Attitude Coordinate Sets

July 21, 2020

Rigid Body Kinematics: Examples and Excercises from "Anaylitical Mechanics of Space Systems" by H. Shaub, J. Junkins

1 Euler angles and Principal Rotation (PR) Parameters

[1]:
```
import numpy as np
import math
from DCMmatrix import *
from Euler_Integrator import *
from scipy.linalg import *
```

1.0.1 Example 3.3:

Let the B frame attitude relative to the $\mathcal{N}$ frame be given by the $(3-2-1)$ Euler angles $(10, 25, -15)$ deg Find the corresponding principal rotation axis and angles.

[2]:
```
Theta= (10, 25,-15)
euler = (3,2,1)
BN = DCMatrix(Theta, euler)
```

[3]:
```
print(BN)
```

```
[[ 0.89253894  0.1573787  -0.42261826]
 [-0.27545116  0.93225732 -0.23456972]
 [ 0.35707269  0.3257733   0.8754261 ]]
```

[4]:
```
Phi, e = DCM_to_PR(BN)
print('Rotation in degrees = ', Phi, 'Rotation vector = ', e)
```

```
Rotation in degrees =  31.77623650635435 Rotation vector =  [-0.53203527
0.74030206  0.4109639 ]
```

[5]:
```
print('The second principal rotation angle is Phi\' = 360 - Phi = ', 360 - Phi)
print('Both pairs (e, Phi), (e, Phi\') descibes the identical attitude given by␣
 ↪the Euler set (3-2-1)')
```

```
The second principal rotation angle is Phi' = 360 - Phi =  328.22376349364566
Both pairs (e, Phi), (e, Phi') descibes the identical attitude given by the
Euler set (3-2-1)
```

1.1 Quiz 2: Concept Check 2 - Principal Rotation Parameter, relation to DCM

Q6 Given the DCM

$$BN = \begin{pmatrix} 0.925417 & 0.33684 & 0.173648 \\ 0.0296956 & -0.521281 & 0.852869 \\ 0.377786 & -0.784102 & -0.492404 \end{pmatrix}$$

what is the equivalent rotation parameter set (e, Φ)

[6]:
```
BN = np.array([[0.925417, 0.33684, 0.173648],[0.0296956, -0.521281, 0.852869␣
 ↪],[0.377786, -0.784102, -0.492404]])
Phi, e = DCM_to_PR(BN)
print(Phi, e)
```

```
122.96550407043027 [0.97555081 0.12165579 0.18304232]
```

1.2 Quiz 3: principal rotation addition:

2. Given the $(3-2-1)$ Euler angles $(20, -10, 120)$ degrees, what is the equivalent principal rotational parameters?

[7]:
```
Phi, e = DCM_to_PR(DCMatrix((20,-10, 120), (3,2,1)))
```

[8]:
```
print(Phi*np.pi/180, e)
```

```
2.1461528375981986 [0.97555054 0.12165573 0.18303284]
```

3. Two orientations are related through $[FB] = [BN] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$. Sum these orientations to find $[FN]$ and express the answer using principal rotation parameters.

Note: the sum of orientations corresponds to the product of the DCM matrices (this is because sum in the Lie Algebra corresponds to product in the group, sort to speak!)

[9]:
```
FB = np.array([[1,0,0],[0,0,1],[0,-1,0]])
BN = FB
FN = FB @ BN
```

[10]:
```
print('FN  matrix is = '+ '\n', FN, '\n It represents a 180 degrees around the␣
 ↪axis e =(1,0,0)')
```

```
FN  matrix is =
 [[ 1  0  0]
 [ 0 -1  0]
```

```
 [ 0  0 -1]]
 It represents a 180 degrees around the axis e =(1,0,0)
```

[11]:
```
Phi1, e1= DCM_to_PR(BN)
Phi2, e2 = DCM_to_PR(FB)
Phi_12, e_12 = Add_PR(Phi1, e1, Phi2, e2)
print(Phi1,Phi2, Phi_12, e_12)
```

```
90.0 90.0 179.99999999999997 [1. 0. 0.]
```

print('Computing the rotation parameters directly from the product matrix FN conduces to errors because Phi = 180
is a singularity') Phi, e = DCM_to_PR(FN) print('The rotation parameters (Phi, e) are :', Phi, e)

