Electronic Supplementary Material (ESI) for Physical Chemistry Chemical Physics.

Supplementary materials to the paper
Atomic charges for conformationally rich molecules obtained through a modified principal component regression

Tymofii Yu. Nikolaienko, Leonid A. Bulavin

The code of the Python program developed as a part of the study:

```
In [1]: %reset -f
```

```
In [2]: import os
        import zipfile
        import re
        import numpy as np
        import sys
        import json

        from matplotlib import pyplot as plt
        %matplotlib inline
```

```
In [3]: molecules = {'ade': ('ade.json', 36, 19),    # ( json_filename, nAtoms, nIndependentAtoms )
                     'thy': ('thy.json', 36, 19),
                     'gua': ('gua.json', 37, 19),
                     'cyt': ('cyt.json', 34, 19) }

        common_names = {1: "HO3'", 2: "O3'", 3: "C3'", 4: "C4'", 5: "H3'", 6: "C2'", 7: "C1'",
                                8: "O4'", 9: "C5'", 10: "H1'", 11: "H2'2", 12: "H2'1", 13: "H4'",
                               14: "O5'", 15: "H5'1", 16: "H5'2", 17: "P"}
        atom_names = {'ade': dict(common_names.items() +
                                  {18: "N9", 19: "C4", 20: "C8", 21: "N7", 22: "C5", 23: "N3",
                                   24: "C2", 25: "N1", 26: "C6", 27: "N6", 28: "H2", 29: "H62",
                                   30: "H61", 31: "H8",
                                   32: "OP", 33: "OP1", 34: "HP1", 35: "OP2", 36: "HP2"}.items()),
                      'thy': dict(common_names.items() +
                                  {18: "N1", 19: "C2", 20: "C6", 21: "C5", 22: "C4", 23: "N3",
                                   24: "O2", 25: "H6", 26: "C7", 27: "O4", 28: "H3", 29: "H73",
                                   30: "H72", 31: "H71",
                                   32: "OP", 33: "OP1", 34: "HP1", 35: "OP2", 36: "HP2"}.items()),
                      'gua': dict(common_names.items() +
                                  {18: "N9", 19: "C4", 20: "C8", 21: "N7", 22: "C5", 23: "N3",
                                   24: "C2", 25: "N1", 26: "C6", 27: "O6", 28: "H1", 29: "N2",
                                   30: "H21", 31: "H22", 32: "H8",
                                   33: "OP", 34: "OP1", 35: "HP1", 36: "OP2", 37: "HP2"}.items()),
                      'cyt': dict(common_names.items() +
                                  {18: "N1", 19: "C2", 20: "N3", 21: "C4", 22: "C5", 23: "C6",
                                   24: "H6", 25: "H5", 26: "N4", 27: "H41", 28: "H42", 29: "O2",
                                   31: "OP", 30: "OP1", 34: "HP1", 32: "OP2", 33: "HP2"}.items())
                     }
```

```
In [4]: # load the molecule data -- geometrical center shift method
```

```
Debye2eA = 0.20819434

def get_dd_charges_gcenter(data_list, apt_avg, nIndependentAtoms):
    """ Calculates a dipole-derived charges for a set of conformers stored
    in data_list  and returns X,Y,q,q_min where q_min is the formal leasy
    squares fit (without 'zero'-th mode)
    Note that the input dipole must be in charge-center coordinate origin, and it
    is automatically translated into geometry-center coordinate origin during the
    calculations.
    """
    global vecs, vals ##

    nAtoms = len(apt_avg)
    # form the X and Y matrices
    X = []
    Y = []
    q_tot = sum(apt_avg) # the total charge
    print 'Total charge: %10.5f' % q_tot

    for R, D in zip(data_list['coords'], data_list['dipole']):
        R = np.array(R)
        gCenter = np.mean( R, axis=0)

        for mu in range(3):
            rec = []

            for ri in R:
                rec.append( ri[mu] - gCenter[mu])

            X.append(rec)
            dmu = D[mu] * Debye2eA - q_tot * gCenter[mu] # converted Debye to e*A
            Y.append(dmu)
    print 'avg |component of dipole moment| =', np.mean(np.abs(Y))


    X = np.array(X)

    nConfs = len( data_list['coords'] )

    A = X.T.dot(X)
    Y = np.array(Y)
    b = X.T.dot( Y )

    vals, vecs = np.linalg.eigh(A)
    print 'Eigenvalues of A:', vals

    q = np.array( [0.0] * nAtoms )
    q_min = np.array( [0.0] * nAtoms )

    a = np.zeros(nAtoms)

    for i in range(nAtoms):

        vec = vecs[ : , i ]
        L = vals[i]

        b_dip = vec.dot(b)
```

```
        if i != 0:
            dq_dip = b_dip / L * vec
            q_min += dq_dip

        a_apt = vec.dot(apt_avg)
        dq_apt = a_apt * vec

        q += dq_apt if  i < (nAtoms - nIndependentAtoms) else dq_dip

    return X, Y, q, q_min
    #---------------------------


# Run the get_dd_charges_gcenter() for each of the nucleotides:

graph_q = { 'APT': plt.figure().gca(), 'RESP': plt.figure().gca() }
graph_evals = plt.figure().gca()
graph_evals.set_yscale("log")

# settings for plots
colors = {'ade': 'k.', 'thy': 'g.', 'gua': 'bx', 'cyt': 'rx'}
labels = {'ade': 'dAMP', 'thy': 'dTMP', 'gua': 'dGMP', 'cyt': 'dCMP'}
fit_all = {'APT': [[],[]], 'RESP': [[],[]]}
final_charges = {}

for key in sorted(molecules.keys()):
    fname, nAtoms, nIndependentAtoms = molecules[key]
    data_list = json.load(open(fname))

    print '-'*80
    nConfs = len(data_list['coords'])
    print key, '%d conformers' % nConfs
    print '-'*80

    # compute some of the 'standard' charges, averaged over all conformers
    # APT
    apt = np.array(data_list['APT']) # (num_conf, num_atoms)
    apt_avg = np.sum(apt, axis=0)/len(apt)
    #print '<apt chagres> =',apt_avg

    # RESP
    resp = np.array(data_list['RESP']) # (num_conf, num_atoms)
    resp_avg = np.sum(resp, axis=0)/len(resp)
    #print '<resp chagres> =',resp_avg

    X, Y, q, q_min = get_dd_charges_gcenter(data_list, apt_avg, nIndependentAtoms)

    idx = np.argsort(-vals)
    graph_evals.plot(range(1,len(vals)), vals[idx][:-1] / nConfs, colors[key],
                     label=labels[key], alpha=0.9)

    # estimate the fitting errors
    dy = Y - X.dot(q)
    dy_min = Y - X.dot(q_min)

    # save the results for further analysis
    final_charges[key] = {}
```

