potential_fit

June 25, 2018

In [52]:
```
import pybel
import os
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np

from scipy.optimize import curve_fit

%matplotlib inline
```

In [2]:
```
def mm_energy(xyzfile, ff = "mmff94"):
    """
    Takes as an input the path of a xyz-file,returns a tuple, first element
    is the energy, second element is the torsion energy.
    """
    ff = pybel._forcefields[ff]
    mol2 = list(pybel.readfile("xyz", xyzfile))[0]
    success = ff.Setup(mol2.OBMol)
    if not success:
        sys.exit("Cannot set up forcefield")
    if ff.GetUnit() != "kcal/mol":
        print("Warning! Energies in {}".format(ff.GetUnit()))
    energy = ff.Energy()
    torsion_energy = ff.E_Torsion()
    return energy, torsion_energy
```

In [3]:
```
def xyz2ff(xyzdir, ff = "mmff94", whole_energy=True):
    """
    Goes through a directory of .xyz files, returns either the whole energy or E-E_torsi
    """
    results = []
    for file in os.listdir(xyzdir):
        if not file.endswith("xyz"):
            continue
        energy, torsion_energy = mm_energy(os.path.join(xyzdir, file), ff)
        idx = [int(s) for s in file.split(".") if s.isdigit()][0]
```

```
            results.append([idx,energy, torsion_energy])
        results = np.asarray(results)
        # sort by first column, remove it
        results = results[results[:,0].argsort()][:,1:]
        if whole_energy:
            results = results[:,0]
        else:
            results = results[:,0] - results[:,1]
        energy_relative = results - np.min(results)

        return energy_relative
```

```
In [4]: res_qc = pd.read_csv("full.csv")
```

```
In [5]: mmff94 = xyz2ff("xyz_opt", whole_energy=True)
        res_qc["MMFF94"] = pd.Series(mmff94)
```

```
In [6]: #uff = xyz2ff("xyz_opt", ff="uff", whole_energy=True)
        #uff = uff*0.23901 # from some reason uff energy is in kJ/mol
        #res_qc["UFF"] = pd.Series(uff)
```

```
In [7]: for method in ['PBEh-3C', 'HF', 'B3LYP']:
            mini = np.min(res_qc[method])
            res_qc[method] = res_qc[method].map(lambda a: (a - mini)*627.509 )
```

```
In [8]: res_qc.columns
```

```
Out[8]: Index(['Angle', 'PBEh-3C', 'HF', 'B3LYP', 'MMFF94'], dtype='object')
```

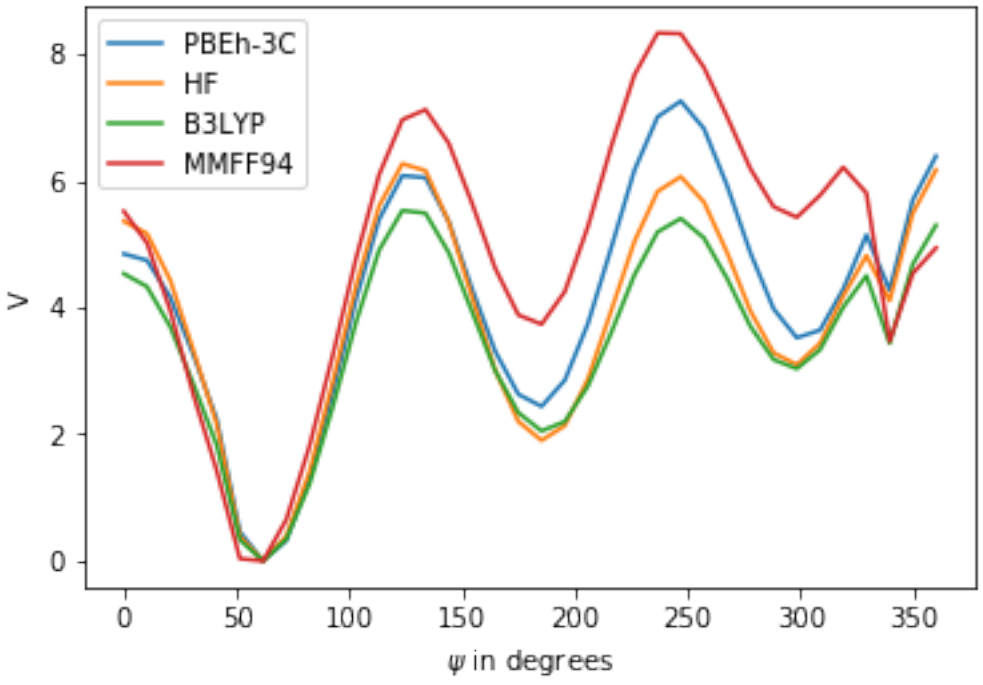
```
In [9]: res_qc
```

Out[9]:

	Angle	PBEh-3C	HF	B3LYP	MMFF94
0	0.000000	4.846503	5.361879	4.533997	5.518483
1	10.285714	4.743667	5.165312	4.328307	5.016390
2	20.571429	4.162681	4.432173	3.696817	3.953678
3	30.857143	3.247767	3.317630	2.789149	2.620025
4	41.142857	2.239366	2.165822	1.832995	1.429641
5	51.428571	0.459180	0.391077	0.329759	0.035050
6	61.714286	0.000000	0.000000	0.000000	0.000000
7	72.000000	0.320588	0.393004	0.343115	0.652476
8	82.285714	1.309561	1.404768	1.209888	1.813415
9	92.571429	2.656126	2.821450	2.427350	3.245083
10	102.857143	4.120224	4.330431	3.762273	4.767402
11	113.142857	5.391551	5.605192	4.905922	6.096223
12	123.428571	6.081692	6.269445	5.531165	6.964298
13	133.714286	6.049513	6.160569	5.489383	7.123833
14	144.000000	5.358048	5.349637	4.867591	6.592998
15	154.285714	4.299278	4.172083	3.940268	5.650908
16	164.571429	3.310794	3.013108	3.004709	4.621460

17	174.857143	2.629696	2.200221	2.340105	3.881962
18	185.142857	2.438745	1.901775	2.051075	3.736354
19	195.428571	2.850359	2.134808	2.199272	4.249829
20	205.714286	3.747057	2.872651	2.759667	5.280141
21	216.000000	4.938333	3.947685	3.599933	6.539115
22	226.285714	6.164466	5.040030	4.511556	7.669766
23	236.571429	7.009671	5.831444	5.191982	8.332103
24	246.857143	7.258466	6.065588	5.405477	8.322727
25	257.142857	6.823765	5.664750	5.097209	7.788116
26	267.428571	5.920654	4.864566	4.456052	7.008593
27	277.714286	4.864776	3.946432	3.701284	6.190482
28	288.000000	3.977974	3.283678	3.177717	5.590689
29	298.285714	3.522057	3.097435	3.034308	5.424022
30	308.571429	3.642709	3.435478	3.333949	5.769926
31	318.857143	4.289513	4.179498	4.004666	6.213436
32	329.142857	5.144218	4.820080	4.500930	5.807205
33	339.428571	4.274384	4.106911	3.435185	3.465107
34	349.714286	5.699269	5.491204	4.684741	4.539134
35	360.000000	6.390476	6.173026	5.298725	4.940716

In [53]:
```
plt.figure()
for i in ['PBEh-3C', 'HF', 'B3LYP', 'MMFF94']:
    plt.plot(res_qc["Angle"], res_qc[i], label=i)
plt.legend()
plt.xlabel(r'$\psi$ in degrees')
plt.ylabel(r'V')
plt.savefig("scan.pdf")
```



In [29]:
```
def fit(V, phi):
    psi = phi - 180
    psi_rad = np.deg2rad(psi)
    V -= np.min(V)
    C = np.polyfit(np.cos(psi_rad), V , deg=5)
    return C
```

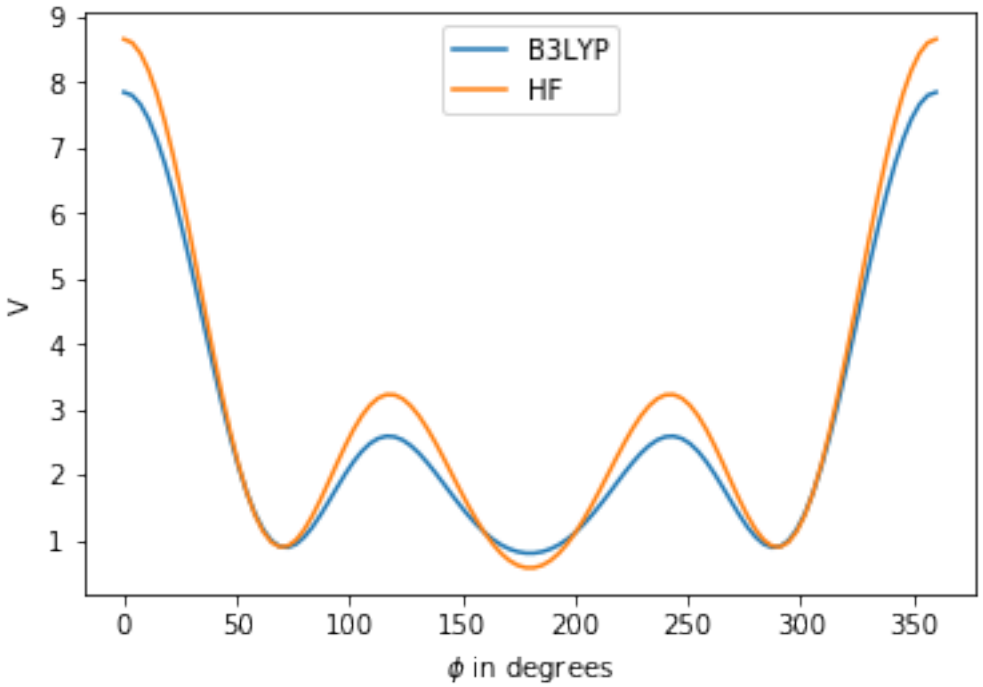
In [30]:
```
def V_RB(phi, C_vec):
    """
    Returns the potential (in kcal/mol) calculated using the
    Ryckaert-Belleman formula.

    Input:
        phi: Angle in degrees
    Output:
        Potential (in kcal/mol)
    """
    psi = phi - 180
    psi_rad = np.deg2rad(psi)
    cos = np.cos(psi_rad)
    p = np.poly1d(C_vec)
    V = p(cos)
    return V
```

In [39]:
```
def plot_RB(C, label):
    phi = np.linspace(0,360,100)
    plt.plot(phi, V_RB(phi, C), label=label)
    plt.xlabel(r'$\phi$ in degrees')
    plt.ylabel(r'V')
```

In [37]:
```
c_dict = {}
md_no_torsion = xyz2ff("xyz_opt", whole_energy=False)
for scf in ["B3LYP", "HF"]:
    V = res_qc[scf] - md_no_torsion
    C = fit(V, res_qc["Angle"])
    c_dict[scf] = C
```

In [54]:
```
for key in c_dict.keys():
    plot_RB(c_dict[key], key)
plt.legend()
plt.savefig("v_rb.pdf")
```




```
In [43]: c_df = pd.DataFrame(c_dict)

In [51]: print(c_df)

      B3LYP        HF
0  1.563306  1.486257
1  1.668620  1.575792
2 -8.396610 -9.882902
3  1.109764  1.209000
4  3.310302  4.355467
5  1.543540  1.830176


In [49]: c_df.to_csv("fits.csv")

In [ ]:
```