第09讲 基底变换 课后习题答案

作者：欧新宇（Xinyu OU）

本文档所展示的测试结果，均运行于：Intel Core i7-7700K CPU 4.2GHz

• 作业要求及提交

1. 将所有运行结果保存为一个`word`文档（特别推荐保存为`pdf`文档进行提交）
2. 要求：使用编程环境完成下列习题，并按题目顺序进行排版，每个题目要求按如下顺序组织（若存在）：
   `0).`题目(将题目完整Copy到作业文档中，可以通过公式编辑器编辑或截图方式）；
   `1).`需要手工书写的部分，请尽量在word文档中进行编辑（迫不得已时，可书写在纸上并拍照）；
   `2).`代码（尽量通过从编程环境截图粘贴）；
   `3).`运行结果贴于文档中。（复制运行结果到文档或通过截图粘贴）
   `x).` **如果熟悉本编程环境'Jupyter Notebook'也可以直接在本环境中编写所有文稿及代码，并打印成pdf文档进行提交。**
3. 将文档上传至 `课堂派` 平台

> 注意：截图只需要截取必要部分。此外，请确保截图清晰可见。

• 提示：

在python中，如果需要对numpy数组中的元素进行格式化，可以使用以下两种方法

`1.` 默认保留精度为n的浮点数, 对于小数位数不足n的，将自动截断

```
np.set_printoptions(precision=3, suppress=True)  # 设置保留三位小数
```

`2.` 按照formatter的格式化要求进行设置，对于小数位数不足的元素，将进行补零

```
np.set_printoptions(formatter={'float': '{: 0.3f}'.format})
```

• 答案及解析

1. 令$v_1=(3,3)^T, v_2=(3,5)^T$ 对有序基$u_1=(2,2)^T, u_2=(-2,3)^T$，求从$[v_1,v_2]$到$[u_1,u_2]$的转移矩阵。

(1). $u_1=(1,2)^T, u_2=(1,-4)^T$

(2). $u_1=(5,4)^T, u_2=(2,5)^T$

解：

> 问题分析：从特定基$[u_1,u_2]$到特定基$[v_1,v_2]$的基底变换（坐标变换）。此处，求$[v_1,v_2]$到$[u_1,u_2]$的转移矩阵，可以直接套用公式$S=U^{-1}V$

(1). $S_1=U^{-1}V=\begin{bmatrix}1 & 1\\ 2 & -4\end{bmatrix}^{-1}\begin{bmatrix}4 & 2\\ 3 & 5\end{bmatrix}$

(2). $S_2 = U^{-1}V = \begin{bmatrix} 5 & 2 \\ 4 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 4 & 2 \\ 3 & 5 \end{bmatrix}$

后续的工作，同样交给Python来实现。

```
import numpy as np
from scipy import linalg

U1 = np.array([[1,1],[2,-4]])
U2 = np.array([[5,2],[4,5]])

V = np.array([[4,2],[3,5]])

S1 = np.dot(linalg.inv(U1), V)
S2 = np.dot(linalg.inv(U2), V)

np.set_printoptions(precision=3, suppress=True)  # 设置保留三位小数
print('S1=\n{}'.format(S1))
print('S2=\n{}'.format(S2))
```

```
S1=
[[ 3.167  2.167]
 [ 0.833 -0.167]]
S2=
[[ 0.824  0.   ]
 [-0.059  1.   ]]
```

2. 令$E = [(2,4)^T, (3,4)^T]$，并令$u = (3,5)^T, v = (6,8)^T$。计算$[u]_E, [v]_E$。

解：

> 问题分析：从标准基$[e_1, e_2]$到特定基$[u_1, u_2]$的基底变换（坐标变换）。根据坐标变换公式，基坐标$x = Uc$，我们可以得到从标准基坐标向特定基坐标的变换公式：$c = U^{-1}x$，其中，U^{-1}就是标准基$[e_1, e_2]$到特定基$[u_1, u_2]$的转移矩阵。

(1). $[u]_E = E^{-1}u = \begin{bmatrix} 2 & 4 \\ 3 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 5 \end{bmatrix}$

(2). $[v]_E = E^{-1}v = \begin{bmatrix} 2 & 4 \\ 3 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 8 \end{bmatrix}$

后续工作交给Python来实现。

```
import numpy as np
from scipy import linalg

E = np.array([[2,3],[4,4]])

u = np.array([[3],[5]])
v = np.array([[6],[8]])

np.set_printoptions(precision=3, suppress=True)  # 设置保留三位小数
print('xE=\n{}'.format(np.dot(linalg.inv(E), u)))
print('yE=\n{}'.format(np.dot(linalg.inv(E), v)))
```

```
xE=
[[0.75]
 [0.5 ]]
yE=
[[0.]
 [2.]]
```

3. 令$v_1=(1,2,3)^T, v_2=(2,3,4)^T, v_3=(3,4,5)^T$，并令$u_1=(9,7,6)^T, u_2=(8,6,3)^T, u_3=(2,4,6)^T$，求：

(1). 求从特定基$[v_1,v_2,v_3]$到特定基$[u_1,u_2,u_3]$的转移矩阵。

(2). 若$x=-3v_1-5v_2+7v_3$，确定向量x相应于u_1,u_2,u_3的坐标。

解：

> 问题分析：从特定基$[u_1,u_2,u_3]$到特定基$[v_1,v_2,v_3]$的基底变换（坐标变换）。根据坐标变换的通用公式$Vc=Ud$，即可求得坐标和转移矩阵。

(1). 要获得v到u的转移矩阵，实际上就是将v到e和e到u的转移矩阵进行合成，那么可以得到：

$$S=U^{-1}V=\begin{bmatrix}9&8&2\\7&6&4\\6&3&6\end{bmatrix}^{-1}\begin{bmatrix}1&2&3\\2&3&4\\3&4&5\end{bmatrix}$$

(2). 基于v的坐标x转换为基于u的坐标，可以通过通用公式的变形获得，即：

$$Vc=Ud=>d=U^{-1}Vc=>x_{new}=U^{-1}Vx=S\begin{bmatrix}-3\\-5\\7\end{bmatrix}$$

接下来的工作，仍然交给Python来实现。

```python
import numpy as np
from scipy import linalg

U = np.array([[9,8,2],[7,6,4],[6,3,6]])
V = np.array([[1,2,3],[2,3,4],[3,4,5]])
xv = np.array([[-3],[-5],[7]])

S = np.dot(linalg.inv(U), V)
xu = np.dot(S,xv)

np.set_printoptions(precision=3, suppress=True)  # 设置保留三位小数
print('S=\n {}'.format(S))
print('xu=\n{}'.format(xu))
```

```
S=
 [[-0.     0.048  0.095]
 [ 0.     0.048  0.095]
 [ 0.5    0.595  0.69 ]]
xu=
[[0.429]
 [0.429]
 [0.357]]
```

4. 给定$v_1 = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, v_2 = \begin{bmatrix} -3 \\ -4 \end{bmatrix}, S = \begin{bmatrix} 2 & -2 \\ 3 & 5 \end{bmatrix}$**，求：**$u_1, u_2$**，使得**$S$**为从**$[v_1, v_2]$**到**$[u_1, u_2]$**的转移矩阵。**

解：

> 问题分析：从特定基$[u_1, u_2]$到特定基$[v_1, v_2]$的基底变换（坐标变换）。由于S是v到u的转移矩阵，所以有$S = U^{-1}V$，由此可以得到$V = US => U = VS^{-1}$

$$U = VS^{-1} = \begin{bmatrix} 3 & -3 \\ 5 & -4 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 3 & 5 \end{bmatrix}^{-1}$$

后续计算同样使用Python实现。

```
import numpy as np
from scipy import linalg

S = np.array([[2,-2],[3,5]])
V = np.array([[3,-3],[5,-4]])

W = np.dot(V, linalg.inv(S))

np.set_printoptions(precision=3, suppress=True)  # 设置保留三位小数
print('W=\n {}'.format(W))
```

```
W=
 [[1.5   0.   ]
 [2.312 0.125]]
```