Report to: **Millsian Software**
Subject: **OH Radical**
From: **Philip Payne, Ph.D., Consultant, InterBiotics LLC**
Date: **Original April 16, 2010/ Second Revision August 14, 2010**

This is the first in a sequence of reports that discuss the experimental validity and theoretical foundations for a new model of chemical bonding that Dr. Randell Mills has presented in his monograph, The Grand Unified Theory of Classical Physics, 2009 Edition, hereafter referenced as GUTCP. These reports have two aims. The first is comparison of calculated molecular properties with publicly available experimental data. The second is to describe the new concepts of chemical bonding in language that will be broadly intelligible to the computational chemistry community.

The hydroxyl radical, which is formed by reaction of hydrogen and oxygen atoms, is one of the simplest examples of chemical bonding between unlike atoms. Formation of this radical also entails recoupling of both spin and orbital angular momentum for the oxygen electrons. So although it is a simple example, the system is complex enough to test diverse components of the GUTCP model for molecular electronic structure. The work discussed in this report confirms that the GUTCP correctly calculates the experimental dissociation energy of the hydroxyl radical at 0 degrees Kelvin. Key results are shown in Table 1.

Table 1. Dissociation Energy of the Hydroxyl Radical

D_e (EV)	Error Bar (EV)	Kelvin Temperature	Source
4.4557	0.0030	298	Y.R. Luo, Comprehensive Handbook of Chemical Bond Energies, CRC Press, 2007. Cited at NIST web site.
4.4172	0.0030	0	Apply ideal gas translational energy correction: - 3RT/2 = -0.0385 EV
4.4104	-	0	GUTCP Volume 2, page 460, Eqn. 13.156
4.4104	-	0	Recomputed for this report

The dissociation energy of the hydroxyl radical was calculated using only fundamental physical constants, the experimental first ionization potentials of oxygen and hydrogen, and the experimental vibrational frequency of the OH radical. Note that the vibrational frequency is itself well-predicted by the Millsian algorithms, but the experimental value was used to ensure greater accuracy of the Doppler term in the bond dissociation energy prediction. The confirmation of the experimental dissociation energy in this first example suggests that the underlying physical model of chemical bonding in GUTCP is correct. Note that all parts of the calculation were verified by independent reprogramming of relevant GUTCP equations. The computer codes and program output are supplied as appendices to this report.

B. Verification of Dissociation Energy

GUTCP interprets the chemical bond energy of the OH radical as the sum of several terms. The present report does not rederive any of the fundamental results of GUTCP corresponding to equations (1) through (7). Instead we will show they are numerically correct with respect to the OH radical.

(a) **MOLECULAR ORBITAL (MO) ENERGY**. The current density vectors (orbits) of the bonding electron pair lie on a surface that is a composite of the oxygen 2Patomic orbital (AO) and a hydrogen-molecule like ellipsoid of revolution (see GUTCP figure 13.3). The surface is a linear combination of a spheroid and a prolate spheroid. The parameterization of the prolate spheroid is adjusted to match its energy to that of the oxygen spheroid. The prolate spheroidal energy E_{MO} is then given as equation (1).

(1)
$$\begin{aligned} E_{MO} &= V_e + T + V_m + V_p + E_{2P} \\ &= -\left(\frac{e^2}{8\pi\varepsilon_0}\right)\frac{1}{f}\left[\left(\frac{3}{2}-\frac{3a_0}{8a}\right)\ln\left(\frac{a+f}{a-f}\right)-1\right]+E_{2P} \end{aligned}$$

The symbols *a* and f respectively denote the semi major axis and focal distance of the prolate spheroidal portion of the bonding molecular orbital. Equation (1) is built from the leading terms in GUTCP equations (13.148) and (13.149) with the energy matched to the oxygen 2P shell energy. Numerical values of the semimajor axis and focal distance are verified below in section (C).

