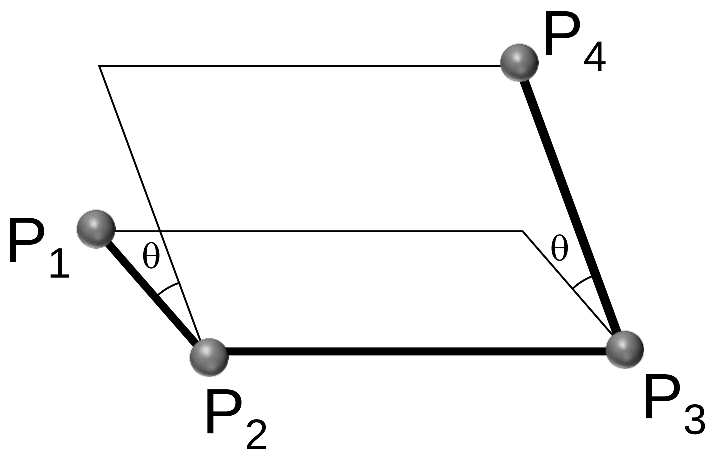
www.python.org

Dihedral Angle in Python

By Prof. Walter F. de Azevedo Jr.

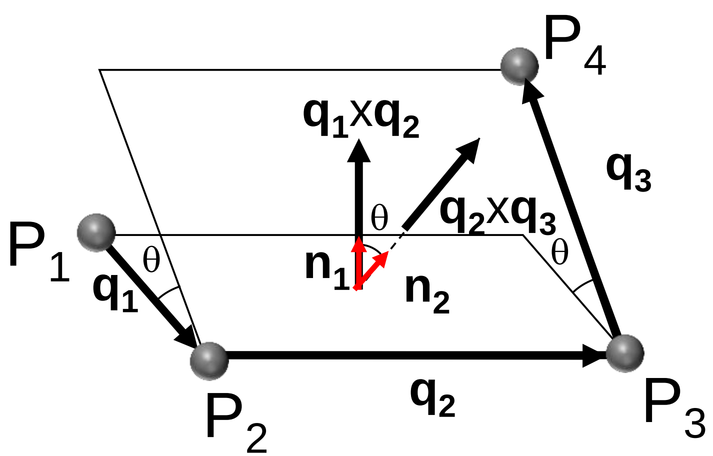
This tutorial presents the equation for the dihedral angle and describes its implementation in the Python programming language.

A dihedral angle is defined by four non collinear points, as shown in the figure here. Points P_1, P_2, and P_3 define the plane $P_1P_2P_3$, points P_2, P_3, and P_4 define a second plane, referred to as $P_2P_3P_4$. The angle between these two planes is referred to as dihedral angle θ.



To determine the dihedral angle θ, we need to consider the three vectors connecting four points P_1, P_2, P_3 , and P_4, named here as vectors $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_3}$. **Boldface** is used to indicate vectors. The cross product of $\mathbf{q_1}$ and $\mathbf{q_2}$ ($\mathbf{q_1}$x$\mathbf{q_2}$) defines a vector perpendicular to the plane $P_1P_2P_3$, and the cross product $\mathbf{q_2}$x$\mathbf{q_3}$ defines a vector normal to the plane $P_2P_3P_4$. In the figure shown here, we clearly see that the angle between $\mathbf{q_1}$x$\mathbf{q_2}$ and $\mathbf{q_2}$x$\mathbf{q_3}$ is also θ. Therefore, we just have to determine the angle between $\mathbf{n_1}$ and $\mathbf{n_2}$, which are the unit vectors along $\mathbf{q_1}$x$\mathbf{q_2}$ and $\mathbf{q_2}$x$\mathbf{q_3}$, respectively. Below we have equations used to calculate the unit vectors $\mathbf{n_1}$ and $\mathbf{n_2}$,

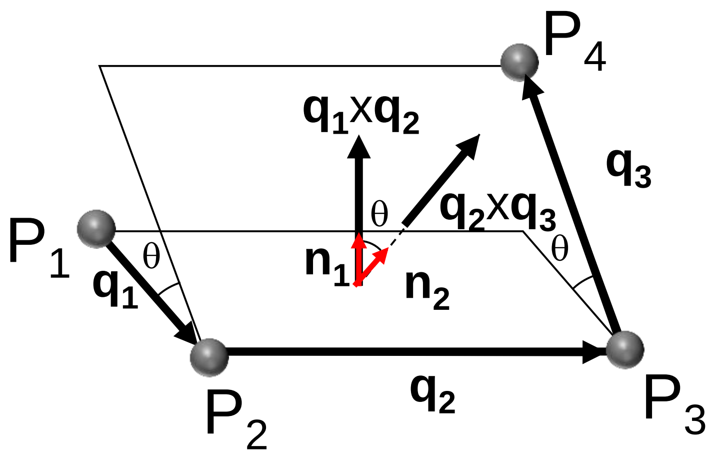
$$\boxed{\mathbf{n}_1 = \frac{\mathbf{q}_1 \times \mathbf{q}_2}{|\mathbf{q}_1 \times \mathbf{q}_2|}} \qquad \boxed{\mathbf{n}_2 = \frac{\mathbf{q}_2 \times \mathbf{q}_3}{|\mathbf{q}_2 \times \mathbf{q}_3|}}$$



Boldface is used to indicate vectors.

In addition, we define orthogonal unit vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ as follows:

$$\boxed{\begin{aligned} \mathbf{u}_1 &= \mathbf{n}_2 \\ \mathbf{u}_3 &= \frac{\mathbf{q}_2}{|\mathbf{q}_2|} \\ \mathbf{u}_2 &= \mathbf{u}_3 \times \mathbf{u}_1 \end{aligned}}$$

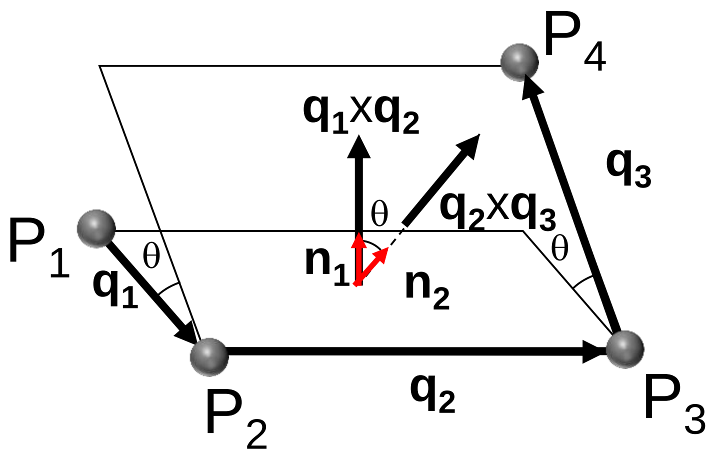


The cosine and sine are given by:

$$\cos -\theta = \mathbf{n_1 . u_1}$$
$$sen - \theta = \mathbf{n_1 . u_2}$$

Then, dihedral angle θ is as follows,

$$\theta = -a\tan 2\left(\frac{\mathbf{n_1 \cdot u_2}}{\mathbf{n_1 \cdot u_1}}\right)$$



You have to use *atan2* function, which is available in Python (https://docs.python.org/3/library/math.html), to determine the dihedral angle θ.

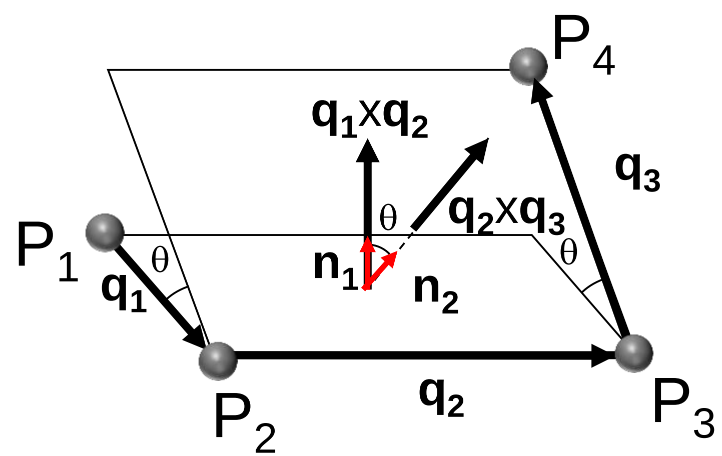
In summary, to calculate the dihedral angle θ for a system with four points, as shown in the figure here, we have to follow the steps shown below.