2 Euler Parameters:

Another popular set of attitude coordinates are the four Euler parameters (quaternions). They provide a redundant, nonsingular attitude description and are well suited to describe arbitrary, large rotations. The Euler parameter vector b is defined in terms of the principal rotation elements as:

$$\beta_0 = \cos(\Phi/2)$$
$$\beta_1 = e_1 \sin(\Phi/2)$$
$$\beta_2 = e_2 \sin(\Phi/2)$$
$$\beta_3 = e_3 \sin(\Phi/2)$$

It is evident since $e_1^2 + e_2^2 + e_3^2 = 1$ that the β_i satisfy the holonomic constraint

$$\beta_0^2 + \beta_1^2 + \beta_2^2 + \beta_3^2 = 1$$

Note that this constraint geometrically describes a four-dimensional unit sphere. Any rotation described through the Euler parameters has a trajectory on the surface of this constraint sphere. Given a certain attitude, there are actu- ally two sets of Euler parameters that will describe the same orientation. This is due to the non-uniqueness of the principal rotation elements themselves. Switching between the sets $(\Phi, \hat{e})$ and $(-\Phi, -\hat{e})$ will yield the same Euler parameter vector β. However, if the second principal rotation angle Φ' is used, another Euler parameter vector β' is found. One can show that

$$\beta_0' = \cos(\Phi'/2) = \cos(\phi/2 - \pi) = -\cos(\Phi/2) = -\beta_0$$
$$\beta_i' = e_i \sin(\Phi'/2) = e_i \sin(\phi/2 - \pi) = -e_i \sin(\Phi/2) = -\beta_i$$

therefore the vector $\beta' = -\beta$ describes the same orientation as β. Because any point on the unit constraint sphere surface represents a specific orientation, the anti-pole to that point represents the exact same orientation. The difference between the two attitude descriptions is that one specifies the orientation through the shortest single axis rotation, the other through the longest.

The cosine direction matrix written in terms ofthe Euler parameters is:

$$C = \begin{bmatrix} \beta_0^2 + \beta_1^2 - \beta_2^2 - \beta_3^2 & 2(\beta_1\beta_2 + \beta_0\beta_3) & 2(\beta_1\beta_3 - \beta_0\beta_2) \\ 2(\beta_1\beta_2 - \beta_0\beta_3) & \beta_0^2 - \beta_1^2 + \beta_2^2 - \beta_3^2 & 2(\beta_2\beta_3 - \beta_0\beta_1) \\ 2(\beta_1\beta_3 + \beta_0\beta_2) & 2(\beta_2\beta_3 - \beta_0\beta_1) & \beta_0^2 - \beta_1^2 - \beta_2^2 + \beta_3^2 \end{bmatrix}$$

From the DCM matrix one can compute the Euler Parameters using Sheppard method:

$$\beta_0^2 = \frac{1}{4}(1 + \text{trace}(C))$$
$$\beta_1^2 = \frac{1}{4}(1 + 2C[0,0] - \text{trace}(C))$$
$$\beta_2^2 = \frac{1}{4}(1 + 2C[1,1] - \text{trace}(C))$$
$$\beta_3^2 = \frac{1}{4}(1 + 2C[2,2] - \text{trace}(C))$$

ThenSheppard takes the square root of thelargest β_i found in eqs above where the sign of β_i is arbitrarily chosen to be positive. The other β_j are found by dividing the appropriate three of the following six in the following eqs by the chosen largest β_i coordinate:

$$\beta_0\beta_1 = (C[1,2] - C[2,1])/4$$
$$\beta_0\beta_2 = (C[2,0] - C[0,2])/4$$
$$\beta_0\beta_3 = (C[0,1] - C[1,0])/4$$
$$\beta_2\beta_3 = (C[1,2] + C[2,1])/4$$
$$\beta_3\beta_1 = (C[2,0] + C[0,3])/4$$
$$\beta_1\beta_2 = (C[0,1] + C[1,0])/4$$

To find the alternate set of Euler parameter, the sign of the chosen β_i would simply be set negative.

2.1 Example 3.6 from Shaub-Junkins book:

Let's use the Sheppard's method to find the Euler parameters of the direction cosine matrix C given by:

[12]:
```
C = np.array([[0.892539, 0.157379, -0.422618],[-0.275451, 0.932257, -0.
↪234570],[0.357073, 0.325773, 0.875426]])
```

[13]:
```
print(C)
```

```
[[ 0.892539  0.157379 -0.422618]
 [-0.275451  0.932257 -0.23457 ]
 [ 0.357073  0.325773  0.875426]]
```

[14]:
```
beta = DCM_to_EP(C)
```

[15]:
```
print(beta)
```

```
[ 0.96179806 -0.14564986  0.20266494  0.11250543]
```

2.2 Addition and subtraction of Euler Parameters:

A very important composite rotation property of the Euler parameters is the manner in which they allow two sequential rotations to be combined into one overall composite rotation. Let the

Euler parameter vector β' describe the first, β'' the second, and β the composite rotation. In terms of the matrices:

$$FN(\beta) = FB(\beta'')BN(\beta')$$

using the expression for the above matrices one finds that

$$\begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\end{pmatrix} = \begin{bmatrix}\beta''_0 & -\beta''_1 & -\beta''_2 & -\beta''_3\\ \beta''_1 & \beta''_0 & \beta''_3 & \beta''_2\\ \beta''_2 & -\beta''_3 & \beta''_0 & \beta''_1\\ \beta''_3 & \beta''_2 & -\beta''_1 & \beta''_0\end{bmatrix}\begin{pmatrix}\beta'_0\\ \beta'_1\\ \beta'_2\\ \beta'_3\end{pmatrix}$$

the relation above is compactly written as: $\beta = G(\beta'')\beta'$ which is also equivalent to $\beta = G(\beta')\beta''$

2.3 Example 3.8 from the book:

Let

$$[BN] \to \beta' = \left(0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)^T$$

and

$$[FB] \to \beta'' = \left(\frac{1}{2}\sqrt{\frac{\sqrt{3}}{2}+1}, -\frac{1}{2}\sqrt{\frac{\sqrt{3}}{2}+1}, \frac{-\sqrt{2}}{4\sqrt{2}+\sqrt{3}}, \frac{\sqrt{2}}{4\sqrt{2}+\sqrt{3}}\right)^T$$

compute the EP β that describes the F frame orientation relative to the N frame: $FN = FB * BN$

[16]:
```
beta2 = np.array([0.5*np.sqrt(np.sqrt(3)/2 + 1), -0.5*np.sqrt(np.sqrt(3)/2 + 1),␣
↪-np.sqrt(2)/(4*np.sqrt(2)+np.sqrt(3)),np.sqrt(2)/(4*np.sqrt(2)+np.sqrt(3))])
```

[17]:
```
import scipy.linalg
from scipy import *
scipy.linalg.norm(beta2)
```

[17]:
```
1.003134214700055
```

[18]:
```
beta1 = np.array([0,1/np.sqrt(2),1/np.sqrt(2),0])
```

[19]:
```
beta = Add_EPs(beta1, beta2)
```

[20]:
```
print(beta)
```

```
[0.61830096 0.61830096 0.34762486 0.34762486]
```

[21]:
```
np.array([np.sqrt(3),np.sqrt(3),1,1])/(2*np.sqrt(2))
```

[21]:
```
array([0.61237244, 0.61237244, 0.35355339, 0.35355339])
```

2.4 Quiz 6: Concept check 5,6 - Euler Parameter Relationship to DCM:

1. given $\beta = (0.235702, 0.471405, -0.471405, 0.707107)$ what is the corresponding DCM?

[22]:
```
beta = np.array([0.235702,0.471405,-0.471405,0.707107])
C = EP_to_DCM(beta)
print(C)
```

```
[[-0.44444488 -0.11111228  0.88888975]
 [-0.77777842 -0.44444488 -0.44444535]
 [ 0.44444535 -0.88888975  0.11111039]]
```

3. Write a program to map a DCM to the equivalent EP. Choose the EP output set that describes a short rotation. Using

$$BN = \begin{bmatrix} -0.529403 & -0.467056 & 0.708231 \\ -0.474115 & -0.529403 & -0.703525 \\ 0.703525 & -0.708231 & 0.0588291 \end{bmatrix}$$

what is the corresponding EP set?

[23]:
```
BN = np.array([[-0.529403, -0.467056, 0.708231],[-0.474115, -0.529403, -0.
↪703525],[0.703525, -0.708231, 0.0588291]])
```

[24]:
```
EP = DCM_to_EP(BN)
print('The corresponding EP set is ', EP)
```

```
The corresponding EP set is  [ 0.00242542  0.48506963 -0.48506963  0.72760482]
```

4. Given the (3-2-1) Euler Angles set $(20, 10, -10)$ degrees, what is the equivalent EP set?

[25]:
```
DCM =  DCMatrix((20,10,-10),(3,2,1))
print('The DCM matrix of the euler angles is ', DCM)
```

```
The DCM matrix of the euler angles is  [[ 0.92541658  0.33682409 -0.17364818]
 [-0.36515929  0.91510341 -0.17101007]
 [ 0.10130573  0.2216648   0.96984631]]
```

[26]:
```
beta = DCM_to_EP(DCM)
print('Passing through the DCM matrix, the corresponding EP set is: ', beta)
```

```
Passing through the DCM matrix, the corresponding EP set is:  [ 0.97600798
-0.10058188  0.07042819  0.17980985]
```

2.5 Quiz 7: Concept Check 7- Euler Parameter Addition

1. Given $\beta_{B/N} = (0.774597, 0.258199, 0.516398, 0.258199)$ and $\beta_{FB} = (0.359211, 0.898027, 0.179605, 0.179605)$ what is $\beta_{F/N}$?

Here we used that in terms of the DCMs $FN = FB * BN$, then $\beta = G(\beta'')\beta'$ with $\beta'' = \beta_{FB}$ and $\beta' = \beta_{BN}$

[27]:
```
beta1 = np.array([0.774597,0.258199,0.516398,0.258199])
beta2 = np.array([0.359211,0.898027,0.179605,0.179605])
beta = Add_EPs(beta1, beta2)
```

```
print('The beta_{FN} set is : ', beta)
```

```
The beta_{FN} set is :  [-0.0927474   0.83473077  0.51011318 -0.1854961 ]
```

2. Given $\beta_{F/N} = (0.359211, 0.898027, 0.179605, 0.179605)$ and $\beta_{B/N} = (-0.377964, 0.755929, 0.377964, 0.377964)$ what is $\beta_{F/B}$. To compute it: $\beta_{F/B} = \beta_{F/N}\beta_{N/B}$ where $\beta_{N/B}$ is the EP set associated to the transpose of the DCM matrix associated to $\beta B/N$

$\beta'' = G(\beta')^T\beta$ with $\beta = \beta_{F/N}, \beta' = \beta_{B/N}$

[28]:
```
beta = np.array([0.359211,0.898027,0.179605,0.179605])
beta1 = np.array([-0.377964,0.755929,0.377964,0.377964])
beta2 = Subtract_EPs(beta, beta1)
print('beta_{F/B} is :', beta2)
```

```
beta_{F/B} is : [ 6.78844274e-01 -6.10959889e-01 -4.07306300e-01
1.98359000e-07]
```

2.5.1 using DCM's to compute $\beta_{F/B}$:

[29]:
```
FN = EP_to_DCM(beta)
BN = EP_to_DCM(beta1)
FB = FN@(BN.T)
beta22 = DCM_to_EP(FB)
print('In this way beta_{F/B} is :', beta22)
```

```
In this way beta_{F/B} is : [ 6.78844282e-01 -6.10959882e-01 -4.07306295e-01
1.98358998e-07]
```

2.6 Quiz 8: Concept Check 8 - EP Dfferential Equations:

Given $\beta(0) = (0.408248, 0., 0.408248, 0.816497)$, write a program to integrate the EP differential kinematic equations. Assume the body angular velocity vector of the craft is given through the B frame components as

$$\omega^B = (\sin(0.1t), 0.01t, \cos(0.01t))^T * 20deg/s$$

Compute the norm of the EP set at time $t = 42$

[30]:
```
from Euler_Integrator import *
beta_0 = np.array([0.408248,0.,0.408248,0.816497])
def EP_kinematics(t, beta0):
    w = (20*np.pi/180)*np.array([math.sin(0.1*t),0.01*t, math.cos(0.01*t)])
    W = np.array([[0, -w[0], -w[1], -w[2]],[w[0], 0, w[2], -w[1]],[w[1], -w[2],␣
 ↪0, w[0]], [w[2], w[1], -w[0], 0]])
    rhs = 0.5*W@ beta0

    return rhs
t0 = 0
tf = 42
```

```
step = 0.1

Beta, _ = euler_integrator(EP_kinematics, beta_0, (t0, tf), step)
Beta4, _ = ODE4_order(EP_kinematics, beta_0, (t0, tf), step)
```

[31]:
```
Beta.shape
```

[31]:
```
(4, 420)
```

[32]:
```
print('Using Euler method, the quantity sqrt{b1^2 + b^2 + b3^2} at t = 42s is ',␣
 ↪np.sqrt(Beta[1,-1]**2 + Beta[2, -1]**2 + Beta[3,-1]**2))
print('Using RK4, the quantity sqrt{b1^2 + b^2 + b3^2} at t = 42s is ', np.
 ↪sqrt(Beta4[1,-1]**2 + Beta4[2, -1]**2 + Beta4[3,-1]**2))
```

```
Using Euler method, the quantity sqrt{b1^2 + b^2 + b3^2} at t = 42s is
0.7252038671263483
Using RK4, the quantity sqrt{b1^2 + b^2 + b3^2} at t = 42s is
0.6634765771677069
```

[33]:
```
scipy.linalg.norm(Beta[:,-1],2)
```

[33]:
```
1.096894205033133
```

[34]:
```
scipy.linalg.norm(Beta4[:,-1],2)
```

[34]:
```
1.0000001047083895
```

3 Classical Rodrigues Parameters

The origin of the classical Rodrigues parameter vector q (or Gibbs vector) dates back over a hundred years to the French mathematician O. M. Rodrigues. This rigid body attitude coordinate set reduces the redundant Euler parameters to a minimal three-parameter set through the transformation

$$q_i = \frac{\beta_i}{\beta_0}, \quad i = 1,2,3 \tag{1}$$

The inverse transformation from classical Rodrigues parameters to Euler para- meters is given by

$$\beta_0 = \frac{1}{\sqrt{1+\mathbf{q}^{\mathrm{T}}\mathbf{q}}}$$
$$\beta_i = \frac{q_i}{\sqrt{1+\mathbf{q}^{\mathrm{T}}\mathbf{q}}} \quad i = 1,2,3$$

Using the definition of EP's in terms of the principal rotation angle Φ we have the elegant and useful transformation

$$\mathbf{q} = \tan\frac{\Phi}{2}\hat{\mathbf{e}}$$

it is evident that the classical Rodrigues para- meters go singular whenever $\Phi \to \pm 180$ deg.

the classical Rodrigues parameters can be viewed as a special set of stereographic orientation parameters. Stereographic projections are used to map a higher-dimension spherical surface onto a lower-dimension hyperplane. In this case, the surface of the four-dimension Euler parameter unit constraint sphere $\beta_0^2 + \beta_1^2 + \beta_2^2 + \beta_3^2 = 1$ is mapped (projected) onto a three-dimensional hyperplane.

3.1 Quiz Concept Check 11, 12 - CRP Addition.

1. Given the CRP attitude description $\mathbf{q}_{B/N} = (0.1, 0.2, 0.3)$ what is the equivalent DCM value?

[35]:
```
q = np.array([0.1,0.2, 0.3]) # very important to make sure q is 2d array
C = CRP_to_DCM(q)
print('The equivalent DCM description is : ' + '\n', C)
```

```
The equivalent DCM description is :
 [[ 0.77192982  0.56140351 -0.29824561]
 [-0.49122807  0.8245614   0.28070175]
 [ 0.40350877 -0.07017544  0.9122807 ]]
```

2. Given the following DCM matrix, what is the equivalent CRP set?

[36]:
```
BN = np.array([[0.333333, -0.666667,0.666667], [0.871795, 0.487179, 0.0512821],␣
 ↪[-0.358974, 0.564103, 0.74359]])
q = DCM_to_CRP(BN)
print('The equivalent CRP set is : \n', q)
```

```
The equivalent CRP set is :
 [[-0.2000002 ]
 [-0.40000008]
 [-0.60000031]]
```

3. Given $\mathbf{q}_{F/N} = (0.1, 0.2, 0.3)$ and $\mathbf{q}_{B/N} = (-0.3, 0.3, 0.1)$. What is the attitude of B relative to F in ters of CRP's?

We know that $FN(q_{FN}) = FB(q_{FB})BN(q_{BN})$, then $BF(q_{B/F}) = BN(q_{B/N})FN(q_{F/N})^T = BN(q_{B/N})FN(-q_{F/N})$ therefore $q_{B/F} = q_{B/N}" - "q_{F/N}$

[37]:
```
q_FN = np.array([0.1, 0.2, 0.3])
q_BN = np.array([-0.3,0.3,0.1])
q_BF = Add_CRPs(-q_FN, q_BN)
print('The CRP set is q_BF = ', q_BF)
```

```
The CRP set is q_BF =  [-0.31132075  0.18867925 -0.27358491]
```

[38]:
```
print('one can also use this formula to ge the subtraction CRP:')
(-q_FN + q_BN - np.cross(q_FN, q_BN))/(1+q_FN.T@q_BN)
```

```
one can also use this formula to ge the subtraction CRP:
```

[38]:
```
array([-0.31132075,  0.18867925, -0.27358491])
```

3.2 Quiz Concept Check 13 - CRP Differential Kinematic Equations.

1. In the CRP differential kinematic equations, is the $B(\mathbf{q})$ matrix orthogonal? Answer: NO.

$$B(\mathbf{q}) = \frac{1}{2}(\mathrm{I} + [\tilde{\mathbf{q}}] + \mathbf{q}\mathbf{q}^T$$
$$B^{-1}(\mathbf{q}) = \frac{2}{1+\mathbf{q}^T\mathbf{q}}(I - [\tilde{\mathbf{q}}])$$

2. Given $\mathbf{q}(0) = (0.4, 0.2, -0.1)$ Integrate the differential equation $\dot{\mathbf{q}} = B(q)\omega^B$, where the body angular velocity is: $\omega^B = (\sin(0..1t), 0.01, \cos(0.1t))3\,\mathrm{deg}/s$.

above we define the function to compute the r.h.s of the differential equation:

[39]:
```
def CRP_kinematics(t, q0):
    """
    this function computes the r.h.s B(q)w
    """
    w = (3*np.pi/180)*np.array([math.sin(0.1*t),0.01, math.cos(0.1*t)])
    rhs = 0.5*(np.eye(3) + Matrix_tilde(q0) + np.outer(q0,q0))@w

    return rhs

q0 = np.array([0.4, 0.2, -0.1])
t_span = (0,42)
step = 0.1
Q, _ = ODE4_order(CRP_kinematics, q0, t_span, step)

print('The CRP norm |q| at 42 seconds is ', scipy.linalg.norm(Q[:,-1],2))
```

```
The CRP norm |q| at 42 seconds is  1.2031251132171317
```

4 Modified Rodrigues Parameters

The modified Rodrigues parameters (MRPs) are an elegant recent addition to the family of attitude parameters. The MRP vector s is defined in terms of the Euler parameters as the transformation

$$\sigma_i = \frac{\beta_i}{1+\beta_0}, \quad i = 1, 2, 3.$$

They are simply the stereographic projection of the sphere $\beta_0^2 + \beta_1^2 + \beta_2^2 + \beta_3^2 = 1$ onto a hyperplane $\{\sigma_1, \sigma_2, \sigma_3\}$. The inverse transformation is:

$$\beta_0 = \frac{1-|\sigma|^2}{1+|\sigma|^2}, \quad \beta_i = \frac{2\sigma_i}{1+|\sigma|^2}, \quad i = 1, 2, 3.$$

he MRP vector can be expressed in terms of the principal rotation elements as:

$$\sigma = \tan\frac{\Phi}{4}\hat{\mathbf{e}}$$

By the above equation is evident that the MRP has a geometric singularity at $\Phi = \pm 360\,\text{deg}$. Any rotation can be described except a complete revolution back to the original orientation. This gives σ twice the rotational range of the classical Rodrigues parameters. Also note that for small rotations the MRPs linearize as $\sigma \approx \Phi/4$.

The MRP vector σ can be transformed directly into the classical Rodrigues parameter vector $\mathbf{q}$ through

$$\mathbf{q} = \frac{\mathbf{2\text{œ}}}{\mathbf{1} - |\text{œ}|^2}$$

with inverse transformation:

$$\text{œ} = \frac{\mathbf{q}}{1 + \sqrt{1 + \mathbf{q}^\mathbf{T}\mathbf{q}}}$$

Naturally, these transformations are singular at $\Phi = \pm 180\,\text{deg}$ because the classical Rodrigues parameters are singular at this orientation.

Note that the corresponding MRP to $-\mathbf{fi}$ is differente to the one associated to β as opossed to the case of CRP.

One set of MRPs always corresponds to a principal rotation $\Phi \leq 180\,\text{deg}$ and the other to $\Phi \geq 180\,\text{deg}$

$$|\sigma| < 1 \;\text{ if } \Phi < 180\,\text{deg}$$
$$|\sigma| = 1 \;\text{ if } \Phi = 180\,\text{deg}$$
$$|\sigma| > 1 \;\text{ if } \Phi > 180\,\text{deg}$$

As one set of MRPs exits the unit sphere, the other (shadow) set enters. The mapping can be written in terms of the principal rotation elements using the definitions of β_i as:

$$\sigma^S = \tan(\frac{\Phi - 2\pi}{4})\hat{\mathbf{e}} = \tan(\frac{\Phi'}{4})\hat{\mathbf{e}}$$

4.1 Direction Cosine MAtrix:

The direction cosine matrix in terms of the MRP is

$$C(\sigma) = I + \frac{8[\tilde{\sigma}]^2 - 4(1 - |\sigma|^2)[\tilde{\sigma}]}{(1 + |\sigma|^2)^2}$$

4.2 Concept Check 18 - MRP to DCM Relation

1. Map $\sigma = (0.1, 0.2, 0.3)$ to the corresponding DCM

[40]:
```
sigma = np.array([0.1,0.2,0.3])
C = MRP_to_DCM(sigma)
print('The corresponding DCM is C = ', C)
```

```
The corresponding DCM is C =  [[ 0.19975377  0.91720529 -0.34472145]
 [-0.67097568  0.38442598  0.63404124]
 [ 0.71406587  0.10464758  0.69221299]]
```

2. Map the DCM to the corresponding short MRP set:

[41]:
```
C = np.array([[0.763314, 0.0946746, -0.639053], [-0.568047, -0.372781, -0.
↪733728], [-0.307692, 0.923077, -0.230769]])
sigma_short = DCM_to_MRP(C)
print('The corresponding short MRP set is : ', sigma_short)
```

```
The corresponding short MRP set is :  [-0.49999988  0.09999998  0.19999983]
```

4.3 Addition and Subtraction of MRP's:

The MRPs enjoy the same relative rotation identity as did the classical Rodrigues parameters

$$C(\sigma)^T = C(-\sigma)$$

Given two MRP's vectors σ_1, σ_2, let the overall MRP vector σ be defined by:

$$FN(\sigma) = FB(\sigma_2)BN(\sigma_1)$$

then

$$\sigma = \frac{(1-|\sigma_1|^2)\sigma_2 + (1-|\sigma_2|^2)\sigma_1 - 2\sigma_2 \times \sigma_1}{1 + |\sigma_1|^2|\sigma_2|^2 - 2\sigma_1 \cdot \sigma_2}$$

And the subtraction, i.e, the attitude vector σ_2 in terms of σ and σ_1 as:

$$\sigma_2 = \frac{(1-|\sigma_1|^2)\sigma - (1-|\sigma_2|^2)\sigma_1 + 2\sigma \times \sigma_1}{1 + |\sigma_1|^2|\sigma_2|^2 + 2\sigma_1 \cdot \sigma}$$

obtained by changing $\sigma_1 \to -\sigma_1$

4.4 Concept Check 19 - MRP Addition and Subtraction

1. If $\sigma_{B/N} = (0.2, 0.2, -0.1)$ what is the equivalent inverse rotation $\sigma_{N/B}$?

[42]:
```
sigma = np.array([0.2,0.2,-0.1])
C_sigma = MRP_to_DCM(sigma)
sigma_inverse = DCM_to_MRP(C_sigma.T)
print('As expected, the inverse rotation is :', sigma_inverse)
```

```
As expected, the inverse rotation is : [-0.2 -0.2  0.1]
```

2. Add $\sigma_{B/N} = (0.1, 0.2, 0.3)$ and $\sigma_{R/B} = (-0.1, 0.3, 0.1)$ to find $\sigma_{R/N}$):

$$RN(\sigma_{R/N}) = RB(\sigma_{R/B})BN(\sigma_{B/N})$$

then in the function Add_MRPs sigma1 =$\sigma_{B/N}$, sigma2= $\sigma_{R/B}$

[43]:
```
sigma1 = np.array([0.1, 0.2, 0.3])
sigma2 = np.array([-0.1, 0.3, 0.1])
sigma = Add_MRPs(sigma1, sigma2)
print('The sum of sigma1 and sigma2 attitudes is :', sigma)
```

```
The sum of sigma1 and sigma2 attitudes is : [-0.16015899  0.41617957
0.52957681]
```

3. Given $\sigma_{B/N} = (0.1, 0.2, 0.3)$ and $\sigma_{R/N} = (0.5, 0.3, 0.1)$ find $\sigma_{B/R}$):

$$BR(\sigma_{B/R}) = BN(\sigma_{B/N})RN(\sigma_{R/N})^T = BN(\sigma_{B/N})RN(-\sigma_{R/N})$$

so the subtraction attitude is just the sum of $\sigma_{B/N}$ with $-\sigma_{R/N}$

[44]:
```
sigma_RN = np.array([0.5,0.3,0.1])
sigma_BN = np.array([0.1, 0.2, 0.3])
sigma_BR = Subtract_MRPs(sigma_RN, sigma_BN)
print('Then sigma_B/R is :', sigma_BR)
```

```
Then sigma_B/R is : [-0.37998495  0.11437171 -0.02332581]
```

4.5 MRP kinematic differential equation:

The kinematic differential equation of the MRPs is found in a similar manner as the one for the classical Rodrigues parameters. The resulting differential equation is:

$$\dot{\sigma} = \frac{1}{4}((1 - |\sigma|^2)I + 2[\tilde{\sigma}] + 2\sigma\sigma^T)\omega = \frac{1}{4}B(\sigma)\omega$$

4.5.1 Concept Check 20 - MRP Differential Kinematic Equation

4. Given $\sigma(0) = (0.4, 0., -0.1)$ write a program to integrate the MRP differential kinematics equations. Assume the body angular velocity vector of the craft is given through the B frame components as $\omega^B = (\sin(0.1t), 0.01, \cos(0.1t))^T 20 \deg/s$ Be sure to only use the short rotation version of the MRPs.

[45]:
```
def MRP_kinematics(t, sigma0):
    B_sigma = 0.25*((1- norm(sigma0,2)**2)*np.eye(3) +2*Matrix_tilde(sigma0)␣
 ↪+2*np.outer(sigma0,sigma0))
    w = np.array([math.sin(0.1*t), 0.01, math.cos(0.1*t)])*20*np.pi/180
    rhs = B_sigma @ w

    return rhs

## we write a special integrator for the MRP so we can check at every step time␣
 ↪whether sigma is a long or a short attitude
def MRP_integrator(function, x0, t_span, step) :
    x = x0
    t0, t_f = t_span
    t = np.arange(t0, t_f, step)
    nt = t.shape[0]
    X = np.zeros((x0.shape[0],nt))
    X[:,0] = x

    for i in range(nt-1):
        k1 = function(t[i],x)
        k2 = function(t[i] + 0.5*step, x + 0.5*step*k1)
        k3 = function(t[i] + 0.5*step, x + 0.5*step*k2)
```

```
        k4 = function(t[i] + step, x + step*k3)
        x = x + step*(k1 +2*k2 + 2*k3 + k4)/6
        if norm(x,2) > 1 :
            x = -x/(norm(x,2)**2)
            #print('flag', i)
        X[:,i+1] = x

    return X,t

sigma0 = np.array([0.4,0.2,-0.1])
t_span = (0,42)
step = 0.1

Sigma, _ = MRP_integrator(MRP_kinematics, sigma0, t_span, step)
```

[46]:
```
print('the MRP norm |sigma(42)| at 42 seconds is :', norm(Sigma[:,-1],2))
```

```
the MRP norm |sigma(42)| at 42 seconds is : 0.6400528858982075
```

[ ]:
```

```