(b) **MAGNETIC COUPLING.** Chemical bonding entails recoupling of electron spins. The GUTCP analysis of electronic structure for the helium atom concluded that the atom is stabilized by a spin-pairing term as the magnetic moment of one electron of the pair interacts with the magnetic field induced by the current density of the other electron. For a spherical orbitsphere occupied by two electrons at radius r, GUTCP equation 7.46 describes the magnetic coupling energy.

(2) $$E_{MAG} = \frac{8\pi\mu_0\mu_B^2}{r^3}$$

The native configuration of the oxygen atom is similarly stabilized by interaction of two unpaired electrons and this stabilization is lost when one of these electrons spin-pairs with the hydrogen electron; the compensatory magnetic stabilization of the bonding MO is already built into equation (1) as part of V_m. Therefore bond formation incurs a magnetic reorganization penalty; we substitute the oxygen 2P shell radius $r=a_0$ into equation (2) to evaluate this penalty for the OH bond.

(3) $$E_{MAG} = \frac{8\pi\mu_0\mu_B^2}{a_0^3}$$

(c) **DOPPLER CORRECTION FOR ELECTRONS**. Doppler coupling of electronic motion to the radiation field supporting an MO is a unique concept in GUTCP. Equation (11.187) is reproduced below as equation (4), where $\overline{E}_k$ is the average kinetic energy of the electron current in a MO.

(4) $$E_{DOP} = (E_{MO}+E_{2P})\sqrt{\frac{2\overline{E}_k}{m_e c^2}} = (E_{MO}+E_{2P})\sqrt{\frac{\overline{v}^2}{c^2}}$$

The right hand side of (4) uses the identity $\overline{E}_k = \frac{1}{2}m_e\overline{v}^2$, the standard definition of particle kinetic energy, except it is averaged over the MO. This is a classical equation derived using known formulas for photon recoil [T. C. Gibb, *Principles of Mössbauer Spectroscopy,* Wiley, (1976), pp. 1-5.] and oscillation in the transition state for a reentrant orbit [G. R. Fowles, *Analytical Mechanics*, Third Edition, Holt, Rinehart, and Winston, New York, (1977), pp. 161-164.]. By combining the Doppler term with the other terms we obtain equation (5).

(5) $$E_{MO}+E_{2P}+E_{DOP} = (E_{MO}+E_{2P})\left(1+\sqrt{\frac{\overline{v}^2}{c^2}}\right)$$

The GUTCP evaluates $\bar{E}_k$ by solving for the electron reentrant oscillatory frequency in the transition state to form the MO. Combining GUTCP Equations (13.142 and 13.143) yields:

$$\bar{E}_k = \hbar\sqrt{\frac{0.75e^2}{4\pi\varepsilon_0 m_e b^3}} \tag{6}$$

The symbol b denotes the semi minor axis of the bonding MO for the OH radical.

(d) **NUCLEAR VIBRATION**. The nuclei also undergo simple harmonic oscillation in the transition state at their corresponding frequency due to the reentrant orbit derived classically. The corresponding equation 13.145 contains a term $\frac{1}{2}\hbar\omega_{vib}$. This is formally identical to the zero point energy of a quantum mechanical harmonic oscillator, but the latter is not physical and is disproved experimentally.

(e) **BOND ENERGY**. The chemical bond energy is the sum of these terms, less the energies of the bonding electrons in the outer orbitspheres of their original atoms:

$$E_{BOND} = (E_{MO} + E_{2P})\left(1 + \sqrt{\frac{2\bar{E}_k}{m_e c^2}}\right) + E_{MAG} + \frac{1}{2}\hbar\omega_{vib} - IP(H) - IP(O_{2P}) \tag{7}$$

Equations (1) through (7) were evaluated numerically using the Python source code provided in Appendix A and Python for Windows release 2.6.5. The relevant module is OH_radical.py, and the bond energy calculation is executed within a Python window using the command OH_radical.BondEnergy(). Computer output is provided in Appendix B. Table 2 compares the numerical test results with the numeric values of corresponding equations in GUTCP. Note that my value for the semi major axis (66.8967 pM) is 0.0072 pM shorter than reported in GUTCP equation 13.78 (see section C, below, for elaboration.)