1) Calculate vectors $\mathbf{q_1}$, $\mathbf{q_2}$ and $\mathbf{q_3}$ as follows:

$$\mathbf{q_1} = (x_2 - x_1)\mathbf{i} + (y_2 - y_1)\mathbf{j} + (z_2 - z_1)\mathbf{k}$$

$$\mathbf{q_2} = (x_3 - x_2)\mathbf{i} + (y_3 - y_2)\mathbf{j} + (z_3 - z_2)\mathbf{k}$$

$$\mathbf{q_3} = (x_4 - x_3)\mathbf{i} + (y_4 - y_3)\mathbf{j} + (z_4 - z_3)\mathbf{k}$$



where **i**, **j**, and **k** are orthogonal unit vectors along x, y, and z axes, respectively. We consider an orthogonal coordinate system.

2) Calculate cross vectors $\mathbf{q_1} \times \mathbf{q_2}$ and $\mathbf{q_2} \times \mathbf{q_3}$.

$$\boxed{\begin{array}{l}\mathbf{q_1} \times \mathbf{q_2} \\ \mathbf{q_2} \times \mathbf{q_3}\end{array}}$$

3) Calculate vectors $\mathbf{n_1}$ and $\mathbf{n_2}$ normal to planes defined by points P_1, P_2, P_3, and P_4, as follows:

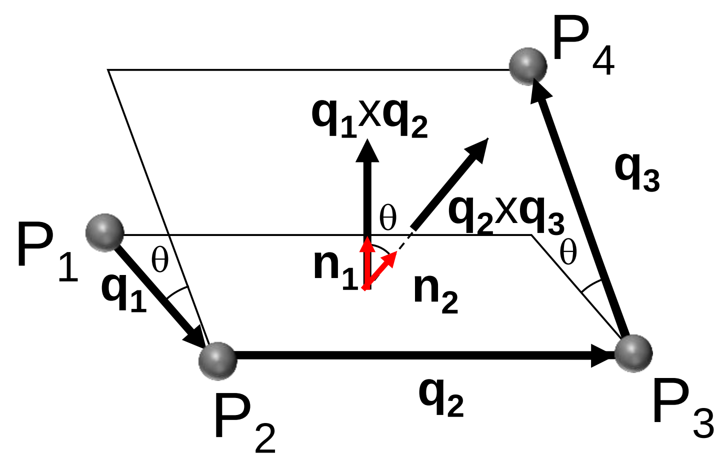
$$\mathbf{n}_1 = \frac{\mathbf{q}_1 \times \mathbf{q}_2}{|\mathbf{q}_1 \times \mathbf{q}_2|} \qquad \mathbf{n}_2 = \frac{\mathbf{q}_2 \times \mathbf{q}_3}{|\mathbf{q}_2 \times \mathbf{q}_3|}$$

4) Calculate orthogonal unit vectors, as indicated below:

$$\mathbf{u}_1 = \mathbf{n}_2$$

$$\mathbf{u}_3 = \frac{\mathbf{q}_2}{|\mathbf{q}_2|}$$

$$\mathbf{u}_2 = \mathbf{u}_3 \times \mathbf{u}_1$$



5) Calculate dihedral angle θ, as indicated here:

$$\cos-\theta = \mathbf{n}_1 . \mathbf{u}_1$$

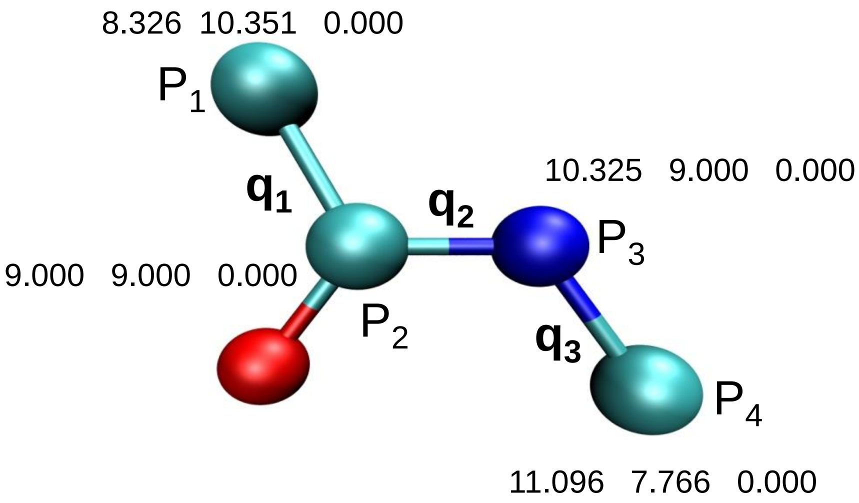
$$sen - \theta = \mathbf{n}_1 . \mathbf{u}_2$$

$$\theta = -a\tan 2\left(\frac{\mathbf{n}_1 \cdot \mathbf{u}_2}{\mathbf{n}_1 \cdot \mathbf{u}_1}\right)$$

The algorithm to calculate the dihedral angle is as follows:

```
Read point coordinates
Calculate vector q1,q2, and q3
Calculate cross vector q1xq2 and q2xq3
Calculate normal vectors n1 and n2
Calculate orthogonal unit vectors
Calculate sin(theta) and cos(theta)
Calculate atan2(sin(theta),cos(theta) )
Show results
```

Example. Calculate the dihedral angle θ between the planes defined by the points P_1, P_2, P_3 and P_4, using the coordinates indicated below.



$$\mathbf{p}_1 = 8.326\mathbf{i} + 10.351\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p}_2 = 9.000\mathbf{i} + 9.000\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p}_3 = 10.325\mathbf{i} + 9.000\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p}_4 = 11.096\mathbf{i} + 7.766\mathbf{j} + 0.000\mathbf{k}$$

We could think that each coordinate is an atomic coordinate, as the ones available in a file following the Protein Data Bank format (Berman, Westbrook, Feng *et al*. 2000; Berman, Battistuz, Bhat *et al*. 2002; Westbrook *et al*., 2003).

Answer

Step 1: Here we calculate the vectors $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_3}$.

$$\mathbf{q_1} = (x_2 - x_1)\mathbf{i} + (y_2 - y_1)\mathbf{j} + (z_2 - z_1)\mathbf{k} = (0.674)\mathbf{i} + (-1.351)\mathbf{j} = 0.674\mathbf{i} - 1.351\mathbf{j}$$

$$\mathbf{q_2} = (x_3 - x_2)\mathbf{i} + (y_3 - y_2)\mathbf{j} + (z_3 - z_2)\mathbf{k} = (1.351)\mathbf{i} + (0)\mathbf{j} = 1.351\mathbf{i}$$

$$\mathbf{q_3} = (x_4 - x_3)\mathbf{i} + (y_4 - y_3)\mathbf{j} + (z_4 - z_3)\mathbf{k} = (0,.771)\mathbf{i} + (-1.234)\mathbf{j} = 0.771\mathbf{i} -1.234\mathbf{j}$$

Step 2: Now we calculate the cross vectors, as follows:

$\mathbf{q}_1$ x $\mathbf{q}_2$ = (0.674**i** – 1.351**j**) x (1.351**i**) = 1.8252**k**

$\mathbf{q}_2$ x $\mathbf{q}_3$ = (1.351**i**) x (0.771**i** -1.234**j**) = - 1.6671**k**

Step 3: Here we calculate the normal vectors:

$$\mathbf{n}_1 = \frac{\mathbf{q}_1 \times \mathbf{q}_2}{|\mathbf{q}_1 \times \mathbf{q}_2|} = \frac{1.8252\mathbf{k}}{1.8252} = \mathbf{k}$$

$$\mathbf{n}_2 = \frac{\mathbf{q}_2 \times \mathbf{q}_3}{|\mathbf{q}_2 \times \mathbf{q}_3|} = \frac{\text{-}1.6671\mathbf{k}}{1.6671} = \text{-}\mathbf{k}$$

Cross vectors:

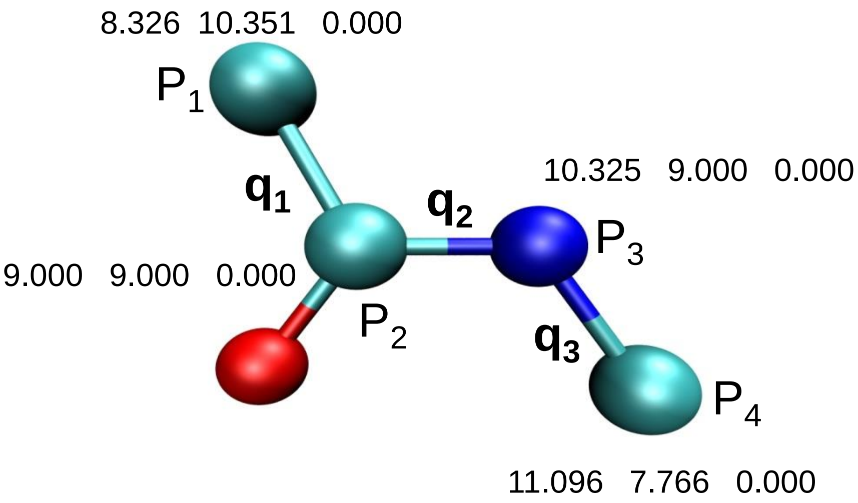
i x **i** = 0	**j** x **j** = 0	**k** x **k** = 0
i x **j** = **k**	**j** x **k** = **i**	**k** x **i** = **j**
i x **k** = - **j**	**j** x **i** = -**k**	**k** x **j** = -**i**

Step 4: Calculate unit vectors:

$$\mathbf{u}_1 = \mathbf{n}_2 = \text{-}\mathbf{k}$$

$$\mathbf{u}_3 = \frac{\mathbf{q}_2}{|\mathbf{q}_2|} = \frac{1.351\mathbf{i}}{1.351} = \mathbf{i}$$

$$\mathbf{u}_2 = \mathbf{u}_3 \times \mathbf{u}_1 = \mathbf{i} \times (\text{-}\mathbf{k}) = \mathbf{j}$$



Step 5.: Finally, the dihedral angle

$$\cos-\theta = \mathbf{n}_1.\mathbf{u}_1 = \mathbf{k} \cdot (-\mathbf{k}) = \text{-}1$$

$$sen - \theta = \mathbf{n}_1.\mathbf{u}_2 = \mathbf{k} \cdot \mathbf{j} = 1$$

$$\boxed{\theta = -a\tan 2\left(\frac{1}{-1}\right) = -180^o}$$

Dihedral angle for system with four points

Program: *dihedral_angle.py*

Abstract

Program to calculate the dihedral angle in degrees for a system with four points (P_1, P_2 ,P_3, P_4). The dihedral angle is between two planes, the first defined by the points P_1, P_2 and P_3 and the second plane by the points P_2, P_3 and P_4. The results are shown on screen.

In the main program we call the following functions: *initial_vectors()*, *calc_q_vectors(p1,p2,p3,p4)*, *calc_cross_vectors(q1,q2,q3)*, *calc_nornalss(q1_x_q2,q2_x_q3)*, *calc_orthogonal_unit_vectors(n2,q2)*, and *calc_dihedral_angle(n1,u1,u2,u3).*

```
def main():
    # Call initial_vectors() functions
    p1,p2,p3,p4 = initial_vectors()
    # Call calc_q_vectors(p1,p2,p3,p4) function
    q1,q2,q3 = calc_q_vectors(p1,p2,p3,p4)
    # Call calc_cross_vectors(q1,q2,q3) function
    q1_x_q2, q2_x_q3 = calc_cross_vectors(q1,q2,q3)
    # Call calc_nornalss(q1_x_q2,q2_x_q3) function
    n1, n2 = calc_nornals(q1_x_q2,q2_x_q3)
    # Call calc_orthogonal_unit_vectors(n2,q2) function
    u1,u2,u3 = calc_orthogonal_unit_vectors(n2,q2)
    # Call calc_dihedral_angle(u1,u2,u3) function
    calc_dihedral_angle(n1,u1,u2,u3)
main()
```

In this function, we define the coordinates for four points and return them to the main program. We use *NumPy* (Bressert, 2013; Idris, 2012) arrays for the coordinates.

```
def initial_vectors():
    """Function to set up initial vectors"""
    import numpy as np

    # Set initial values for arrays
    p1 = np.zeros(3)
    p2 = np.zeros(3)
    p3 = np.zeros(3)
    p4 = np.zeros(3)

    # Set initial coordinates (http://www.stem2.org/je/proteina.pdf)
    p1[:] = [8.326, 10.351, 0.000]
    p2[:] = [9.000, 9.000, 0.000]
    p3[:] = [10.325, 9.000, 0.000]
    p4[:] = [11.096, 7.766, 0.000]

    return p1,p2,p3,p4
```

We set initial values for the arrays using *np.zeros(3),* which defines a vector filled with zeros, then we assign values to the arrays, as shown below.

```
def initial_vectors():
    """Function to set up initial vectors"""
    import numpy as np

    # Set initial values for arrays
    p1 = np.zeros(3)
    p2 = np.zeros(3)
    p3 = np.zeros(3)
    p4 = np.zeros(3)

    # Set initial coordinates (http://www.stem2.org/je/proteina.pdf)
    p1[:] = [8.326, 10.351, 0.000]
    p2[:] = [9.000, 9.000, 0.000]
    p3[:] = [10.325, 9.000, 0.000]
    p4[:] = [11.096, 7.766, 0.000]

    return p1,p2,p3,p4
```

This function calculates *q* vectors and returns them. We use the *.subtract* from *NumPy* library for each pair the vectors, as shown below.

```
def calc_q_vectors(p1,p2,p3,p4):
    """Function to calculate q vectors"""
    import numpy as np

    # Calculate coordinates for vectors q1, q2 and q3
    q1 = np.subtract(p2,p1) # b - a
    q2 = np.subtract(p3,p2) # c - b
    q3 = np.subtract(p4,p3) # d - c

    return q1,q2,q3
```

Here we calculate the cross vectors ($\mathbf{q}_1$x$\mathbf{q}_2$ and $\mathbf{q}_2$x$\mathbf{q}_3$), using *.cross* from *NumPy* library, as shown below.

```
def calc_cross_vectors(q1,q2,q3):
    """Function to calculate cross vectors"""
    import numpy as np

    # Calculate cross vectors
    q1_x_q2 = np.cross(q1,q2)
    q2_x_q3 = np.cross(q2,q3)

    return q1_x_q2, q2_x_q3
```

Now we calculate normal vectors to planes, using *.dot* and *.sqrt* from *NumPy* library, as shown below.

```
def calc_nornals(q1_x_q2,q2_x_q3):
    """Function to calculate normal vectors to planes"""
    import numpy as np

    # Calculate normal vectors
    n1 = q1_x_q2/np.sqrt(np.dot(q1_x_q2,q1_x_q2))
    n2 = q2_x_q3/np.sqrt(np.dot(q2_x_q3,q2_x_q3))

    return n1,n2
```

This function calculates orthogonal unit vectors, using *.cross*, *.dot,* and *.sqrt* from *NumPy* library, as shown below.

```
def calc_orthogonal_unit_vectors(n2,q2):
    """Function to calculate orthogonal unit vectors"""
    import numpy as np

    # Calculate unit vectors
    u1 = n2
    u3 = q2/(np.sqrt(np.dot(q2,q2)))
    u2 = np.cross(u3,u1)

    return u1,u2,u3
```

Finally, we calculate the dihedral angle using *atan2* from *math* library and the *.degree* and *.dot* from *NumPy* library, as shown below.

```
def calc_dihedral_angle(n1,u1,u2,u3):
    """Function to calculate dihedral angle"""
    import numpy as np
    import math

    # Calculate cosine and sine
    cos_theta = np.dot(n1,u1)
    sin_theta = np.dot(n1,u2)

    # Calculate theta
    theta = -math.atan2(sin_theta,cos_theta)    # it is different from Fortran math.atan2(y,x)
    theta_deg = np.degrees(theta)

    # Show results
    print("theta (rad) = %8.3f"%theta)
    print("theta (deg) = %8.3f"%theta_deg)
```

To run *dihedral_angle.py,* type *python dihedral_angle.py*, as shown below.

```
C:\Users\Walter>python dihedral_angle.py
theta (rad) =  -3.142
theta (deg) =  -180.0

C:\Users\Walter>
```

-BERMAN HM, Westbrook J, Feng Z, *et al.* The Protein Data Bank. Nucleic Acids Res 2000; 28: 235-42.

-BERMAN HM, Battistuz T, Bhat TN, *et al.* The Protein Data Bank. Acta Crystallogr D Biol Crystallogr 2002; 58(Pt 6 No 1): 899-907.

-BRESSERT, Eli. **SciPy and NumPy.** Sebastopol: O'Reilly Media, Inc., 2013. 56 p.

-IDRIS, Ivan. **NumPy 1.5. An action-packed guide dor the easy-to-use, high performance, Python based free open source NumPy mathematical library using real-world examples. Beginner's Guide.** Birmingham: Packt Publishing Ltd., 2011. 212 p.

- WESTBROOK J, Feng Z, Chen L, Yang H, Berman HM. The Protein Data Bank and structural genomics. Nucleic Acids Res 2003; 31(1): 489-491.

Last update July, 6th 2016.

Colophon

This text was produced in a DELL Inspiron notebook with 6GB of memory, a 750 GB hard disk, and an Intel® Core® i5-3337U CPU @ 1.80 GHz running Windows 8.1. Text and layout were generated using PowerPoint 2013 and graphical figures shown in the slides 9 and 12 were generated by *Visual Molecular Dynamics (VMD)(http://www.ks.uiuc.edu/Research/vmd/)*. This tutorial uses Arial font.

Author

I graduated in Physics (BSc in Physics) at University of Sao Paulo (USP) in 1990. I completed a Master Degree in Applied Physics also at USP (1992), working under supervision of Prof. Yvonne P. Mascarenhas, the founder of crystallography in Brazil.
My dissertation was about X-ray crystallography applied to organometallics compounds (De Azevedo Jr. *et al*.,1995).
During my PhD I worked under supervision of Prof. Sung-Hou Kim (University of California, Berkeley. Department of Chemistry), on a split PhD program with a fellowship from Brazilian Research Council (CNPq)(1993-1996). My PhD was about the crystallographic structure of CDK2 (Cyclin-Dependent Kinase 2) (De Azevedo Jr. *et al*.,1996). In 1996, I returned to Brazil. In April 1997, I finished my PhD and moved to Sao Jose do Rio Preto (SP, Brazil) (UNESP) and worked there from 1997 to 2005. In 1997, I started the Laboratory of Biomolecular Systems-Department of Physics-UNESP - São Paulo State University. In 2005, I moved to Porto Alegre/RS (Brazil), where I am now. My current position is coordinator of the Laboratory of Computational Systems Biology at Pontifical Catholic University of Rio Grande do Sul (PUCRS). My research interests are focused on application of computer simulations to analyze protein-ligand interactions. I'm also interested in the development of biological inspired computing and machine learning algorithms. We apply these algorithms to molecular docking simulations, protein-ligand interactions and other scientific and technological problems. I published over 160 scientific papers about protein structures and computer simulation methods applied to the study of biological systems (H-index: 36). These publications have over 4000 citations. I am editor for the following journals: