01.01_newton_on_harmonic_potential_of_water.solutions

May 23, 2017

In [1]:
```
from ase.io import read
from ase import Atoms
from ase.optimize import *
from ase.calculators.mopac import  *
import math
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib notebook
```

In [2]:
```
# define global variables; in general, this is not great programming practice, but conve
global bond_0
global angle_0
global k_bond
global k_angle

bond_0  =   0.9572
k_bond  = 450.0
angle_0 = 104.52 / 180 * math.pi
k_angle =  55.0 #/180*math.pi

def Vwat(bond, angle):
    # this is a docstring, e.g. for the help() command
    ''' computes the potential energy of a water molecule '''
    global bond_0
    global angle_0
    global k_bond
    global k_angle
    Vbond  = 0.5 * k_bond  * ( bond -  bond_0)**2
    Vangle = 0.5 * k_angle * (angle - angle_0)**2
    V = 2 * Vbond + Vangle
    return V

def gbond(bond):
    # FIX ME: What does this function do?
    ''' computes the gradient of the bond potential '''
    global k_bond
```

```
    global bond_0
    return 2 * k_bond * (bond - bond_0)

def gangle(angle):
    # FIX ME: What does this function do?
    ''' computes the gradient of the angle potential '''
    global k_angle
    global angle_0
    return k_angle * (angle - angle_0)

def gwat(bond, angle):
    # FIX ME: What does this function do?
    ''' computes the gradient of the potential energy '''
    g = np.zeros(2)
    g = (gbond(bond), gangle(angle))
    return g

def Hwat(bond, angle):
    ''' computes the Hessian of the potential energy '''
    # FIX ME: Compute the Hesse matrix in terms of k_bond and k_angle
    Hbondbond   = 2 * k_bond
    Hbondangle  = 0.0
    Hangleangle = k_angle
    Hanglebond  = 0.0
    H = [[Hbondbond, Hbondangle],[Hanglebond, Hangleangle]]
    return H

def newton(f,df,a,b,delta):
    ''' simple Newton solver '''
    if delta <= 0:
        print("error3: delta is non-positive")
        return
    n    = 0
    xold = b
    yold = a
    y    = yold
    x    = xold
    Hinv = (1/(2*k_bond), 1/k_angle)
    step = 1

    Fold = f(xold,yold)
    Fnew = Fold
    gnew = df(x,y)
    while((abs(gnew[0]) > delta) or (abs(gnew[1] > delta))):
        # print "energy", Fnew, "gradient", gnew
        # print n
        xold = x
        yold = y
```

```
                Fold = Fnew
                # FIX ME: Help! In each of the next two lines, there a bug!
                x = xold - df(xold,yold)[0] * Hinv[0] * step
                y = yold - df(xold,yold)[1] * Hinv[1] * step
                n = n + 1
                Fnew = f(x,y)
                gnew = df(x,y)
                if (n > 100):
                    return x,y
                print("converged in ", n , "steps")
                return x, y

        help(newton)
```

```
Help on function newton in module __main__:

newton(f, df, a, b, delta)
    simple Newton solver
```

In [3]:
```
        #
        # Plot the 2D-potential energy
        #
        wat_bond  = np.zeros(10)
        wat_angle = np.zeros(10)

        # set up initial water coordinates as angle and bond length
        for i in range(len(wat_bond)):
            wat_angle[i] = 1.1 + i * 0.2
            wat_bond[i]  = 0.5 + i * 0.1

        E = np.zeros((10,10))
        for a in range(len(wat_angle)):
            for b in range(len(wat_bond)):
                E[a, b] = Vwat(wat_bond[b], wat_angle[a])

        xp = wat_bond
        yp = wat_angle
        xp, yp = np.meshgrid(xp,yp)

        fig = Axes3D(plt.figure())
        fig.plot_wireframe(xp, yp, np.vectorize(Vwat)(xp, yp))
        plt.xlabel('bond')
        plt.ylabel('angle')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Out[3]: <matplotlib.text.Text at 0x11b31bb50>

In [4]:
```
print "testing simple 2d-energy function for water"
print "equilibrium bond", bond_0
print "equilibrium angle", angle_0

