Matrices_Orthogonales

June 26, 2020

1 Le groupe orthogonal de $\mathbb{R}^3$

Marc Lorenzi

17 juillet 2016

1.1 0 Préliminaires

On se place dans l'espace euclidien orienté $E = \mathbb{R}^3$. Les endomorphismes orthogonaux de E sont de l'un des types suivants : 1. Les réflexions, symétries orthogonales par rapport à un plan, caractérisées par un vecteur normal au plan de la réflexion. 2. Les rotations. L'identité mise à part, elles sont caractérisées par un axe, orienté par un vecteur e, et un angle $\theta \not\equiv 0[2\pi]$. 3. Les composées d'une réflexion et d'une rotation différente de l'identité, dont l'axe est orthogonal au plan de la réflexion. L'axe et l'angle de la rotation caractérisent complètement ces endomorphismes orthogonaux.

On se propose de répondre aux deux questions suivantes : 1. Connaissant les éléments caractéristiques d'un endomorphisme orthogonal, déterminer sa matrice dans la base canonique. 2. Connaissant la matrice d'un endomorphisme orthogonal, déterminer sa nature et ses éléments caractéristiques.

[1]:
```
from sympy import *
init_printing()
```

Définissons tout d'abord les vecteurs de la base canonique de $\mathbb{R}^3$.

[2]:
```
e1 = Matrix([1, 0, 0])
e2 = Matrix([0, 1, 0])
e3 = Matrix([0, 0, 1])
e1, e2, e3
```

[2]:
$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

Et voici une petite fonction renvoyant un vecteur unitaire colinéaire à un vecteur non nul u.

[3]:
```
def normaliser(u):
    return u / u.norm()
```

[4]:
```
normaliser(Matrix([1, 2, 1]))
```

[4]:
$$\begin{bmatrix} \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \end{bmatrix}$$

1.2 1 Les réflexions

1.2.1 1.1 Image d'un vecteur par une réflexion

Soit f la réflexion de plan P, où P est donné par un vecteur normal e. On a alors pour tout $u \in \mathbb{R}^3$, $f(u) = u - 2\frac{<u,e>}{||e||^2}e$.

[5]:
```
def reflexion(e, u):
    return u - 2 * u.dot(e) / e.norm() ** 2 * e
```

[6]:
```
reflexion(Matrix([1,1,1]), Matrix([1,2,1]))
```

[6]:
$$\begin{bmatrix} -\frac{5}{3} \\ -\frac{2}{3} \\ -\frac{5}{3} \end{bmatrix}$$

1.2.2 1.2 Matrice d'une réflexion

Pour obtenir la matrice d'une réflexion, il suffit de calculer les images des vecteurs de la base canonique par celle-ci, puis de les ranger dans une matrice.

[7]:
```
def matrice_reflexion(e):
    u1 = reflexion(e, e1)
    u2 = reflexion(e, e2)
    u3 = reflexion(e, e3)
    return u1.col_insert(1, u2).col_insert(2, u3)
```

[8]:
```
A0 = matrice_reflexion(Matrix([1, 2, 3]))
A0
```

[8]:
$$\begin{bmatrix} \frac{6}{7} & -\frac{2}{7} & -\frac{3}{7} \\ -\frac{2}{7} & \frac{3}{7} & -\frac{6}{7} \\ -\frac{3}{7} & -\frac{6}{7} & -\frac{2}{7} \end{bmatrix}$$

Quelques vérifications … la matrice A_0 est symétrique, ce qui est une bonne chose. Ensuite,

[9]:
```
A0 * A0
```

[9]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

et donc A_0 est une matrice de symétrie orthogonale. Enfin,

[10]:
```
det(A0)
```

[10]:
-1

donc A_0 est une matrice de réflexion.

1.2.3 1.3 Caractéristiques d'une réflexion

Donnons-nous une matrice A. La matrice A est-elle une matrice de réflexion ? Et si oui, par rapport à quel plan ?

[11]:
```
def est_reflexion(A):
    return A != - eye(3) and A == A.T and A.T * A == eye(3) and det(A) == -1
```

[12]:
```
est_reflexion(A0)
```

[12]:
```
True
```

[13]:
```
def analyse_reflexion(A):
    return (A + eye(3)).nullspace()[0]
```

[14]:
```
analyse_reflexion(A0)
```

[14]:
$$\begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 1 \end{bmatrix}$$

1.3 2 Rotations

1.3.1 2.1 Image d'un vecteur par une rotation

Une rotation f de l'espace différente de l'identité est caractérisée par son axe, orienté par un vecteur ω et son angle $\theta \not\equiv 0[2\pi]$. En prenant ω unitaire, on a pour tout vecteur u de l'espace la formule d'Euler-Rodrigues :

$f(u) = \cos\theta u + < \omega, u > (1 - \cos\theta)\omega + \sin\theta \ \ \omega \wedge u$

La fonction **rotation** renvoie l'image du vecteur u par la rotation caractérisée par ω unitaire et l'angle θ.

[15]:
```
def rotation(omega, theta, u):
    c = cos(theta)
    s = sin(theta)
```

```
    v1 = c * u
    v2 = s * omega.cross(u)
    v3 = omega.dot(u) * (1 - c) * omega
    return v1 + v2 + v3
```

[16]:
```
x, y, z = symbols('x y z')
```

[17]:
```
rotation(normaliser(Matrix([1, 1, 1])), pi/2, Matrix([x, y, z]))
```

[17]:
$$\begin{bmatrix} -\frac{\sqrt{3}y}{3} + \frac{\sqrt{3}z}{3} + \frac{\sqrt{3}\left(\frac{\sqrt{3}x}{3} + \frac{\sqrt{3}y}{3} + \frac{\sqrt{3}z}{3}\right)}{3} \\ \frac{\sqrt{3}x}{3} - \frac{\sqrt{3}z}{3} + \frac{\sqrt{3}\left(\frac{\sqrt{3}x}{3} + \frac{\sqrt{3}y}{3} + \frac{\sqrt{3}z}{3}\right)}{3} \\ -\frac{\sqrt{3}x}{3} + \frac{\sqrt{3}y}{3} + \frac{\sqrt{3}\left(\frac{\sqrt{3}x}{3} + \frac{\sqrt{3}y}{3} + \frac{\sqrt{3}z}{3}\right)}{3} \end{bmatrix}$$

[18]:
```
simplify(_)
```

[18]:
$$\begin{bmatrix} \frac{x}{3} - \frac{\sqrt{3}y}{3} + \frac{y}{3} + \frac{z}{3} + \frac{\sqrt{3}z}{3} \\ \frac{x}{3} + \frac{\sqrt{3}x}{3} + \frac{y}{3} - \frac{\sqrt{3}z}{3} + \frac{z}{3} \\ -\frac{\sqrt{3}x}{3} + \frac{x}{3} + \frac{y}{3} + \frac{\sqrt{3}y}{3} + \frac{z}{3} \end{bmatrix}$$

Pour obtenir la matrice de la rotation, il suffit de calculer les images des vecteurs de la base canonique par celle-ci.

[19]:
```
def matrice_rotation(omega, theta):
    omega = normaliser(omega)
    u = rotation(omega, theta, e1)
    v = rotation(omega, theta, e2)
    w = rotation(omega, theta, e3)
    return u.col_insert(1, v).col_insert(2, w)
```

[20]:
```
A = matrice_rotation(Matrix([1, 1, 1]), pi / 2)
A
```

[20]:
$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} - \frac{\sqrt{3}}{3} & \frac{1}{3} + \frac{\sqrt{3}}{3} \\ \frac{1}{3} + \frac{\sqrt{3}}{3} & \frac{1}{3} & \frac{1}{3} - \frac{\sqrt{3}}{3} \\ \frac{1}{3} - \frac{\sqrt{3}}{3} & \frac{1}{3} + \frac{\sqrt{3}}{3} & \frac{1}{3} \end{bmatrix}$$

1.3.2 2.2 Déterminer les caractéristiques d'une rotation

Il est facile de savoir si une matrice A est la matrice d'une rotation différente de l'identité.

[21]:
```
def est_rotation(A):
    return simplify(A.T * A) == eye(3) and A != eye(3) and simplify(det(A)) == 1
```

L'axe de la rotation est l'ensemble des invariants.

[22]:
```
def invariants(A):
    return (A - eye(3)).nullspace()[0]
```

[23]:
```
invariants(A)
```

[23]:
$$\begin{bmatrix} -\frac{-\left(\frac{1}{3}-\frac{\sqrt{3}}{3}\right)\left(-\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)^2-\frac{2}{9}+\frac{2\sqrt{3}}{9}\right)+\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)\left(-\left(\frac{1}{3}-\frac{\sqrt{3}}{3}\right)\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)+\frac{4}{9}\right)}{-\frac{8}{27}+\frac{2\left(\frac{1}{3}-\frac{\sqrt{3}}{3}\right)\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)}{3}} \\ -\frac{-\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)^2-\frac{2}{9}+\frac{2\sqrt{3}}{9}}{-\left(\frac{1}{3}-\frac{\sqrt{3}}{3}\right)\left(\frac{1}{3}+\frac{\sqrt{3}}{3}\right)+\frac{4}{9}} \\ 1 \end{bmatrix}$$

[24]:
```
e=simplify(_)
e
```

[24]:
$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Le cosinus de l'angle est donné par la trace de la rotation : $Tr\ A = 1 + 2\cos\theta$.

[25]:
```
def cos_angle(A):
    return (A.trace() - 1) / 2
```

[26]:
```
cos_angle(A)
```

[26]:
$$0$$

Pour obtenir le signe du sinus, on prend un vecteur u qui n'est pas sur l'axe et on calcule le déterminant de $u, f(u), e$ où e est un vecteur qui dirige l'axe (pas forcément unitaire). Le signe de $\sin\theta$ est alors le signe de ce daterminant.

[27]:
```
def signe_sin_angle(A, e):
    if e.cross(Matrix([1, 0, 0])) != 0:
        u = Matrix([1, 0, 0])
    else:
        u = Matrix([0, 1, 0])
    v = A * u
    M = Matrix([u.T, v.T, e.T]).T
    return M.det()
```

[28]:
```
signe_sin_angle(A, e)
```

[28]:
$$\frac{2\sqrt{3}}{3}$$

On regroupe le tout en une seule fonction. La fonction **analyse_rotation** prend en paramètre une matrice A censée être une matrice de rotation différente de l'identité. Elle renvoie le couple (e, θ) où e est un vecteur de l'axe (pas nécessairement unitaire) et θ est l'angle de la rotation.

[29]:
```
def analyse_rotation(A):
    e = simplify(invariants(A))
    c = cos_angle(A)
    theta = acos(c)
    s = signe_sin_angle(A, e)
    if s >= 0: return (e, theta)
    else: return (e, -theta)
```

[30]:
```
analyse_rotation(A)
```

[30]:
$$\left(\begin{bmatrix}1\\1\\1\end{bmatrix}, \frac{\pi}{2}\right)$$

Essayons deux autres exemples.

[31]:
```
A2 = Matrix([[3, 1, sqrt(6)],[1, 3, -sqrt(6)],[-sqrt(6),sqrt(6), 2]])/4
A2
```

[31]:
$$\begin{bmatrix}\frac{3}{4} & \frac{1}{4} & \frac{\sqrt{6}}{4}\\ \frac{1}{4} & \frac{3}{4} & -\frac{\sqrt{6}}{4}\\ -\frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} & \frac{1}{2}\end{bmatrix}$$

[32]:
```
est_rotation(A2)
```

[32]:
```
True
```

[33]:
```
analyse_rotation(A2)
```

[33]:
$$\left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \frac{\pi}{3}\right)$$

[34]:
```
A3 = Matrix([[8,1,-4],[-4,4,-7],[1,8,4]])/9
A3
```

[34]:
$$\begin{bmatrix}\frac{8}{9} & \frac{1}{9} & -\frac{4}{9}\\ -\frac{4}{9} & \frac{4}{9} & -\frac{7}{9}\\ \frac{1}{9} & \frac{8}{9} & \frac{4}{9}\end{bmatrix}$$

[35]:
```
est_rotation(A3)
```

[35]:
```
True
```

[36]:
```
analyse_rotation(A3)
```

[36]:
$$\left(\begin{bmatrix}-3\\1\\1\end{bmatrix}, -\operatorname{acos}\left(\frac{7}{18}\right)\right)$$

Remarquons pour terminer que l'identité, qui est une rotation, n'a pas d'axe défini de façon unique. Cela dit, notre fonction renvoie des résultats tout à fait logiques.

[37]:
```
analyse_rotation(eye(3))
```

[37]:
$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ 0 \right)$$

1.4 3 Composée d'une réflexion et d'une rotation

Soit f un endomorphisme orthogonal de l'espace qui n'est ni ni une rotation ni une réflexion et qui est différent de $-id$. Alors f s'crit de façon unique $g \circ h$ où g est une rotation différente de l'identité, h est une réflexion, et le plan de la réflexion h est orthogonal à l'axe de la rotation g. De plus, g et h commutent.

Pour obtenir les caractéristiques de f, on cherche tout d'abord l'ensemble D des vecteurs changés en leur opposé. Ce sera l'axe de g. On oriente D par un vecteur e. Le plan de h est alors l'orthogonal de e et h est complètement déterminée. Comme $f = g \circ h$, on a aussi $g = f \circ h$ et la détermination de g est alors facile.

[38]:
```
def analyse_reflex_rot(A):
    e = (A + eye(3)).nullspace()[0]
    B = matrice_reflexion(e) * A
    (e1, theta) = analyse_rotation(B)
    return (e, theta)
```

Prenons l'une des matrices de rotation vues ci-dessus et prenons l'opposé de sa première colonne.

[39]:
```
A4 = Matrix([[-3, 1, sqrt(6)],[-1, 3, -sqrt(6)],[sqrt(6),sqrt(6), 2]])/4
A4
```

[39]:
$$\begin{bmatrix} -\frac{3}{4} & \frac{1}{4} & \frac{\sqrt{6}}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{\sqrt{6}}{4} \\ \frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} & \frac{1}{2} \end{bmatrix}$$

[40]:
```
e, theta = analyse_reflex_rot(A4)
```

[41]:
```
e
```

[41]:
$$\begin{bmatrix} -\sqrt{6} \\ 0 \\ 1 \end{bmatrix}$$

[42]:
```
theta
```

[42]:
$$-\operatorname{acos}\left(\frac{3}{4}\right)$$

Vérifications …

[43]:
```
matrice_reflexion(e) * matrice_rotation(e, theta)
```

[43]:
$$\begin{bmatrix} -\frac{3}{4} & \frac{1}{4} & \frac{\sqrt{6}}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{\sqrt{6}}{4} \\ \frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} & \frac{1}{2} \end{bmatrix}$$

[44]:
```
matrice_rotation(e, theta) * matrice_reflexion(e)
```

[44]:
$$\begin{bmatrix} -\frac{3}{4} & \frac{1}{4} & \frac{\sqrt{6}}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{\sqrt{6}}{4} \\ \frac{\sqrt{6}}{4} & \frac{\sqrt{6}}{4} & \frac{1}{2} \end{bmatrix}$$

On retrouve bien la matrice A_4, ceci quel que soit le sens dans lequel on effectue les produits.

[45]:
```
analyse_rotation(Matrix([[1,0,0],[0,1,0],[0,0,1]]))
```

[45]:
$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ 0 \right)$$

Remarquons pour terminer que $-id$ se décompose aussi comme un produit de réflexion et de rotation, mais on n'a plus unicité.

[46]:
```
analyse_reflex_rot(-eye(3))
```

[46]:
$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ \pi \right)$$

1.5 4 Regroupons le tout

[47]:
```
def analyse(A):
    if simplify(A * A.T) != eye(3):
        return (0, [])
    elif est_reflexion(A):
        return (1, analyse_reflexion(A))
    elif est_rotation(A):
        return (2, analyse_rotation(A))
    else:
        return (3, analyse_reflex_rot(A))
```

[48]:
```
analyse(A)
```

[48]:
$$\left(2, \ \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \ \frac{\pi}{2} \right) \right)$$

[49]:
```
analyse(A4)
```

[49]:
$$\left(3,\ \left(\begin{bmatrix} -\sqrt{6} \\ 0 \\ 1 \end{bmatrix},\ -\operatorname{acos}\left(\frac{3}{4}\right)\right)\right)$$

[50]:
```
analyse(A0)
```

[50]:
$$\left(1,\ \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ 1 \end{bmatrix}\right)$$

[51]:
```
analyse(2*eye(3))
```

[51]:
$$(0,\ [])$$

[ ]:
```

```