1) 消元后结果如表 1 所示：

Table 1. Elimination results
表 1. 消元结果

消元后的阶梯型矩阵	非零行数	是否进行列变换
[[1, 1, −0.5, 3] [0, 1, −1.5, 0] [0, 0, 1, 2]]	3	否，序列为[0, 1, 2]

2) 回代后结果如表 2 所示：

Table 2. Substitution results
表 2. 回代结果

回代后矩阵	对应方程组
[[1, 0, 0, 1] [0, 1, 0, 3] [0, 0, 1, 2]]	$\begin{cases} x_1=1 \\ x_2=3 \\ x_3=2 \end{cases}$

由表 1，表 2 可知，原方程组的解为 $\begin{cases} x_1 = 1 \\ x_2 = 3 \\ x_3 = 2 \end{cases}$。

2.2. Python 在实二次型相关问题中的应用

2.2.1. 利用正交变换化实二次型为标准型

用正交变换化实二次型 $f\left(x_1, x_2, \cdots, x_n\right) = x^{\mathrm{T}} Ax$ 为标准型，主要有三个步骤，如下[5]：

1) 写出二次型 f 的矩阵 $A = \left(a_{ij}\right)_{n\times n}$，将 A 化为实对称矩阵；

2) 求 n 阶正交矩阵 Q，使得

$$Q^{-1}AQ = Q^{\mathrm{T}}AQ = diag\left(\lambda_1, \lambda_2, \cdots, \lambda_n\right);$$

3) 正交变换化二次型为 $f = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2$。

上述步骤所对应的 Python 程序[4]如附件 2.1 所示。

例 2 求一正交变换，化二次型

$$f\left(x_1, x_2, x_3\right) = x_1^2 + 4x_2^2 + 4x_3^2 - 4x_1x_2 + 4x_1x_3 - 8x_2x_3$$

为标准型。

解：上述二次型可化为

$f = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} 1 & -4 & 4 \\ 0 & 4 & -8 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$，将 $A = \begin{pmatrix} 1 & -4 & 4 \\ 0 & 4 & -8 \\ 0 & 0 & 4 \end{pmatrix}$ 代入程序，可得如表 3 所示的结果。

Table 3. Orthogonalization results
表 3. 正交化结果

特征值	正交矩阵	对角化矩阵
[0, 0, 9]	[[−0.0778, 0.9396, 0.3333] [−0.7241, 0.1766, −0.6667] [−0.6853, −0.2932, 0.6667]]	[[0, 0, 0] [0, 0, 0] [0, 0, 9]]

由表 3 可知，对角化后为 $f = 9y_3^2$。

2.2.2. 实二次型正定性的判定

n 阶对称矩阵 A 是正定的充分必要条件是 A 的特征值都大于零。

依据上述定理，给出判定实二次型正定的 Python 程序[4]如附件 2.2 所示。

例 3 判断 $f\left(x_1, x_2, x_3\right) = 2x_1^2 + 5x_2^2 + 5x_3^2 + 4x_1x_2 - 4x_1x_3 - 8x_2x_3$ 是否为正定二次型。

解：上述二次型可化为

$f = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} 2 & 4 & -4 \\ 0 & 5 & -8 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$，将 $A = \begin{pmatrix} 2 & 4 & -4 \\ 0 & 5 & -8 \\ 0 & 0 & 5 \end{pmatrix}$ 代入上述程序，可得如表 4 所示结果。

由表 4 结果可知，上述二次型为正定二次型。

Table 4. Eigenvalues of the quadratic form
表 4. 二次型的特征值

特征值	是否正定
[1, 1, 10]	正定

2.3. Python 在线性无关向量组正交化中的应用

欧氏空间 V 中的任一组线性无关的向量 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 都可以通过施密特正交化过程化为两两正交的向量 $\beta_1,\beta_2,\cdots,\beta_m$，且向量组 $\alpha_1,\alpha_2,\cdots,\alpha_m$ 与 $\beta_1,\beta_2,\cdots,\beta_m$ 等价，施密特正交化过程如下[5]：

$$\beta_1=\alpha_1$$

$$\beta_2=\alpha_2-\frac{\langle\alpha_2,\beta_1\rangle}{\langle\beta_1,\beta_1\rangle}\beta_1$$

$$\vdots$$

$$\beta_m=\alpha_m-\frac{\langle\alpha_m,\beta_1\rangle}{\langle\beta_1,\beta_1\rangle}\beta_1-\frac{\langle\alpha_m,\beta_2\rangle}{\langle\beta_2,\beta_2\rangle}\beta_2-\cdots-\frac{\langle\alpha_m,\beta_{m-1}\rangle}{\langle\beta_{m-1},\beta_{m-1}\rangle}\beta_{m-1}$$

由施密特正交化过程，给出正交化的 Python 代码：

例 4 设线性无关的向量组 $\alpha_1=(1,1,1,1)^{\mathrm{T}}$，$\alpha_2=(3,3,-1,-1)^{\mathrm{T}}$，$\alpha_3=(-2,0,6,8)^{\mathrm{T}}$，将 $\alpha_1,\alpha_2,\alpha_3$ 正交化。

解：将 $\alpha_1,\alpha_2,\alpha_3$ 代入上述程序中，可得如表 5 所示结果。

Table 5. Orthogonalization matrix
表 5. 正交化矩阵

正交化前矩阵	正交化后矩阵
[[1, 3, −2] [1, 3, 0] [1, 1, −6] [1, −1, 8]]	[[1, 2, −1] [1, 2, 1] [1, −2, −1] [1, −2, 1]]

由表 5 可知，$\alpha_1,\alpha_2,\alpha_3$ 正交化后 $\beta_1=(1,1,1,1)^{\mathrm{T}}$，$\beta_2=(2,2,-2,-2)^{\mathrm{T}}$，$\beta_3=(-1,1,-1,1)^{\mathrm{T}}$。

3. 结论

文章结合线性代数中的具体实例，探讨 Python 软件在解决线性代数问题中的应用，许多复杂的代数问题都可以通过编程实现，可以帮助实现做题过程的动态化展示，从而更深刻地理解解题思路和方法，通过写代码的过程，也可以让学生再次去理解和应用线性代数抽象的理论知识[6]。计算在线性代数的知识体系中占比很重，贯穿于线性代数学习的始终，还有很多问题可采用类似的方法去应用，使得 Python 和线性代数形成一个完整的互相促进的课程体系，可大大提高学生学习的积极性，增强数学的趣味性，实现复杂内容的简单化、抽象问题的具体化。

参考文献

[1] 张健, 张良均. Python 编程基础[M]. 北京: 人民邮电出版社, 2019.

[2] 刘剑. 将 MATLAB 融入高等代数教学的探究[J]. 桂林师范高等专科学校学报, 2019, 33(1): 113-115.

[3] 赵树嫄. 线性代数[M]. 第六版. 北京: 中国人民大学出版社, 2021.

[4] 戊崂石. 线性代数 Python 计算导引[EB/OL]. https://blog.csdn.net/u012958850/article/details/125171081

[5] 徐仲, 陆全, 张凯院, 等. 高等代数考研教案[M]. 西安: 西北工业大学出版社, 2011.

[6] 刘秀英, 孔祥强. Maple 软件在高等代数课程中的应用探索[J]. 赤峰学院学报(自然科学版), 2018, 34(3): 13-15.

附录

1. 消元法的 Python 程序

1.1. 消元

```
import numpy as np                                  #调用 numpy
def Q1(M, i, j, row=True):                          #定义第一种初等变换
    if row:
M[[i,j]]=M[[j,i]]
    else:
M[:,[i,j]]=M[:,[j,i]]
def Q2(M,i,l, row=True):                            #定义第二种初等变换
    if row:
        M[i]=l*M[i]
    else:
        M[:,i]=l*M[:,i]
def Q3(m,i,j,l,row=True):                           #定义第三种初等变换
    if row:
        M[j]+=l*M[i]
  else:
        M[:,j]+=l*M[:,i]
def rowStep(M, m, n):                                   #定义 rowStep 函数
    rank=0
    zero=m
    i=0
    order=np.array(range(n))
    while i<min(m,n) and i<zero:
        flag=False
        index=np.where(abs(M[i:,i])>1e-10)
        if len(index[0])>0:
            rank+=1
            flag=True
            k=index[0][0]
            if k>0:
                Q1(A,i,i+k)
        else:
            index=np.where(abs(M[i,i:n])>1e-10)
            if len(index[0])>0:
                rank+=1
                flag=True
```

```
                k=index[0][0]
                Q1(M,i,i+k,row=False)
                order[[i, k+i]]=order[[k+i, i]]
        if flag:
            Q2(M,i,1/M[i,i])
            for t in range(i+1, zero):
                Q3(M,i,t,-M[t,i])
            i+=1
        else:
            Q1(M,i,zero-1)
            zero-=1
    return rank, order
M=np.array([[   ,   ,   ],                          #要求解方程组的系数矩阵,以 3 阶为例
            [   ,   ,   ],[   ,   ,   ]],dtype='float')
b=np.array([   ,   ,   ])                           #要求解方程组的常数项，以 3 阶为例
B=np.hstack((M,b.reshape(3,1)))
rank, order=rowStep(B,3,3)
print(B)
print(rank)
print(order)
```

1.2. 回代

```
def simplestStep(M,rank):
    for i in range(rank-1,0,-1):
        for j in range(i-1, -1,-1):
            Q3(M,i,j,-M[j,i])
np.set_printoptions(suppress=True)
M=np.array([[   ,   ,   ],                          #要求解方程组的系数矩阵,以 3 阶为例
            [   ,   ,   ],[   ,   ,   ]],dtype='float')
b=np.array([   ,   ,   ])                           #要求解方程组的常数项，以 3 阶为例
B=np.hstack((M,b.reshape(3,1)))
rank, order=rowStep(B,3,3)
simplestStep(B,rank)
print(B)
```

2. 实二次型的 Python 程序

2.1. 正交变换化二次型为标准型

```
import numpy as np                                  #调用 numpy
np.set_printoptions(precision=4, suppress=True)
A=np.array([[   ,   ,   ],                          #二次型对应的矩阵
```

```
            [  ,  ,  ],[  ,  ,  ]])
def symmetrization(A):                          #将矩阵 A 对称化
    n,_=A.shape
    for i in range(n):
        for j in range(i+1,n):
            A[i,j]=(A[i,j]+A[j,i])/2
            A[j,i]=A[i,j]
symmetrization(A)
t,Q=np.linalg.eigh(A)                                  #计算正交阵 Q
print(t)
print(Q)
print(np.matmul(np.matmul(Q.T,A),Q))
```

2.2. 实二次型正定性的判定

```
import numpy as np                              #调用 numpy
def symmetrization(A):                          #定义矩阵 A 对称化函数
    n,_=A.shape
    for i in range(n):
        for j in range(i+1,n):
            A[i,j]=(A[i,j]+A[j,i])/2
            A[j,i]=A[i,j]
A=np.array([[,,],                          #二次型对应的矩阵
            [  ,  ,  ], [  ,  ,  ]])
symmetrization(A)                          #对称化 A
t=np.linalg.eigvalsh(A)                     #计算 A 的特征值
print(t)
```

3. Python 在线性无关向量组正交化中的应用

正交化的 Python 代码

```
import numpy as np                              #调用 numpy
def orthogonalize(M):
    _,m=M.shape
    B=M.copy()
    for i in range(1,m):
        for j in range(i):
            B[:,i]-=(np.dot(B[:,j],M[:,i])/np.dot(B[:,j],B[:,j]))*B[:,j]
    return B                                    #正交化结果
np.set_printoptions(precision=4, suppress=True)
M=np.array([[,,],                          #向量组矩阵
```

```
                [   ,   ,   ],
                [   ,   ,   ],
                [   ,   ,   ]],dtype='float')
B=orthogonalize(M)
print(B)
```