for model in ("h2o.xyz", "h2o_linear.xyz", "h2o_90.xyz"):
    infile  = "input/" + model
    water   = read(infile, format="xyz")
    molecule= Atoms(water)
    crds    = molecule.get_positions()
    print
    print "////////////////////"
    print "model", model
    print "coordinates"
    print crds

    bondHH  = molecule.get_distance(1,2)
    bondOH  = molecule.get_distance(0,2)
    angleHOH= molecule.get_angle([1,0,2])

    print "bond OH",  bondOH, "angle HOH", angleHOH, "angle in degree", angleHOH/math.pi
    ener=Vwat(bondOH, angleHOH)
    print "initial energy: " ,ener
    print "initial gradient:", gwat(bondOH, angleHOH)

    x,y = newton(Vwat, gwat, angleHOH, bondOH, 0.01)

    print "final position", x,y, "energy", Vwat(x,y), "gradient", gwat(x,y)
```

```
testing simple 2d-energy function for water
equilibrium bond 0.9572
equilibrium angle 1.82421813418

////////////////////
model h2o.xyz
coordinates
[[ 0.        0.        0.      ]
 [ 0.        0.        0.947   ]
 [ 0.892841  0.       -0.315663]]
bond OH 0.946999567503 angle HOH 1.91062929097 angle in degree 109.470994587
initial energy:  0.252161390898
initial gradient: (-9.1803892477129505, 4.7526136234232972)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)
```

```
////////////////////
model h2o_linear.xyz
coordinates
[[ 0.     0.     0.   ]
 [ 0.     0.    -0.958]
 [ 0.     0.     0.958]]
bond OH 0.958 angle HOH 3.14159265359 angle in degree 180.0
initial energy:  47.7258676704
initial gradient: (0.7199999999999207, 72.455598567292597)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)

////////////////////
model h2o_90.xyz
coordinates
[[ 0.     0.     0.   ]
 [ 0.     0.     0.958]
 [ 0.     0.958  0.   ]]
bond OH 0.958 angle HOH 1.57079632679 angle in degree 90.0
initial energy:  1.76640984267
initial gradient: (0.7199999999999207, -13.938199406426715)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)
```

In [12]: *# for your reference, this is how the final output should look like*

```
testing simple 2d-energy function for water
equilibrium bond 0.9572
equilibrium angle 1.82421813418

////////////////////
model h2o.xyz
coordinates
[[ 0.        0.        0.      ]
 [ 0.        0.        0.947   ]
 [ 0.892841  0.       -0.315663]]
bond OH 0.946999567503 angle HOH 1.91062929097 angle in degree 109.470994587
initial energy:  0.252161390898
initial gradient: (-9.1803892477129505, 4.7526136234232972)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)

////////////////////
model h2o_linear.xyz
coordinates
[[ 0.     0.     0.   ]
```

```
 [ 0.     0.    -0.958]
 [ 0.     0.     0.958]]
bond OH 0.958 angle HOH 3.14159265359 angle in degree 180.0
initial energy:  47.7258676704
initial gradient: (0.7199999999999207, 72.455598567292597)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)

////////////////////
model h2o_90.xyz
coordinates
[[ 0.     0.     0.   ]
 [ 0.     0.     0.958]
 [ 0.     0.958  0.   ]]
bond OH 0.958 angle HOH 1.57079632679 angle in degree 90.0
initial energy:  1.76640984267
initial gradient: (0.7199999999999207, -13.938199406426715)
('converged in ', 1, 'steps')
final position 0.9572 1.82421813418 energy 0.0 gradient (0.0, 0.0)
```

In [ ]: