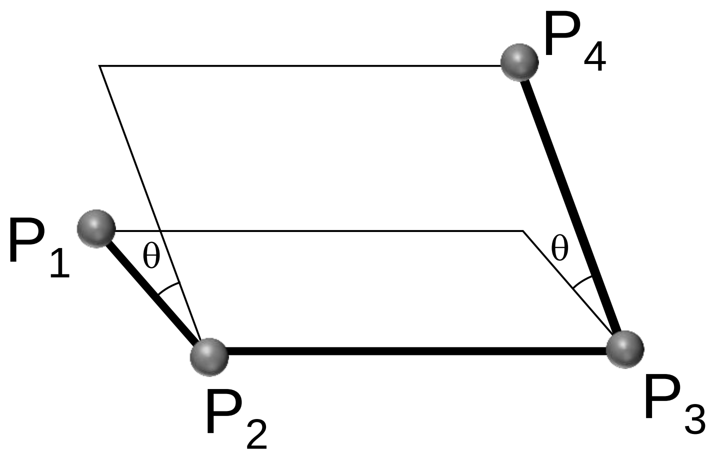
Torsion Angle in Python

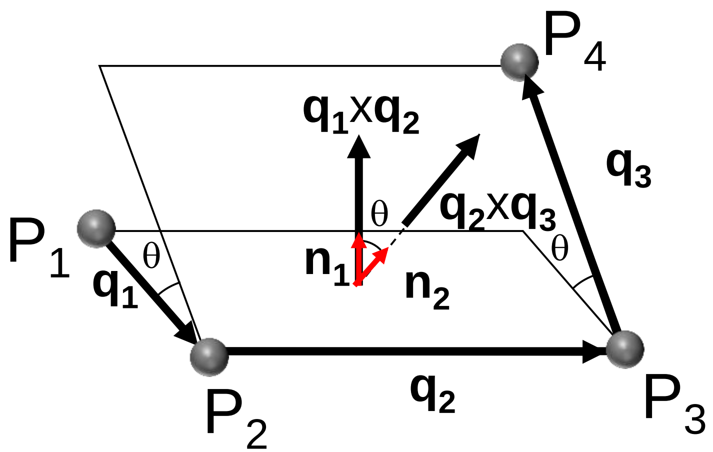
By Prof. Walter F. de Azevedo Jr.

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



To determine the torsion angle θ, we need to consider three vectors connecting points P_1, P_2, P_3 , and P_4 , named here as $\mathbf{q}_1$, $\mathbf{q}_2$, 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, 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 unit orthogonal vectors $\mathbf{u_1}$, $\mathbf{u_2}$, and $\mathbf{u_3}$ as follows:

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

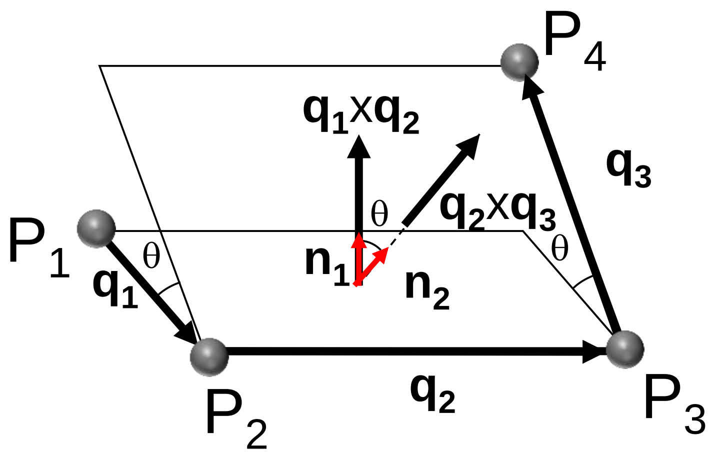
$$\mathbf{u}_3 = \frac{\mathbf{q}_2}{|\mathbf{q}_2|}$$

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

the cosine and sine are given by:

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

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



Then, torsion angle θ is as follows,

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

You have to use *atan2* function, which is available in Python, to determine the torsion angle.

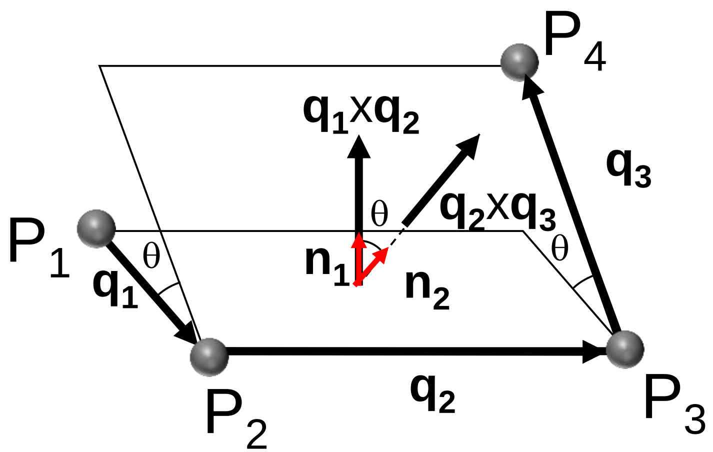
In summary, to calculate the torsion angle θ for a system with four points, we have to follow the steps shown below.

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

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

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

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



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

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

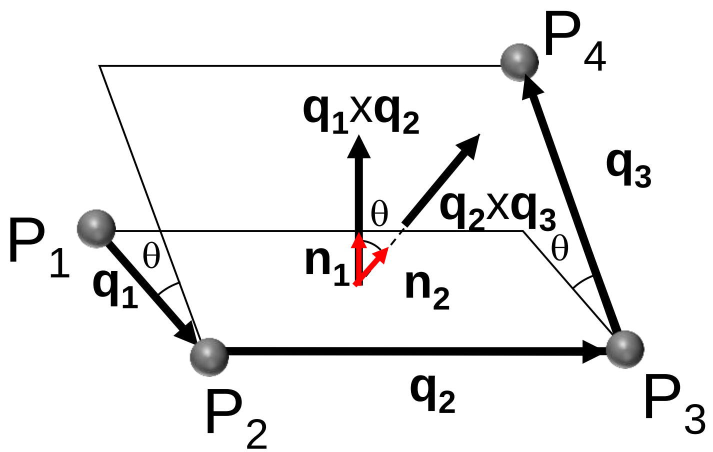
$$\boxed{\begin{array}{l}\mathbf{q_1} \times \mathbf{q_2} \\ \mathbf{q_2} \times \mathbf{q_3}\end{array}}$$

3) Calculate normal vector to planes, as follows:

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

4) Calculate unit orthogonal vectors

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



5) Calculate torsion angle θ:

$$\boxed{\begin{aligned} \cos - \theta &= \mathbf{n}_1 . \mathbf{u}_1 \\ sen - \theta &= \mathbf{n}_1 . \mathbf{u}_2 \end{aligned}} \qquad \boxed{\theta = -a\tan 2\left(\frac{\mathbf{n}_1 \cdot \mathbf{u}_1}{\mathbf{n}_1 \cdot \mathbf{u}_2}\right)}$$

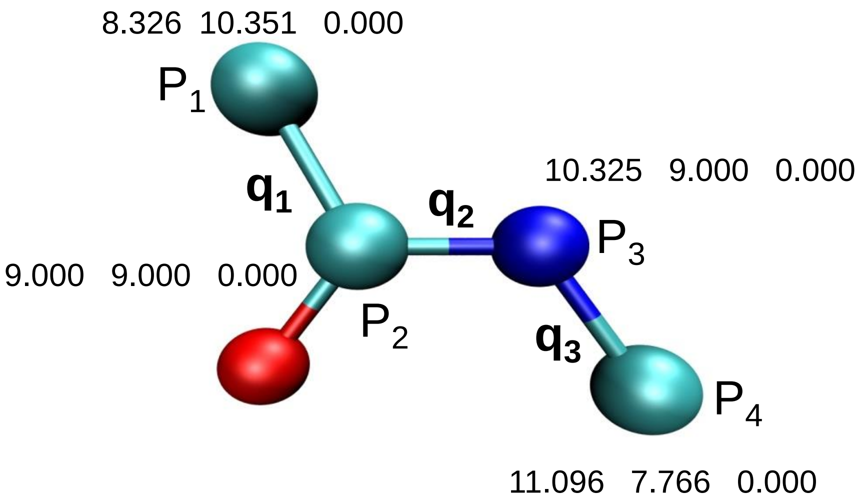
Example. Calculate the torsion angle θ between the planes defined by the points P_1, P_2, P_3 e P_4, using the coordinates indicated below.

$$\mathbf{p_1} = 8.326\mathbf{i} + 10.351\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p_2} = 9.000\mathbf{i} + 9.000\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p_3} = 10.325\mathbf{i} + 9.000\mathbf{j} + 0.000\mathbf{k}$$

$$\mathbf{p_4} = 11.096\mathbf{i} + 7.766\mathbf{j} + 0.000\mathbf{k}$$



We could think that each coordinate is an atomic coordinate, as the ones available in a PDB file.

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:

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

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

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**

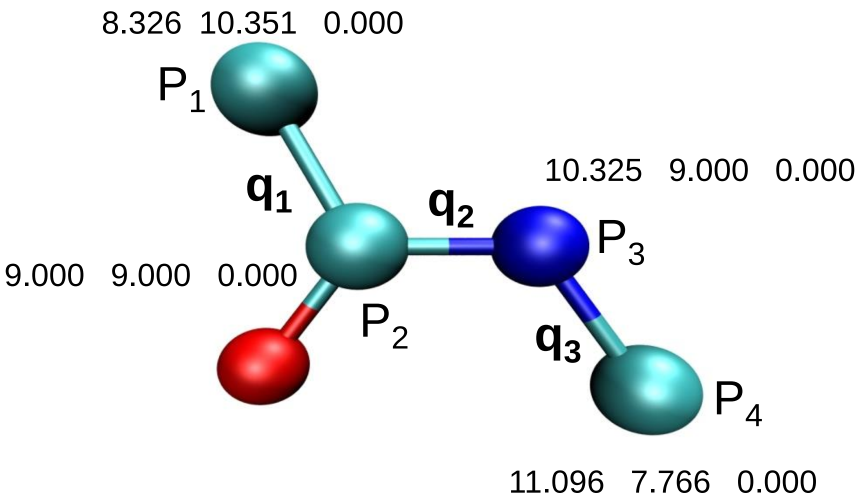
Torsion Angle (Example)

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 torsion angle

$$\cos - \theta = \mathbf{n_1}.\mathbf{u_1} = \mathbf{k} \cdot (-\mathbf{k}) = \text{-}1$$

$$sen - \theta = \mathbf{n_1}.\mathbf{u_2} = \mathbf{k} \cdot \mathbf{j} = 1$$

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

Torsion angle for system with four points

Program: *torsion_angle.py*

Abstract

Program to calculate torsion angle in degrees for a system with four points (P1,P2,P3,P4). The torsion angle is between two planes, the first defined by the points P1, P2 and P3 and the second plane by the points P2, P3 e P4. The results is shown on screen.

python
www.python.org

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_normals(q1_x_q2,q2_x_q3)*, *calc_unit_orthogonal_vectors(n2,q2)*, and *calc_torsion_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_normals(q1_x_q2,q2_x_q3) function
    n1, n2 = calc_normals(q1_x_q2,q2_x_q3)
    # Call calc_unit_orthogonal_vectors(n2,q2) function
    u1,u2,u3 = calc_unit_orthogonal_vectors(n2,q2)
    # Call calc_torsion_angle(u1,u2,u3) function
    calc_torsion_angle(n1,u1,u2,u3)
main()
```

In this function we define the coordinates for four points and return them. We use *NumPy* 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
```

Function: *calc_q_vectors()*

This function calculates q vectors and returns them. We use the *.subtract* from *NumPy* library.

```
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, 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_normal(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 unit orthogonal vectors, using *.cross*, *.dot,* and *.sqrt* from *NumPy* library, as shown below.

```
def calc_unit_orthogonal_vectors(n2,q2):
    """Function to calculate unit orthogonal 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 torsion angle using *atan2* from *math* library and the *.degree* and *.dot* from *NumPy* library, as shown below.

```
def calc_torsion_angle(n1,u1,u2,u3):
    """Function to calculate torsion 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 atan2 from fortran math.atan2(y,x)
    theta_deg = np.degrees(theta)

    # Show results
    print("theta (rad) = ",theta)
    print("theta (deg) = ",theta_deg)
```

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

```
C:\Users\Walter>python torsion_angle.py
theta (rad) =  -3.141592653589793
theta (deg) =  -180.0

C:\Users\Walter>
```

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

-DAWSON, Michael. Python Programming, for the absolute beginner. 3ed. Boston: Course Technology, 2010. 455 p.

-HETLAND, Magnus Lie. Python Algorithms. Mastering Basic Algorithms in the Python Language. Nova York: Springer Science+Business Media LLC, 2010. 316 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.

-KIUSALAAS, Jaan. Numerical Methods in Engineering with Python. 2ed. Nova York: Cambridge University Press, 2010. 422 p.

-LANDAU, Rubin H. A First Course in Scientific Computing: Symbolic, Graphic, and Numeric Modeling Using Maple, Java, Mathematica, and Fortran90. Princeton: Princeton University Press, 2005. 481p.

-LANDAU, Rubin H., PÁEZ, Manuel José, BORDEIANU, Cristian C. A Survey of Computational Physics. Introductory Computational Physics. Princeton: Princeton University Press, 2008. 658 p.

-LUTZ, Mark. Programming Python. 4ed. Sebastopol: O'Reilly Media, Inc., 2010. 1584 p.

-MODEL, Mitchell L. Bioinformatics Programming Using Python. Sebastopol: O'Reilly Media, Inc., 2011. 1584 p.

-TOSI, Sandro. Matplotlib for Python Developers. Birmingham: Packt Publishing Ltd., 2009. 293 p.

Last update February, 4th 2016.

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 were generated by *Visual Molecular Dynamics (VMD)(http://www.ks.uiuc.edu/Research/vmd/)*. This tutorial uses Arial font.

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) (http://dx.doi.org/doi:10.1107/S0108270194009868). 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-Depedent Kinase 2) (De Azevedo Jr. et al., 1996)(http://www.ncbi.nlm.nih.gov/pubmed/9552391). 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 application of 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: 33). These publications have over 3700 citations. I am regional editor for South and Central America for Current Drug Target (ISSN: 1873-5592 (Online), ISSN: 1389-4501 (Print))(http://benthamscience.com/journal/editorial-board.php?journalID=cdt#top), academic editor for Current Bioinformatics (http://benthamscience.com/journals/current-bioinformatics/editorial-board/#top) and guest editor for Current Medicinal Chemistry. Academic Profile on Google Scholar   Link to Facebook
More information at www.azevedolab.net .