C. Computation of the Semi major Axis for the OH Bonding MO

The bonding current structure for the OH MO is a linear combination of a H_2-like MO and the oxygen 2P AO. The physical reasoning used to derive properties of the bond is given on pages 451-452 of GUTCP. Of the -2e charge density of the bond, the H_2-like MO and oxygen 2P AO are combined such that the electron charge of the H_2 MO is -1.5e and that of the O2P AO is -0.5e to achieve an energy minimum while matching the H_2 MO and O2P AO energies as discussed at Eq. (13.57). The charge distribution and motion on the MO maintains a +1 central charge at each focus.

The energy balance equations 13.76 and 13.77 equate the total energy for the OH MO including the energy match to O2P to the value previously obtained for diatomic hydrogen. For the author, use of the hydrogen molecule energy was not an obvious thing to do. But the derivation is given in GUTCP Chapters 11 and 13 and such choice represents the minimum energy prolate spheroid that is energy matched to the 2P orbital In general, the MO equations are energy matched to the atomic orbitals or hybrid orbitals by using the energy of the latter and the corresponding minimum energy matching MO charge density. This report has already established that the calculation works for OH. In fact, this approach works for all functional groups treated to date (100's).

Table 2
Energy Components of OH Radical Bond Energy

Description	Recomputed Value	GUTCP Value	GUTCP Equation ref
Semi major axis a (pM)	66.8967	66.9039	13.78
Semi minor axis b (pM)	45.9908	44.9985	13.82
Focal distance f (pM) (C' in GUTCP)	48.5800	48.5826	13.79
MO Energy Terms (EV)			
V_elec	-40.93263	-40.92709	13.96
V_mag	-8.09480	-8.09284	13.99
T_elec	16.18960	16.18567	13.98
V_nuc	14.82056	14.81988	13.97
E(Oxygen 2p) Experimental IP	-13.6181	-13.6181	
ET_OH	**-31.63537**	-31.63247	13.100
E_Doppler (EV)	**-00.33753**	-00.33749	13.144
Nuclear Vibration (EV)	**00.23155**	00.23156	13.145
Magnetic recoupling (EV)	**00.11441**	00.11441	13.152
Molecular energy (EV)	-31.62694	-31.62689	none
Ionization Potential sum for H and O(2p)	-27.21650	-27.21650	13.156
Bond Energy (EV)	4.41044	4.41039	13.156

So, let's proceed with the computation of the semi major axis. The energy balance condition represented by GUTCP equations 13.76 or 13.77 determines the semi major axis 'a' of the ellipsoidal MO section. The value of 'a' is critical to the energy calculations presented above and was obtained by claimed numerical solution of (13.77). Due diligence therefore required rechecking the computed value of 'a'. A python program Solve1377.py was written to solve a reformulation of equation 13.77 by Newton-Raphson iterations, resulting in value a= 66.8967 pM, which is 0.01% smaller than the value 66.9039 pM given in GUTCP equation (13.78). The source code of this program is included in Appendix A, and computer output from the program is shown in Appendix C.

The GUTCP equation 13.77 is rearranged by regrouping some symbols. We define

(8) $\beta = 2aa_0/3$

(9) $c = 2a_0/3$

(10) $Q = (18.01726831)(8\pi\varepsilon_0)/e$

Equation 13.77 then is equivalent to equation (11).

(11) $Q = f(\beta) \equiv \frac{1}{\sqrt{\beta}}\left[\left(\frac{3}{2} - \frac{a_0^2}{4}\frac{1}{\beta}\right)\left(\ln\left(\frac{\beta + c\sqrt{\beta}}{\beta - c\sqrt{\beta}}\right)\right) - 1\right]$

Next define auxiliary functions $G(\beta)$ and $H(\beta)$:

(12) $$G(\beta)=\left(\frac{3}{2}-\frac{a_0^2}{4}\frac{1}{\beta}\right)$$

(13) $$H(\beta)=\ln\left(\frac{\beta+c\sqrt{\beta}}{\beta-c\sqrt{\beta}}\right)$$

These transform equation (11) to the simpler form shown in (14).

(14) $$Q=f(\beta)=\frac{1}{\sqrt{\beta}}\left[G(\beta)H(\beta)-1\right]$$

The root β^* can be calculated by Newton-Raphson iteration. Given an approximate root β_k, an improved estimate β_{k+1} is estimated by first order Taylor series expansion.

(15) $$f(\beta_{k+1})=Q\approx f(\beta_k)+(\beta_{k+1}-\beta_k)f'(\beta)$$

Rearranging we solve for β_{k+1}.

(16) $$\beta_{k+1}=\beta_k+\frac{Q-f(\beta_k)}{f'(\beta_k)}$$

The required derivatives are given by equations (17) through (19).

(17) $$f'(\beta)=-\frac{f(\beta)}{2\beta}+\frac{G'(\beta)H(\beta)+G(\beta)H'(\beta)}{\sqrt{\beta}}$$

(18) $$G'(\beta)=\frac{a_0^2}{4\beta^2}$$

(19) $$H'(\beta)=\frac{c}{\sqrt{\beta}\left(c^2-\beta\right)}$$

Table 3 shows that an initial estimate a=6.0e-11 was refined to 10-figure precision in 5 iterations.

Table 3.
Refinement of Major Axis for Ellipsoidal Orbit (GUTCP 13.77)

Semi major Axis a	Relative Error of a	$f(\beta(a))$	Error of f relative to Q
6.59294e-11	9.417e-02	2.97291e+10	1.880e-01
6.68781e-11	1.429e-02	2.56044e+10	2.317e-02
6.68967e-11	2.770e-04	2.50355e+10	4.356e-04
6.68967e-11	1.014e-07	2.50246e+10	1.594e-07
6.68967e-11	1.323e-14	2.50246e+10	2.073e-14

Revised Report submitted August 14, 2010

Philip W. Payne

Philip W. Payne, President & Principal Scientist
InterBiotics LLC

Appendix A
Python Source Code Supporting this report

File PhysConstants.py

```
# Physical constants and conversion factors for GUTCP
# Written by Philip Payne under contract to Blacklight Power/ Millsian
Software
# April 1, 2010/ Revised April 16
#
# define a dictionary of universal constants - UNVCS
#!/usr/bin/python
class CNST_UNV:
    def __init__(self):
       self.setvalues()

    def setvalues(self):
       # provide values of universal constants in SI units
       self.val_pi       = 3.1415926535       # value of pi
       self.val_planck   = 6.62606896e-34     # Planck constant joule-sec
       self.val_hbar     = 1.0545716e-34      # Planck/(2*pi)
       self.val_eCoul      =  1.602176487e-19      #  elementary  charge  in
                                                Coulombs
       self.val_elecmass =  9.10938215e-31    # electron mass in kg
       self.val_BohrRad = 5.2917720859e-11    # Bohr radius in meters
       self.val_Avogadro = 6.02214879e23      # molecules per mole
       self.val_Boltzmann = 1.3806503e-23     # Boltzmann constant
                                                J/molecule-deg
       self.val_Lightspeed = 2.99792458e8     # speed of light in meter/sec
       self.val_eps0    = 8.854187817e-12     # electric permittivity
                                                constant
       self.val_eps04pi = 1.112650056e-10     # 4pi times eps0
       self.val_mu0 = 1.2566370614e-06        # vacuum permeability

    def pi(self):
       return self.val_pi
    def h(self):
       return self.val_planck
    def hbar(self):
       return self.val_hbar
    def eps0(self):
       return self.val_eps0
    def e4pi(self):
       return self.val_eps04pi
    def mu0(self):
          return  self.val_mu0
    def cLight(self):
       return self.val_Lightspeed
    def eC(self):
       return self.val_eCoul
    def a0(self):
       return self.val_BohrRad
    def eM(self):
       return self.val_elecmass
    def Boltz(self):
        return self.val_Boltzmann
    def BohrMagneton(self):
        uB=(self.val_eCoul*self.val_hbar)/(2.0*self.val_elecmass)
        return uB
    def kJtoEV(self,kJperMol):
         print "kJperMol = %12.5e" % kJperMol
         return 1000*kJperMol/(self.val_Avogadro*self.val_eCoul)
    def EVtokJ(self, eVolt):
         return (eVolt*self.val_eCoul*self.val_Avogadro/1000.0)
```

File OH_radical.py

```
#!/usr/bin/python
# Check OH radical dissociation energy
# Based in GUTCP Vol 2 pp 450-460
# by Philip Payne
# for BlackLight Power/ Millsian
# April 3, 2010/ Revised April 15, 2010 to match report text
# ----------------------------------------
import PhysConstants
from math import *
CS=PhysConstants.CNST_UNV()  # universal contants
D_expt = 4.4172
# expt dissn energy in EV at 0 degrees (Y-R. Luo, CRC handbook 2007)
a = 6.6896677315e-11 # from soln of equation 13.77)
IP_Oxy = 13.6181 #EV  (from NIST PhysRefData)
IP_Hyd = 13.5984 #EV   (from NIST PhysRefData)
wvib = 3735.21  # cm-1 OH frequency from Herzberg

def MOShape(a):
     a0val=CS.a0()
     focus = sqrt(2.0*a*a0val/3.0)
     baxis = sqrt(a*(a-(2.0/3.0)*a0val))
     return (baxis,focus)

def Doppler(E_tot_EV,b):
   # computes Doppler correction to MO energy
   # see GUTCP page 459
   # ET_OH is electronic + nuclear energy of bonding orbitoid components
   # b is ellipsoidal semiminor axis
   # we is estimate for nuclear vibration frequency (cm-1)

   ecoul=CS.eC()
   emass=CS.eM()
   c = CS.cLight()
   # The efreq formula is specific to OH
   efreq=sqrt(0.75*ecoul*ecoul/(CS.e4pi()*b*b*b*emass))
   Ek=CS.hbar()*efreq
   Doppler_EV = E_tot_EV*sqrt(2.0*Ek/(emass*c*c))
   return Doppler_EV

def BondEnergy():
    # Purpose: predict OH radical bond energy
    # Standard constants
```

```
c = CS.cLight()
ecoul=CS.eC()   # electron charge
a0 = CS.a0()   # Bohr radius
picoM = 1.0e12     # meter to picometer conversion
EightPiEps0 = 2.0*CS.e4pi()

# solve elliptic MO shape
(b,f)= MOShape(a)   # ellipse focus f is called c' in GUTCP
print "Bonding orbitoid ellipse parameters:"
print "    semimajor axis a %8.4f pM" %(a*picoM)
print "    semiminor axis b %8.4f pM" %(b*picoM)
print "    focal distance f %8.4f pM" %(f*picoM)
print "    OH bond length   %8.4f pM" %(2*f*picoM)
#
# ----------------------------------
# Solve for MO energy
# -------------------------------------
# elliptic energy terms. (report eqn 1)
Omega = 0.75   #  occupancy factor. See GUTCP page 451
prefactor = ecoul*ecoul/EightPiEps0
if (fabs(f/a) > 0.001):
   logterm = log((a+f)/(a-f))
else:
   # safety valve for singularity
   logterm = 2.0*f/a
   # linear approx to natural log.  rough estimate - could improve

V_elec = -2.0*Omega*prefactor*logterm/f           # GUTCP 13.96
V_mag = -0.5*Omega*prefactor*(a0/a)*logterm/f     # GUTCP 13.99
T_elec = Omega*prefactor*(a0/a)*logterm/f         # GUTCP  13.98
V_nuc = prefactor/f                               # GUTCP 13.97 called
                                                   it Vp
E_mo_Joule = (V_elec+V_mag+T_elec) +V_nuc
E_mo_EV = E_mo_Joule/ecoul
print "\nMO Energy terms"
print "V_elec(EV) = %10.5f" % (V_elec/ecoul)
print "V_mag(EV)  = %10.5f" % (V_mag/ecoul)
print "T_elec(EV) = %10.5f" % (T_elec/ecoul)
print "V_nuc(EV)  = %10.5f" % (V_nuc/ecoul)
print "E_mo(EV)   = %10.5f" % E_mo_EV

# ------------------------------------
# calculate electronic Doppler term
# ------------------------------------
ET_OH = (E_mo_EV -IP_Oxy)
E_Doppler_EV = Doppler(ET_OH, b) # GUTCP 13.146 in EV
print "E_Doppler(EV) = %10.5f" %E_Doppler_EV

freqvib = 100*c*wvib   # converts cm-1 to sec-1
vibterm_EV = 0.5*CS.h()*freqvib/CS.eC()     # GUTCP eqn 13.149
print "vibterm (EV) %10.5f" %vibterm_EV

  # ----------------------------------------
  # calculate magnetic recoupling term -
  # ----------------------------------------
```

```
shellsize = a0   # specific to neutral oxygen
muBohr = CS.BohrMagneton()
ratio=muBohr/shellsize
E_Mag_EV =8.0*CS.pi()*CS.mu0()*ratio*ratio/(shellsize*ecoul)
print "E_Magnetic (EV) = %10.5f" % E_Mag_EV

E_atoms = -(IP_Oxy+IP_Hyd)
E_molecule = ET_OH +E_Doppler_EV +E_Mag_EV +vibterm_EV
E_bond = E_molecule-E_atoms
print "D_expt=%8.4f D_model=%8.4f" % (D_expt,-E_bond)
```

File Solve1377.py

```
#!C:/Python26/python
# test code for equation 13.77 in GUTCP
# Written by PPayne for Blacklight Power/Millsian April 1-3, 2010
# revised April 16
# Dependency: PhysConstants.py
# ___________________________________________________________________________-
from PhysConstants import *
from math import *
import sys
CS=CNST_UNV()   # create an instance of the physical constants class
def funct(a):
   a0=CS.a0()
   z = sqrt(2*a*a0/3)
   ecoul=CS.eC()   # electron charge in coulombs
   c1= ecoul*ecoul/(2*CS.e4pi()*z)
   c2 = (1.5-0.375*(a0/a))
   logterm= log((a+z)/(a-z))
   lhs = c1*(c2*logterm-1.0)
   rhs = 18.01726831*ecoul
   tstratio = lhs/rhs
   print "Test GUTCP equation 13.77 page 452"
   print "Parameter a = %15.8e" % a
   print "Test ratio %12.8f expect 1.0" %tstratio
   print "lhs = %15.8e" % lhs
   print "rhs = %15.8e" % rhs

def Gterm(b_old):
   a0=CS.a0()
   return 1.5-(0.25*a0*a0/b_old)
def Hterm(b_old):
   a0=CS.a0()
   c =2.0*a0/3.0
   broot=sqrt(b_old)
   return log((b_old+c*broot)/(b_old-c*broot))

def refine(a_inp,errlevel):
  # use Newton-raphson iterations to refine an estimate of a for eqn 13.77
  # a_inp = initial guess for a, errlevel = desired relative error in final answer
  # Notation follows page 9, notebook Mills_01
  a0=CS.a0()
  c=2.0*a0/3.0
  a_new =a_inp
  IPsum = 18.01726831  # per Mills GUTCP Vol2 p452
  Q = (2*CS.e4pi()*IPsum/CS.eC()) +1.0

  # set up iterations loop to refine a. more convenient to use b = c*a then back correct
  # Rewrite 13.77 as (G*H-1)/sqrt(b) = Q, where G and H are defined above
  converge = False
  iter = 0
  while (iter < 100 and not converge):
    iter = iter+1
    a_old = a_new   # reset from prior iteration
    b_old = a_old*c
    broot =  sqrt(b_old)
    G = Gterm(b_old)
    Gdrv = 0.25*a0*a0/(b_old*b_old)
    H = Hterm(b_old)
    Hdrv = c/(broot*(c*c-b_old))
    Fb = (G*H-1.0)/broot;
    Ferror = Q-Fb   # absolute error in fit of eqn 13.77
    Rel_error = fabs(Ferror/Q)
    Fdrv = -Fb/(2.0*b_old) +(G*Hdrv +Gdrv*H)/broot
    b_new = b_old + (Ferror/Fdrv)
    a_step = (b_new-b_old)/c
    a_new = a_old +a_step
    a_err = 2.0*fabs(a_step)/(fabs(a_new)+fabs(a_old))
    print "iter %3d a=%16.5e rel_err=%10.3e Fb=%16.5e rel_err=%10.3e" %
(iter,a_new,a_err,Fb,Rel_error)
    converge =(a_err < errlevel)
  if (iter < 100):
    return (a_new,iter)
  else:
    print "Convergence failed"
    sys.exit(1)

def runtest(a_start):
   print "Initial estimate for a = %12.5e" %a_start
   (a_out,iters)=refine(a_start,1.0e-10)
   print "Refined a = %17.10e at %3d iterations" %(a_out,iters)
   if (a_out < 0.0):
     print "Aborting - negative a"
     sys.exit(1)
   funct(a_out)
```

Appendix B. Screen output from execution of Python program OH_radical

```
Python 2.6.5 (r265:79096, Mar 19 2010, 18:02:59) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.5
>>> mydir='C:\Users\Phil\My Documents\Interbiotics\Blacklight\Python'
>>> import os
>>> os.chdir(mydir)
>>> import OH_radical
>>> OH_radical.BondEnergy()

Bonding orbitoid ellipse parameters:
   semimajor axis a  66.8967 pM
   semiminor axis b  45.9908 pM
   focal distance f  48.5800 pM
   OH bond length    97.1599 pM

MO Energy terms
V_elec(EV) =  -40.93263
V_mag(EV)  =  -8.09480
T_elec(EV) =   16.18960
V_nuc(EV)  =   14.82056
E_mo(EV)   =  -18.01727
E_Doppler(EV) =  -0.33753
vibterm (EV)    0.23155
E_Magnetic (EV) =    0.11441
D_expt=  4.4172 D_model=  4.4104
>>>
```

Appendix C. Screen Output from Python Program Solve1377.py

```
Python 2.6.5 (r265:79096, Mar 19 2010, 18:02:59) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.5
>>> mydir = 'C:\Users\Phil\My Documents\Interbiotics\Blacklight\Python'
>>> import os
>>> os.chdir(mydir)
>>> import Solve1377
>>> aguess = 6.0e-11
>>> Solve1377.runtest(aguess)
Initial estimate for a =  6.00000e-11
iter   1 a=    6.59294e-11 rel_err= 9.417e-02 Fb=    2.97291e+10 rel_err= 1.880e-01
iter   2 a=    6.68781e-11 rel_err= 1.429e-02 Fb=    2.56044e+10 rel_err= 2.317e-02
iter   3 a=    6.68967e-11 rel_err= 2.770e-04 Fb=    2.50355e+10 rel_err= 4.356e-04
iter   4 a=    6.68967e-11 rel_err= 1.014e-07 Fb=    2.50246e+10 rel_err= 1.594e-07
iter   5 a=    6.68967e-11 rel_err= 1.323e-14 Fb=    2.50246e+10 rel_err= 2.073e-14
Refined a =  6.6896677315e-11 at   5 iterations
Test GUTCP equation 13.77 page 452
Parameter a =  6.68966773e-11
Test ratio   1.00000000 expect 1.0
lhs =  2.88668436e-18
rhs =  2.88668436e-18
```