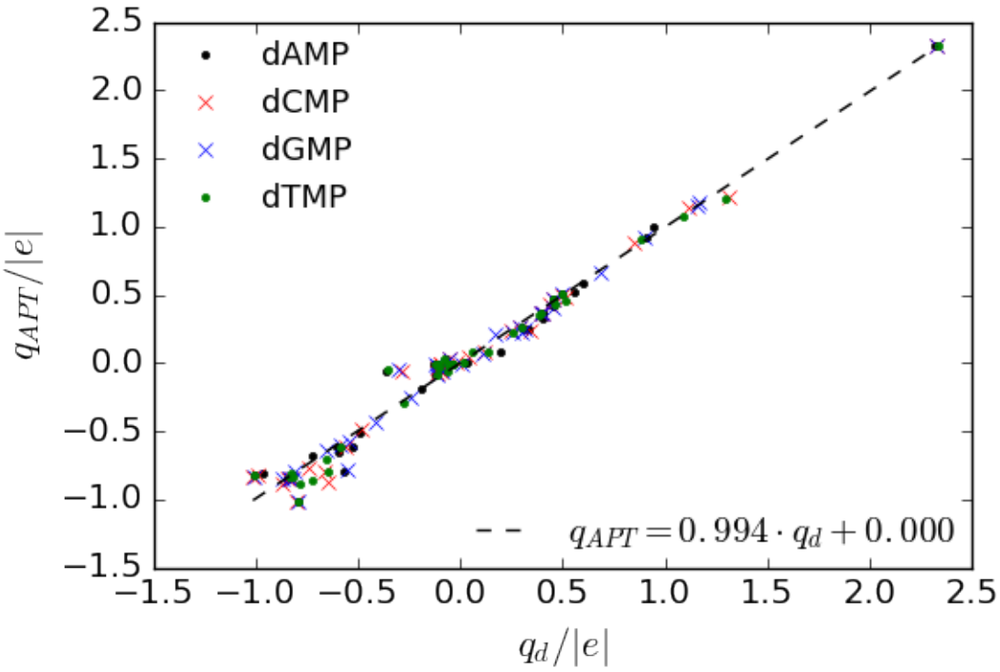
```
    final_charges[key]['dd'] = q
    final_charges[key]['apt'] = apt_avg
    final_charges[key]['resp'] = resp_avg

    print 'sum(charges) = %.5f' % sum(q)
    print 'RMS(D_mu - calculated_D_mu) =', np.sqrt(dy.dot(dy)/len(dy))
    print ' <|D_mu - calculated_D_mu|> =', sum(np.abs(dy))/len(dy)
    print 'max|D_mu - calculated_D_mu| =', np.max(np.abs(dy))

    print 'lowest possible RMS(D_mu - calculated_D_mu) =', \
        np.sqrt(dy_min.dot(dy_min)/len(dy_min))
    print 'lowest possible max|D_mu - calculated_D_mu| =', np.max(np.abs(dy_min))
    print 'lowest possible <|D_mu - calculated_D_mu|> =', sum(np.abs(dy_min))/len(dy_min)
    print


    for s, lbl in zip([apt_avg, resp_avg],['APT', 'RESP']):
        graph_q[lbl].plot(q, s, colors[key], label=labels[key] )
        fit_all[lbl][0] += list(q)
        fit_all[lbl][1] += list(s)
        graph_q[lbl].set_xlabel( '$q_{d} / |e|$', fontsize=16)
        graph_q[lbl].set_ylabel( '$q_{%s} / |e| $' % lbl, fontsize=16)

        dy = Y - X.dot(s)
        print lbl, 'sum(charges) = %.5f' % sum(s)
        print lbl, 'max|D_mu - calculated_D_mu| =', np.max(np.abs(dy))
        print lbl, '<|D_mu - calculated_D_mu|> =', sum(np.abs(dy))/len(dy)


# fit line
def lin_fit(key, line_templ, label_templ):
    X = fit_all[key][0]
    Y = fit_all[key][1]
    ax = graph_q[key]
    fit = np.polyfit(X,Y,1)
    xr = np.linspace(min(X), max(X), num=100)
    title = label_templ%(fit[0], fit[1])
    line, = ax.plot(xr, fit[0]*xr + fit[1], line_templ, label=title)

    # Create another legend for the second line.
    ax.legend(handles=[line], loc='lower right', frameon=False, fontsize=15,
              borderaxespad=0.0)

    return line, title


graph_evals.legend(loc='best', numpoints=1, frameon=False)
graph_evals.set_xlabel(r'Eigenvalue number', fontsize=13.5)
graph_evals.set_ylabel(r'$\mathrm{Eigenvalue} / N_{confs}$', fontsize=14)
# add arrows
_ = nConfs
graph_evals.annotate('19-th', xy=(19, 1e2/_), xytext=(19, 1e3/_), ha='center',
                     arrowprops =
                     {'width': 1, 'headwidth': 4, 'color': 'black', 'headlength': 5},
                    fontsize=13 )

graph_evals.annotate('20-th', xy=(20, 1e-1/_), xytext=(20, 0.5e-2/_), ha='center',
```

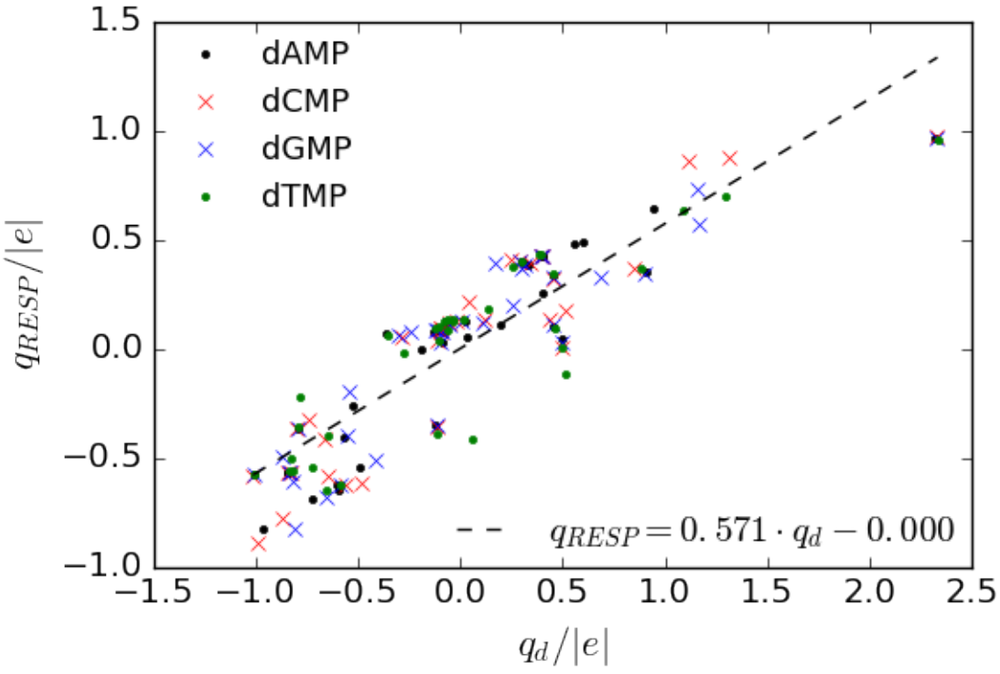
```
                            arrowprops =
                            {'width': 1, 'headwidth': 4, 'color': 'black', 'headlength': 5},
                           fontsize=13 )
        graph_evals.tick_params(labelsize=12)

        graph_evals.figure.savefig('eigenvals.png', dpi=300, bbox_inches='tight')


        for key in graph_q:
            axis = graph_q[key]
            # make legend for points
            lg1 = axis.legend(loc='upper left', numpoints=1, frameon=False,
                        fancybox=True, framealpha=0.25, ncol=1, borderaxespad=0.2,
                             fontsize=13)
            axis.add_artist(lg1)
            # add linear fit

            lin_fit(key, 'k--', '$q_{%s}'% key+' = %.3f\cdot q_{d} %+.3f$' )

            axis.tick_params(labelsize=14)

            axis.figure.savefig('fig%s.png' % key, dpi=300, bbox_inches='tight')


        plt.show()
```

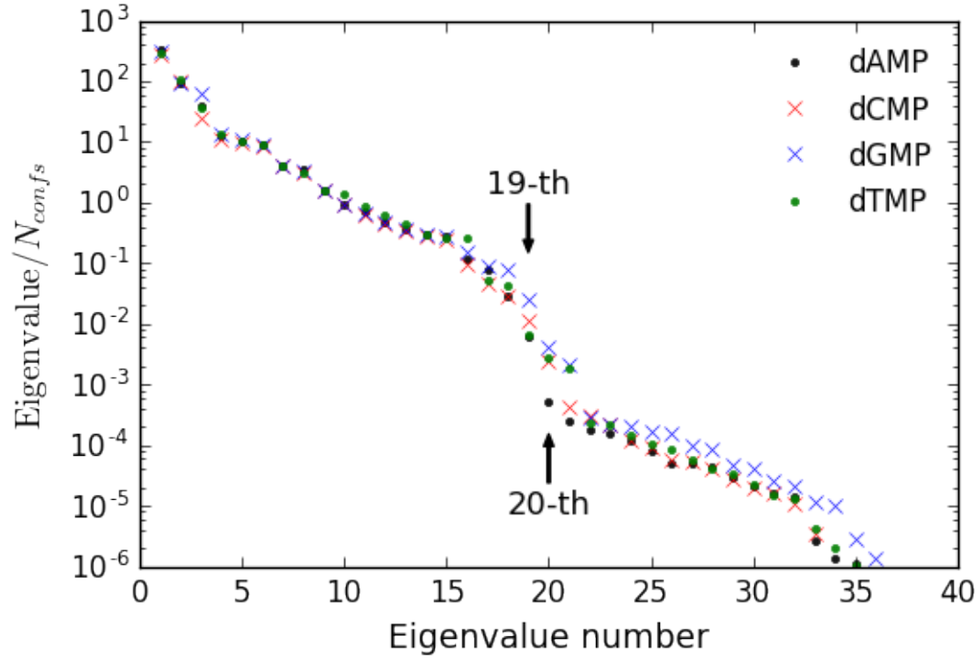
```
--------------------------------------------------------------------------------
ade 726 conformers
--------------------------------------------------------------------------------
Total charge:    0.00000
avg |component of dipole moment| = 0.40471265597
Eigenvalues of A: [ -2.40851617e-12   7.84648207e-04   9.49309342e-04   1.88408460e-03
   9.89617413e-03   1.15552228e-02   1.51488186e-02   2.08939918e-02
   3.16950675e-02   3.56759768e-02   3.68723800e-02   5.78252151e-02
   8.88378514e-02   1.14684245e-01   1.27889421e-01   1.76899550e-01
   3.79567459e-01   4.60963391e+00   2.08178735e+01   5.60218474e+01
   8.36479204e+01   1.97226534e+02   2.11938134e+02   2.64728927e+02
   3.54211145e+02   5.02435401e+02   6.63306719e+02   1.11937512e+03
   2.58274527e+03   2.95857962e+03   6.48474446e+03   7.59402614e+03
   9.27731841e+03   2.83765078e+04   6.97952304e+04   2.39302249e+05]
sum(charges) = 0.00000
RMS(D_mu - calculated_D_mu) = 0.0978088579449
 <|D_mu - calculated_D_mu|> = 0.075025726918
max|D_mu - calculated_D_mu| = 0.373181557663
lowest possible RMS(D_mu - calculated_D_mu) = 0.0706875587578
lowest possible max|D_mu - calculated_D_mu| = 0.28890248044
lowest possible <|D_mu - calculated_D_mu|> = 0.0548813171769

APT sum(charges) = 0.00000
APT max|D_mu - calculated_D_mu| = 1.67814534085
APT <|D_mu - calculated_D_mu|> = 0.423198676347
RESP sum(charges) = 0.00000
RESP max|D_mu - calculated_D_mu| = 0.555602114343
RESP <|D_mu - calculated_D_mu|> = 0.115567175206
--------------------------------------------------------------------------------
cyt 613 conformers
--------------------------------------------------------------------------------
Total charge:   -0.00000
```

```
avg |component of dipole moment| = 0.67557455898
Eigenvalues of A: [  1.27353004e-13   2.14220338e-03   6.40311129e-03   9.77936382e-03
   1.23460236e-02   1.64111247e-02   2.57186304e-02   3.36898707e-02
   3.56176349e-02   5.57991859e-02   7.18883952e-02   1.36658422e-01
   1.81055423e-01   2.60938238e-01   1.50340110e+00   6.98302036e+00
   1.78244822e+01   2.76963876e+01   5.99354146e+01   1.52139935e+02
   1.65979923e+02   2.01393381e+02   2.76116205e+02   3.84041083e+02
   5.52597101e+02   9.73163971e+02   1.88978892e+03   2.43984577e+03
   5.13314249e+03   5.98922263e+03   6.79336807e+03   1.50127963e+04
   6.19591225e+04   1.67129687e+05]
sum(charges) = -0.00000
RMS(D_mu - calculated_D_mu) = 0.112761622642
 <|D_mu - calculated_D_mu|> = 0.0886505890927
max|D_mu - calculated_D_mu| = 0.343946684109
lowest possible RMS(D_mu - calculated_D_mu) = 0.0583482032492
lowest possible max|D_mu - calculated_D_mu| = 0.245538963823
lowest possible <|D_mu - calculated_D_mu|> = 0.0449979828796

APT sum(charges) = -0.00000
APT max|D_mu - calculated_D_mu| = 1.35718006804
APT <|D_mu - calculated_D_mu|> = 0.444123955612
RESP sum(charges) = 0.00000
RESP max|D_mu - calculated_D_mu| = 0.546781091517
RESP <|D_mu - calculated_D_mu|> = 0.126493292933
--------------------------------------------------------------------------------
gua 745 conformers
--------------------------------------------------------------------------------
Total charge:   -0.00000
avg |component of dipole moment| = 0.720558297303
Eigenvalues of A: [ -5.85218116e-13   9.91451546e-04   2.13216000e-03   7.56943914e-03
   8.36809447e-03   1.58907254e-02   1.86571074e-02   3.04436343e-02
   3.53926258e-02   6.24959903e-02   7.07865536e-02   1.16449816e-01
   1.23348965e-01   1.47739940e-01   1.67329429e-01   2.11165690e-01
   1.59137802e+00   3.05786405e+00   1.86637082e+01   5.96235298e+01
   6.53085379e+01   1.11131354e+02   2.01383330e+02   2.17131083e+02
   2.63552365e+02   3.60281690e+02   4.96598020e+02   6.92570054e+02
   1.16626626e+03   2.46236069e+03   2.99625801e+03   6.69000037e+03
   8.06758724e+03   1.01270054e+04   4.55320709e+04   6.98270995e+04
   2.39073909e+05]
sum(charges) = -0.00000
RMS(D_mu - calculated_D_mu) = 0.0873775873443
 <|D_mu - calculated_D_mu|> = 0.0694381438874
max|D_mu - calculated_D_mu| = 0.276736884434
lowest possible RMS(D_mu - calculated_D_mu) = 0.0682115296926
lowest possible max|D_mu - calculated_D_mu| = 0.412876029072
lowest possible <|D_mu - calculated_D_mu|> = 0.0528898168784

APT sum(charges) = -0.00000
APT max|D_mu - calculated_D_mu| = 1.13397454397
APT <|D_mu - calculated_D_mu|> = 0.373569556123
RESP sum(charges) = -0.00000
RESP max|D_mu - calculated_D_mu| = 0.450893695995
RESP <|D_mu - calculated_D_mu|> = 0.116060722949
--------------------------------------------------------------------------------
thy 660 conformers
--------------------------------------------------------------------------------
Total charge:    0.00000
```

```
avg |component of dipole moment| = 0.533152481358
Eigenvalues of A: [ -2.37945476e-13   6.83756605e-04   1.35198577e-03   2.83240101e-03
   8.80661202e-03   1.01652169e-02   1.43918256e-02   2.24038844e-02
   2.66953509e-02   3.64475216e-02   5.64690382e-02   7.10660702e-02
   9.51073527e-02   1.45281704e-01   1.55610646e-01   1.21487834e+00
   1.85739478e+00   4.23827344e+00   2.80359026e+01   3.40250386e+01
   1.68708280e+02   1.74241767e+02   1.99644487e+02   2.86733801e+02
   4.11422659e+02   5.83390988e+02   8.81782052e+02   1.06294822e+03
   2.01839807e+03   2.64389501e+03   5.81506059e+03   6.62449773e+03
   8.94119714e+03   2.47662231e+04   6.94905431e+04   1.91765085e+05]
sum(charges) = 0.00000
RMS(D_mu - calculated_D_mu) = 0.0938277883159
 <|D_mu - calculated_D_mu|> = 0.0737873756397
max|D_mu - calculated_D_mu| = 0.324114767921
lowest possible RMS(D_mu - calculated_D_mu) = 0.058050736819
lowest possible max|D_mu - calculated_D_mu| = 0.22356873809
lowest possible <|D_mu - calculated_D_mu|> = 0.0443742870311

APT sum(charges) = 0.00000
APT max|D_mu - calculated_D_mu| = 1.43351287103
APT <|D_mu - calculated_D_mu|> = 0.473736212581
RESP sum(charges) = -0.00000
RESP max|D_mu - calculated_D_mu| = 0.497907430054
RESP <|D_mu - calculated_D_mu|> = 0.109973277496
```








```
In [5]: # Print the tables with charges
        dd_minus_apt_all = []
        dd_minus_apt_mad = 0.0
```

```
        dd_tot_cnt = 0

        for key,rec in final_charges.items():
            q, apt_avg, resp_avg = rec['dd'], rec['apt'], rec['resp']

            print 'Here are the charges for << %s >>:' % key
            nAtoms = len(atom_names[key].items())
            print '%3s \t %7s \t %8s \t %8s \t %8s' % ('at_N', 'at_name', 'q_dd', 'q_apt', 'q_resp')
            for i in range(nAtoms):
                args = (i+1, atom_names[key][i+1], q[i], apt_avg[i], resp_avg[i])
                print '%3d \t %7s \t %8.3f \t %8.3f \t %8.3f' % args
                dff = abs(q[i] - apt_avg[i])
                dd_minus_apt_all.append( [dff, key, args] )
                dd_minus_apt_mad += dff
                dd_tot_cnt += 1

            print 'sum(charges) = %.4f' % sum(q)
            print
            print

        # analyze differences w.r.t. APT charges
        print '<|dd-apt|> = ', dd_minus_apt_mad / dd_tot_cnt
        print 'Quantiles:'
        s = sorted(dd_minus_apt_all, key=lambda x: -x[0])
        nAll = len(dd_minus_apt_all)
        for q in range(0,100,5):
            n = sum([ 1 if x[0] > q/100.0 else 0 for x in dd_minus_apt_all]  )
            print 'N_{diff > %.2f}: %4d / %4d = %8.2f %%' % (q/100.0,  n, nAll, n*100.0/nAll )

        print 'sorted |dd-apt|:'
        plt.hist([rec[0] for rec in dd_minus_apt_all])
        plt.show()
```

```
Here are the charges for << gua >>:
at_N          at_name              q_dd             q_apt            q_resp
  1             HO3'             0.293             0.261             0.402
  2              O3'            -0.590            -0.609            -0.627
  3              C3'             0.454             0.467             0.330
  4              C4'             0.450             0.410             0.110
  5              H3'            -0.093            -0.032             0.027
  6              C2'            -0.113            -0.084            -0.351
  7              C1'             0.901             0.916             0.345
  8              O4'            -0.553            -0.788            -0.398
  9              C5'             0.497             0.504             0.032
 10              H1'            -0.307            -0.053             0.061
 11             H2'2            -0.054             0.025             0.112
 12             H2'1             0.009            -0.003             0.125
 13              H4'            -0.090            -0.057             0.075
 14              O5'            -0.792            -1.013            -0.365
 15             H5'1            -0.115            -0.017             0.087
 16             H5'2            -0.125            -0.010             0.086
 17                P             2.329             2.330             0.966
 18               N9            -0.541            -0.572            -0.195
 19               C4             0.682             0.661             0.325
 20               C8             0.255             0.221             0.198
 21               N7            -0.411            -0.433            -0.511
 22               C5            -0.247            -0.260             0.073
 23               N3            -0.823            -0.847            -0.607
```

24	C2	1.155	1.149	0.726
25	N1	-0.659	-0.647	-0.678
26	C6	1.165	1.172	0.568
27	O6	-0.868	-0.844	-0.492
28	H1	0.168	0.209	0.391
29	N2	-0.813	-0.800	-0.823
30	H21	0.296	0.223	0.369
31	H22	0.320	0.255	0.381
32	H8	0.114	0.074	0.115
33	OP	-1.007	-0.832	-0.577
34	OP1	-0.839	-0.854	-0.570
35	HP1	0.392	0.366	0.427
36	OP2	-0.838	-0.853	-0.565
37	HP2	0.399	0.368	0.425

sum(charges) = -0.0000

Here are the charges for << ade >>:

at_N	at_name	q_dd	q_apt	q_resp
1	HO3'	0.298	0.262	0.402
2	O3'	-0.600	-0.611	-0.628
3	C3'	0.452	0.468	0.335
4	C4'	0.451	0.412	0.104
5	H3'	-0.089	-0.027	0.031
6	C2'	-0.119	-0.087	-0.347
7	C1'	0.909	0.923	0.348
8	O4'	-0.572	-0.792	-0.404
9	C5'	0.499	0.503	0.048
10	H1'	-0.366	-0.058	0.068
11	H2'2	-0.070	0.033	0.118
12	H2'1	0.023	-0.001	0.124
13	H4'	-0.080	-0.054	0.078
14	O5'	-0.793	-1.021	-0.371
15	H5'1	-0.115	-0.014	0.084
16	H5'2	-0.130	-0.012	0.081
17	P	2.319	2.329	0.966
18	N9	-0.523	-0.612	-0.264
19	C4	0.555	0.513	0.477
20	C8	0.404	0.327	0.251
21	N7	-0.495	-0.513	-0.541
22	C5	-0.193	-0.187	-0.002
23	N3	-0.600	-0.654	-0.649
24	C2	0.598	0.584	0.484
25	N1	-0.723	-0.687	-0.688
26	C6	0.946	0.997	0.643
27	N6	-0.964	-0.816	-0.824
28	H2	0.029	0.007	0.052
29	H62	0.337	0.247	0.385
30	H61	0.307	0.253	0.393
31	H8	0.198	0.081	0.109
32	OP	-1.008	-0.824	-0.575
33	OP1	-0.848	-0.848	-0.569
34	HP1	0.402	0.364	0.427
35	OP2	-0.843	-0.850	-0.567
36	HP2	0.404	0.365	0.425

sum(charges) = 0.0000

Here are the charges for << thy >>:

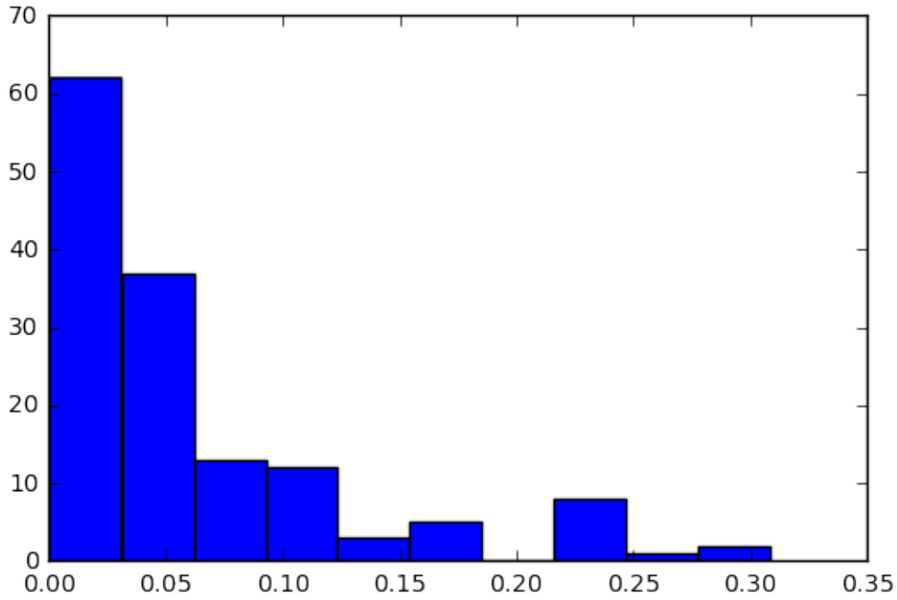
at_N	at_name	q_dd	q_apt	q_resp
1	HO3'	0.296	0.264	0.401
2	O3'	-0.583	-0.612	-0.626
3	C3'	0.454	0.471	0.345
4	C4'	0.461	0.426	0.094
5	H3'	-0.108	-0.033	0.034
6	C2'	-0.117	-0.082	-0.391
7	C1'	0.880	0.903	0.365
8	O4'	-0.651	-0.795	-0.403
9	C5'	0.497	0.501	0.005
10	H1'	-0.354	-0.045	0.061
11	H2'2	-0.081	0.025	0.127
12	H2'1	0.018	-0.000	0.132
13	H4'	-0.059	-0.058	0.084
14	O5'	-0.797	-1.016	-0.357
15	H5'1	-0.102	-0.013	0.099
16	H5'2	-0.126	-0.013	0.097
17	P	2.333	2.330	0.956
18	N1	-0.781	-0.882	-0.218
19	C2	1.293	1.208	0.698
20	C6	0.511	0.450	-0.114
21	C5	-0.273	-0.288	-0.023
22	C4	1.086	1.073	0.635
23	N3	-0.656	-0.710	-0.648
24	O2	-0.726	-0.856	-0.542
25	H6	0.136	0.080	0.180
26	C7	0.058	0.084	-0.416
27	O4	-0.831	-0.812	-0.505
28	H3	0.254	0.218	0.376
29	H73	-0.036	-0.006	0.128
30	H72	-0.065	-0.009	0.136
31	H71	-0.036	-0.011	0.136
32	OP	-1.012	-0.829	-0.576
33	OP1	-0.835	-0.840	-0.564
34	HP1	0.391	0.360	0.431
35	OP2	-0.819	-0.838	-0.563
36	HP2	0.382	0.353	0.429

sum(charges) = 0.0000

Here are the charges for << cyt >>:

at_N	at_name	q_dd	q_apt	q_resp
1	HO3'	0.294	0.258	0.401
2	O3'	-0.565	-0.612	-0.628
3	C3'	0.453	0.472	0.321
4	C4'	0.433	0.423	0.129
5	H3'	-0.117	-0.030	0.039
6	C2'	-0.117	-0.080	-0.360
7	C1'	0.851	0.880	0.367
8	O4'	-0.668	-0.791	-0.416
9	C5'	0.495	0.497	0.000
10	H1'	-0.284	-0.066	0.053
11	H2'2	-0.051	0.032	0.120
12	H2'1	-0.022	0.001	0.125
13	H4'	-0.098	-0.058	0.075

```
 14              O5'            -0.800            -1.016            -0.363
 15             H5'1            -0.099            -0.011             0.096
 16             H5'2            -0.100            -0.011             0.095
 17                P             2.326             2.329             0.968
 18               N1            -0.740            -0.776            -0.328
 19               C2             1.315             1.213             0.871
 20               N3            -0.873            -0.887            -0.780
 21               C4             1.112             1.144             0.858
 22               C5            -0.484            -0.492            -0.619
 23               C6             0.510             0.476             0.174
 24               H6             0.120             0.085             0.129
 25               H5             0.041             0.041             0.216
 26               N4            -0.991            -0.826            -0.895
 27              H41             0.249             0.240             0.408
 28              H42             0.342             0.235             0.390
 29               O2            -0.646            -0.878            -0.583
 30              OP1            -0.846            -0.848            -0.567
 31               OP            -1.014            -0.830            -0.582
 32              OP2            -0.827            -0.847            -0.565
 33              HP2             0.394             0.366             0.425
 34              HP1             0.405             0.369             0.425
sum(charges) = -0.0000


<|dd-apt|> =  0.0615956738245
Quantiles:
N_{diff > 0.00}:  143 /  143 =   100.00 %
N_{diff > 0.05}:   53 /  143 =    37.06 %
N_{diff > 0.10}:   30 /  143 =    20.98 %
N_{diff > 0.15}:   16 /  143 =    11.19 %
N_{diff > 0.20}:   11 /  143 =     7.69 %
N_{diff > 0.25}:    3 /  143 =     2.10 %
N_{diff > 0.30}:    2 /  143 =     1.40 %
N_{diff > 0.35}:    0 /  143 =     0.00 %
N_{diff > 0.40}:    0 /  143 =     0.00 %
N_{diff > 0.45}:    0 /  143 =     0.00 %
N_{diff > 0.50}:    0 /  143 =     0.00 %
N_{diff > 0.55}:    0 /  143 =     0.00 %
N_{diff > 0.60}:    0 /  143 =     0.00 %
N_{diff > 0.65}:    0 /  143 =     0.00 %
N_{diff > 0.70}:    0 /  143 =     0.00 %
N_{diff > 0.75}:    0 /  143 =     0.00 %
N_{diff > 0.80}:    0 /  143 =     0.00 %
N_{diff > 0.85}:    0 /  143 =     0.00 %
N_{diff > 0.90}:    0 /  143 =     0.00 %
N_{diff > 0.95}:    0 /  143 =     0.00 %
sorted |dd-apt|:
```

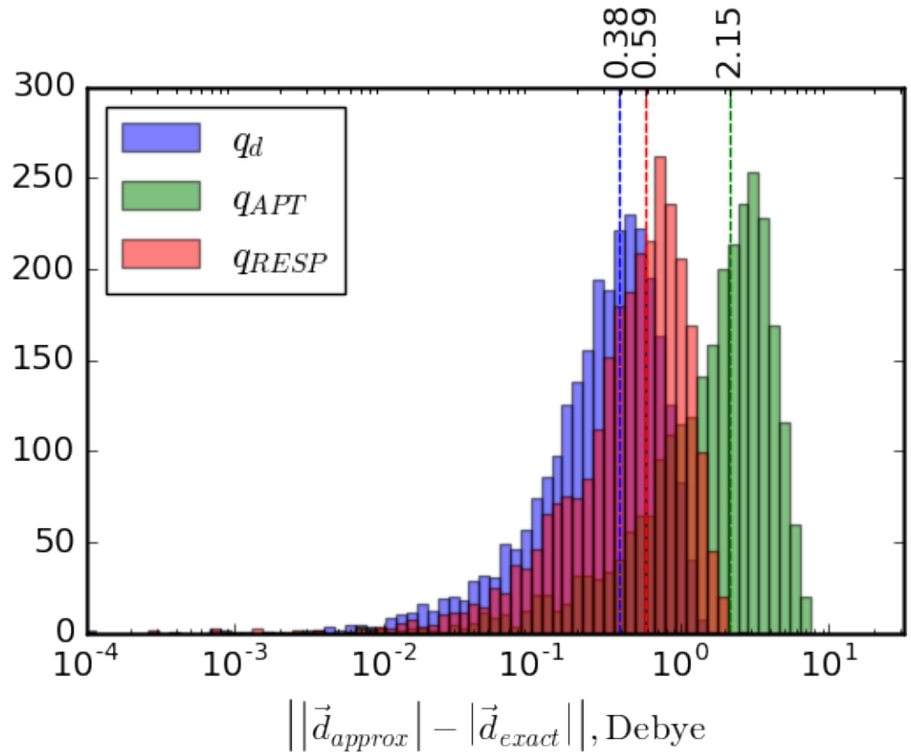


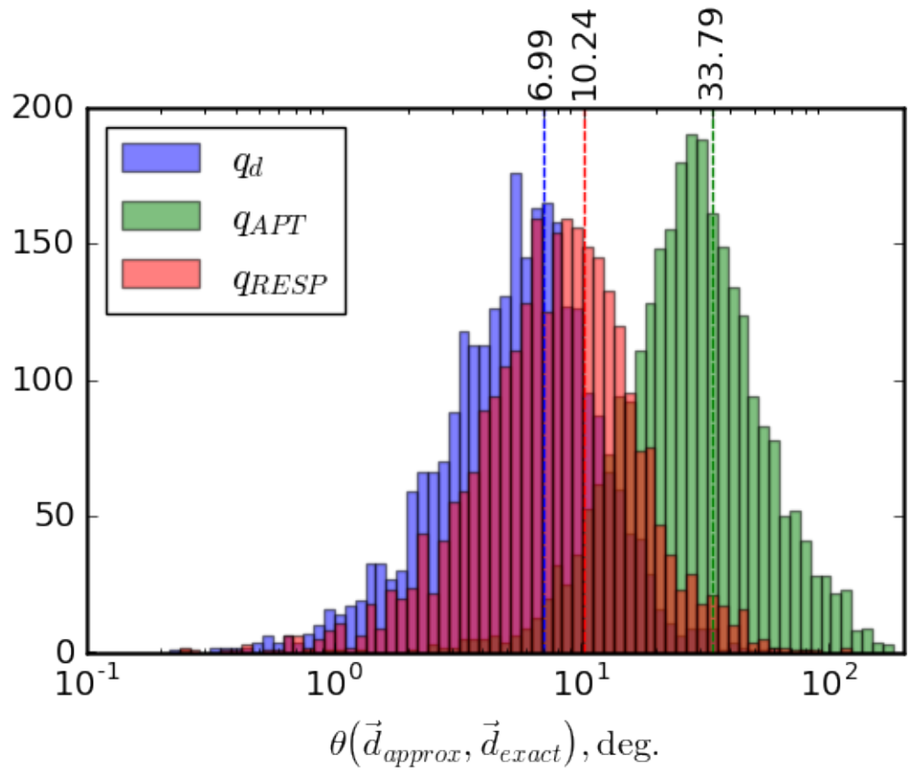
In [6]:
```
# Plot the distributions of approximation errors

symb2Z = {'H': 1.0, 'C': 6.0, 'N': 7.0, 'O': 8.0, 'P': 15}


distrs = {'abs': plt.figure().gca(), 'ang': plt.figure().gca() }

for g in distrs:
    #pass#
    distrs[g].set_xscale("log")
    distrs[g].tick_params(labelsize=14)

accumulated_distr = {}

for ch_type in ['dd', 'apt', 'resp']:
    d_abs_diffs = []
    d_angs = []
    for key,rec in final_charges.items():
        q = rec[ch_type]
        fname, _, __ = molecules[key]
        data_list = json.load(open(fname))

        nAtoms = len(atom_names[key].items())

        # sort atoms by their IDs and preserve their names only
        at_names = [v for (k,v) in sorted( atom_names[key].items(), key=lambda (k,v): k )]

        for R, D in zip( data_list['coords'], data_list['dipole']):
            ch_center = np.zeros(3)
            d_charges = np.zeros(3)
            geom_center = np.zeros(3)
            Ztot = 0.0
            for at_name, ri in zip(at_names, np.array(R)):
```

```
                Z = symb2Z[at_name[0]]
                ch_center += ri * Z
                Ztot += Z
            ch_center /= Ztot
            if np.linalg.norm( ch_center ) > 1e-6:
                print 'Warning: ch_center is not zero!', ch_center

            for i, ri in enumerate(np.array(R)):
                d_charges += q[i] * (ri - ch_center)
            d_charges /= Debye2eA
            D_eA = np.array(D)
            d_abs_diffs.append( np.abs(np.linalg.norm(d_charges) - np.linalg.norm(D_eA))  )
            cs = d_charges.dot(D_eA) / np.linalg.norm(d_charges)/np.linalg.norm(D_eA)
            d_angs.append( np.arccos( cs  )/np.pi*180 )
    print ch_type

    for lbl, data in zip(['||d|-|D|| ', 'angle(d,D)'],
                         [d_abs_diffs, d_angs] ):
        print '%s:  mean = %7.3f, median = %.5f, min = %.5f, max = %.3f' % (lbl,
                                                                            np.mean(data),
                                                                            np.median(data),
                                                                            min(data),
                                                                            max(data) )


    Lbls = {'dd': 'd', 'apt': 'APT', 'resp': 'RESP'}
    colors = {'dd': 'b', 'apt': 'g', 'resp': 'r'}

    for k, data, bins in zip(['abs', 'ang'],
                             [d_abs_diffs, d_angs],
                             [ np.logspace(-4, 1.5, 80), np.logspace(-1, 2.3, 80) ]):

        distrs[k].set_xlim(bins[0], bins[-1])
        distrs[k].hist(data, label="$q_{%s}$" % Lbls[ch_type], alpha=0.5,
                       bins=bins, color=colors[ch_type])
        x_av = np.mean(data)
        distrs[k].axvline(x_av, color=colors[ch_type],
                          linestyle=(0,(4,1)), linewidth=1)
        ax2 = distrs[k].twiny()
        ax2.set_xlim(distrs[k].get_xlim())
        ax2.set_xscale("log")
        ax2.tick_params(labelsize=14)
        ax2.set_xticks( [x_av])
        ax2.set_xticklabels( ['%.2f' % x_av], rotation=90, ha='center'  )
        #ax2.set_xlabel('Average')



    accumulated_distr[ch_type] = d_abs_diffs


distrs['abs'].set_xlabel(
    r'$\left|\left|\vec{d}_{approx}\right| - \left|\vec{d}_{exact}\right|\right|, '+
    r'\mathrm{Debye}$',
    fontsize=15)
distrs['ang'].set_xlabel(
    r'$\theta\left(\vec{d}_{approx}, \vec{d}_{exact}\right), \mathrm{deg.}$',
    fontsize=15)
```

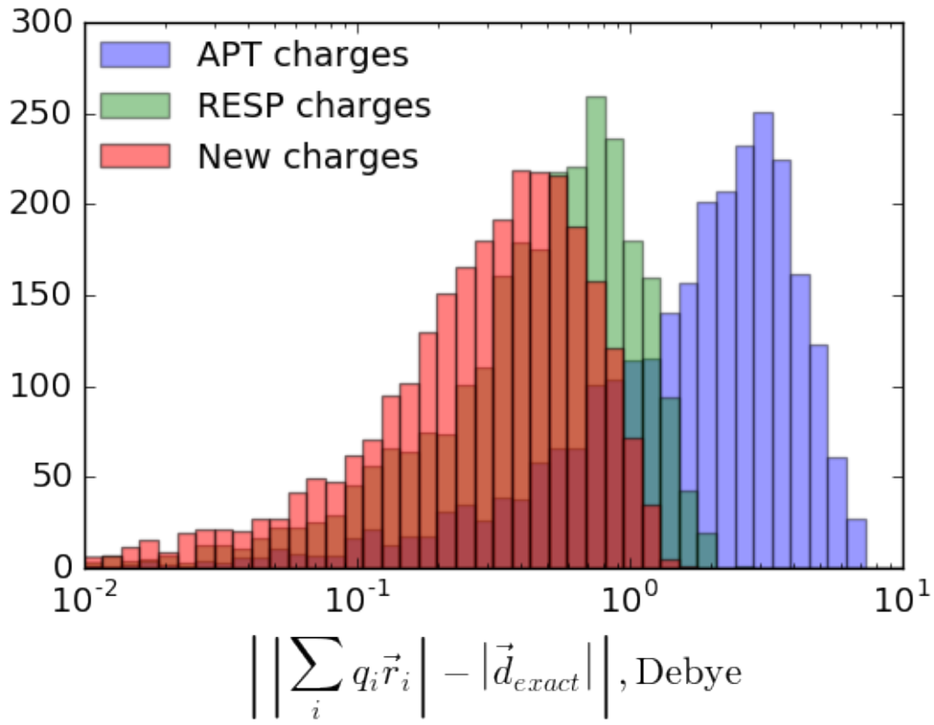
```
        for g in distrs:
            distrs[g].legend(loc='upper left', fontsize=16)
            distrs[g].figure.savefig('distr__%s.png' % g, dpi=300, bbox_inches='tight')

        plt.show()
```

```
dd
||d|-|D|| :  mean =   0.383, median = 0.33050, min = 0.00003, max = 1.646
angle(d,D):  mean =   6.987, median = 5.70319, min = 0.23297, max = 78.070
apt
||d|-|D|| :  mean =   2.153, median = 1.94872, min = 0.00038, max = 6.930
angle(d,D):  mean =  33.794, median = 27.89656, min = 0.23861, max = 170.198
resp
||d|-|D|| :  mean =   0.588, median = 0.53005, min = 0.00026, max = 2.805
angle(d,D):  mean =  10.236, median = 8.16056, min = 0.05253, max = 122.302
```





In [7]:
```
# Produce a similar plots for graphical abstract

bins = np.logspace(-2, 1.0, 45)

plt.hist(accumulated_distr['apt'], alpha=0.4,bins=bins, label='APT charges')
plt.hist(accumulated_distr['resp'], alpha=0.4,bins=bins, label='RESP charges')
plt.hist(accumulated_distr['dd'], alpha=0.5,bins=bins, label='New charges')

plt.gca().set_xscale("log")
plt.gca().tick_params(labelsize=14)
plt.gca().set_xlabel(
    r'$\left|\left| \sum_i{ \  q_i \vec{r}_i }  \right| - \left|\vec{d}_{exact}\right|\right|, \mathrm{D
    fontsize=16)
plt.legend(loc='upper left', fontsize=14, frameon=False, borderaxespad=0.1)


plt.savefig('GA_template.png', dpi=300, bbox_inches='tight')

plt.show()
```




```
In [8]: def base_dipole(q, key, data_list):
            base_atoms = dict([it for it in atom_names[key].items()
                               if it[1].find('P') == -1 and
                               (it[1].find("'") == -1 or it[1] == "C1'" ) ])
            print 'Base atoms:', base_atoms
            base_atoms_Z = [symb2Z[x[0]] for x in base_atoms.values()]
            base_atoms_Names = base_atoms.values()
            base_atoms = [x-1 for x in base_atoms.keys()] # make them 0-based
            d_vals = []

            # find the charge which we should assign to C1'
            Q_wo_c1prime = 0
            for i, nm in zip(base_atoms, base_atoms_Names):
                if nm != "C1'":
                    Q_wo_c1prime += q[i]
            print "Total charge without C1'", Q_wo_c1prime

            for R in data_list['coords']:


                ch_center = np.zeros(3)
                Ztot = 0.0

                # calculate the charge center
                for Z, i, nm, ri in zip(base_atoms_Z, base_atoms,
                                        base_atoms_Names, np.array(R)[base_atoms]):
                    if nm == "C1'":
                        Z = 6.0 + 3 * 1.0 # emulate methyl group
```

```
                ch_center += ri * Z
                Ztot += Z
            ch_center /= Ztot

            # calculate dipole moment
            dip_base = np.zeros(3)
            for i, nm, ri in zip(base_atoms, base_atoms_Names, np.array(R)[base_atoms]):
                at_q = q[i] if nm != "C1'" else -Q_wo_c1prime
                dip_base += at_q * (ri - ch_center)
            d_vals.append(  np.linalg.norm(dip_base) )

        d_avg = np.mean(d_vals)
        d_std = np.std(d_vals)
        print '%s: %.5f +/- %.5f eA, %.5f +/- %.5f Debye' % (key,
                                                             d_avg, d_std,
                                                             d_avg/Debye2eA, d_std/Debye2eA)
        print 'total base charge: %.5f' % sum( q[i] for i in base_atoms)
        print

    for key,rec in final_charges.items():

        fname, _, __ = molecules[key]
        data_list = json.load(open(fname))

        q, apt_avg, resp_avg = rec['dd'], rec['apt'], rec['resp']
        print key
        # analyze dipole moment of the nucleotide base
        print 'Averaged dipole moment of nucleotide base:'
        print ' - with new dipole-derived charges'
        base_dipole(q, key, data_list)
        print ' - with new <RESP>'
        base_dipole(resp_avg, key, data_list)
        print ' - with new <APT>'
        base_dipole(apt_avg, key, data_list)
```

```
  gua
  Averaged dipole moment of nucleotide base:
   - with new dipole-derived charges
  Base atoms: {32: 'H8', 7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2',
25: 'N1', 26: 'C6', 27: 'O6', 28: 'H1', 29: 'N2', 30: 'H21', 31: 'H22'}
  Total charge without C1' -0.207013582463
  gua: 1.49715 +/- 0.01494 eA, 7.19112 +/- 0.07175 Debye
  total base charge: 0.69397
   - with new <RESP>
  Base atoms: {32: 'H8', 7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2',
25: 'N1', 26: 'C6', 27: 'O6', 28: 'H1', 29: 'N2', 30: 'H21', 31: 'H22'}
  Total charge without C1' -0.160091150336
  gua: 1.31254 +/- 0.01434 eA, 6.30440 +/- 0.06890 Debye
  total base charge: 0.18539
   - with new <APT>
  Base atoms: {32: 'H8', 7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2',
25: 'N1', 26: 'C6', 27: 'O6', 28: 'H1', 29: 'N2', 30: 'H21', 31: 'H22'}
  Total charge without C1' -0.439996774497
  gua: 1.60838 +/- 0.01787 eA, 7.72540 +/- 0.08583 Debye
  total base charge: 0.47616
```

```
  ade
  Averaged dipole moment of nucleotide base:
   - with new dipole-derived charges
  Base atoms: {7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2', 25: 'N1',
26: 'C6', 27: 'N6', 28: 'H2', 29: 'H62', 30: 'H61', 31: 'H8'}
  Total charge without C1' -0.125158092584
  ade: 0.55459 +/- 0.00941 eA, 2.66379 +/- 0.04521 Debye
  total base charge: 0.78395
   - with new <RESP>
  Base atoms: {7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2', 25: 'N1',
26: 'C6', 27: 'N6', 28: 'H2', 29: 'H62', 30: 'H61', 31: 'H8'}
  Total charge without C1' -0.174762059229
  ade: 0.47603 +/- 0.01552 eA, 2.28646 +/- 0.07456 Debye
  total base charge: 0.17315
   - with new <APT>
  Base atoms: {7: "C1'", 18: 'N9', 19: 'C4', 20: 'C8', 21: 'N7', 22: 'C5', 23: 'N3', 24: 'C2', 25: 'N1',
26: 'C6', 27: 'N6', 28: 'H2', 29: 'H62', 30: 'H61', 31: 'H8'}
  Total charge without C1' -0.460179413223
  ade: 0.82920 +/- 0.01457 eA, 3.98282 +/- 0.06997 Debye
  total base charge: 0.46240
  thy
  Averaged dipole moment of nucleotide base:
   - with new dipole-derived charges
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'C6', 21: 'C5', 22: 'C4', 23: 'N3', 24: 'O2', 25: 'H6',
26: 'C7', 27: 'O4', 28: 'H3', 29: 'H73', 30: 'H72', 31: 'H71'}
  Total charge without C1' -0.0675305190885
  thy: 1.01273 +/- 0.00861 eA, 4.86435 +/- 0.04136 Debye
  total base charge: 0.81253
   - with new <RESP>
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'C6', 21: 'C5', 22: 'C4', 23: 'N3', 24: 'O2', 25: 'H6',
26: 'C7', 27: 'O4', 28: 'H3', 29: 'H73', 30: 'H72', 31: 'H71'}
  Total charge without C1' -0.179451927273
  thy: 0.88473 +/- 0.01453 eA, 4.24955 +/- 0.06980 Debye
  total base charge: 0.18534
   - with new <APT>
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'C6', 21: 'C5', 22: 'C4', 23: 'N3', 24: 'O2', 25: 'H6',
26: 'C7', 27: 'O4', 28: 'H3', 29: 'H73', 30: 'H72', 31: 'H71'}
  Total charge without C1' -0.459266366667
  thy: 1.84764 +/- 0.02592 eA, 8.87458 +/- 0.12450 Debye
  total base charge: 0.44380
```

```
  cyt
  Averaged dipole moment of nucleotide base:
   - with new dipole-derived charges
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'N3', 21: 'C4', 22: 'C5', 23: 'C6', 24: 'H6', 25: 'H5',
26: 'N4', 27: 'H41', 28: 'H42', 29: 'O2'}
  Total charge without C1' -0.0445629429197
  cyt: 1.28646 +/- 0.01314 eA, 6.17915 +/- 0.06314 Debye
  total base charge: 0.80637
   - with new <RESP>
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'N3', 21: 'C4', 22: 'C5', 23: 'C6', 24: 'H6', 25: 'H5',
26: 'N4', 27: 'H41', 28: 'H42', 29: 'O2'}
  Total charge without C1' -0.158119004894
  cyt: 1.38063 +/- 0.01640 eA, 6.63146 +/- 0.07879 Debye
  total base charge: 0.20879
   - with new <APT>
  Base atoms: {7: "C1'", 18: 'N1', 19: 'C2', 20: 'N3', 21: 'C4', 22: 'C5', 23: 'C6', 24: 'H6', 25: 'H5',
26: 'N4', 27: 'H41', 28: 'H42', 29: 'O2'}
  Total charge without C1' -0.425615606852
  cyt: 1.76089 +/- 0.03706 eA, 8.45791 +/- 0.17800 Debye
  total base charge: 0.45462
```

In [ ]:

Dipole-derived charges of and 5'-deoxycytidylic acid (dCMP) in polarizable dielectric envoronment (IEFPCM water, ε = 78.39)

Computation details. Each of the 613 conformers obtained earlier in vacuum[1] was additionally optimized in the presence of a continuum polarizable dielectric environment described by IEFPCM[2] solvation model with the settings corresponding to a standard solvent 'water' (in particular, dielectric constant of the solvent ε = 78.39, an optional value for the dielectric constant at infinite frequency ε_∞ = 1.776, the solvent radius 1.385 Å). The cavity GePol encompassing the dCMP molecule was built using the UFF atomic radii[3] (with the sphere radius scaling factor set to 1.0) and Hydrogen atoms having individual spheres (explicit hydrogens).
After geometry optimization, harmonic vibrational analysis was performed in order to obtain the values of APT charges in the presence of the environment. The calculations performed at this stage employed the same level of theory B3LYP/6-31G(d,p) as those performed in vacuum.
The same Python program was used for obtaining the dipole-derived charges in dielectric environment as that for vacuum case. Input data files used for the program can be found at *figshare* repository:
Nikolaienko, Tymofii (2017): Conformers of canonical 5'-deoxycytidylic acid molecule (dCMP) in continuum polarizable dielectric envoronment (IEFPCM water, ε = 78.39). figshare.; https://doi.org/10.6084/m9.figshare.5687419 ; Retrieved: 22:41, Dec 10, 2017 (GMT).

Results.

a) b)

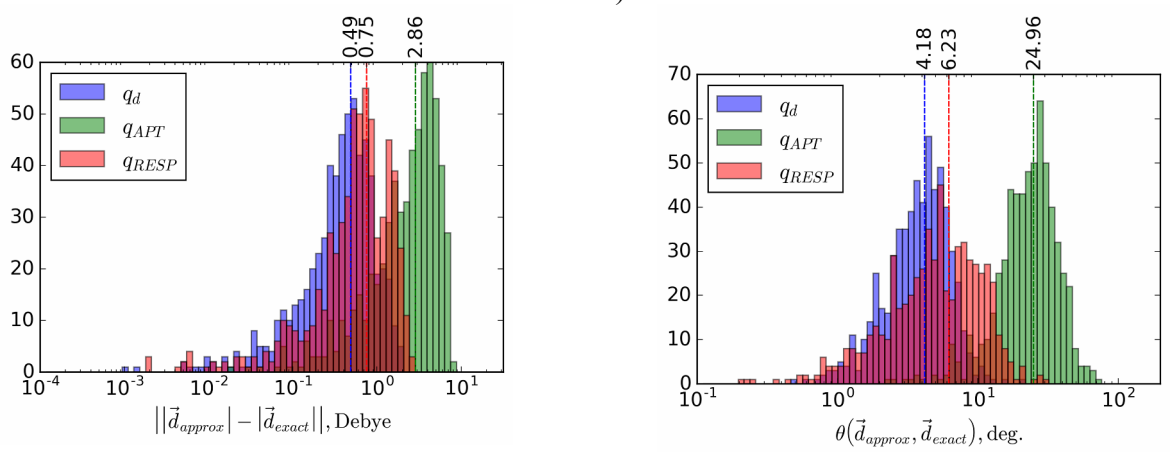


Figure SF1. The distribution of error measures for the approximation of exact dipole moment $\vec{d}_{exact}$ (obtained in polarizable continuum dielectric envoronment) with the value $\vec{d}_q$ calculated using the fixed RESP (q_{RESP}), APT (q_{APT}) and the dipole-derived (q_d) charges: errors in the absolute value (a) and the angle between $\vec{d}_{exact}$ and $\vec{d}_q$ (b). Note the logarithmic scale on the horizontal axis.

[1] See [T. Yu. Nikolaienko and D. M. Hovorun. Dopov. *Nac. akad. nauk Ukr.* 2010, ***(9)***, 173–184] and [T. Yu. Nikolaienko, L. A. Bulavin and D. M. Hovorun. (2017): Conformers of canonical 2'-deoxyribonucleotides, the model DNA monomers. figshare. https://doi.org/10.6084/m9.figshare.5258737.v1 Retrieved: 20:42, Jul 29, 2017 (GMT)] for details
[2] See [J. Tomasi, B. Mennucci and R. Cammi, *Chem. Rev.* 2005, ***105***, 2999–3094] for a review.
[3] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.*, ***114***:10024–10035, 1992.

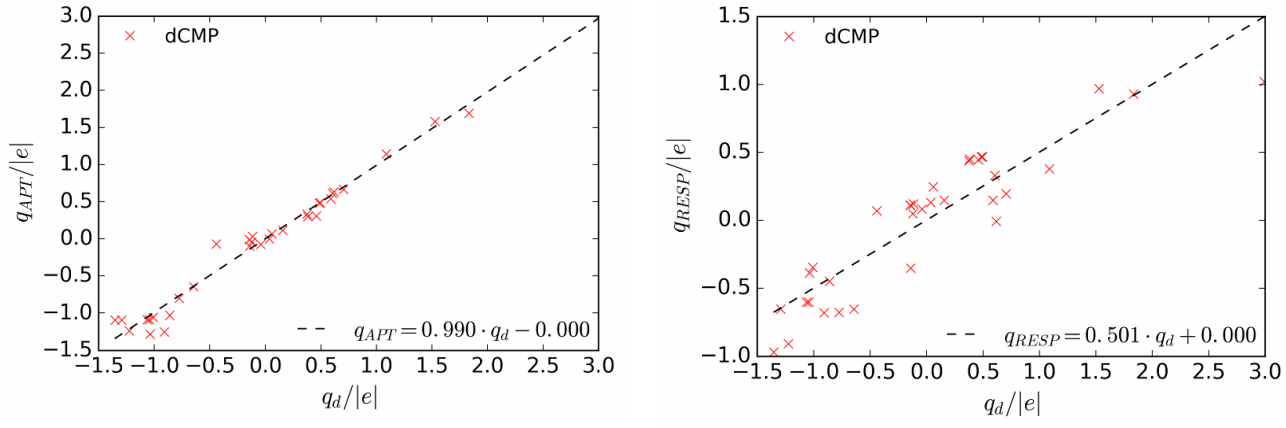


Figure SF2. Scatter plot of the effective atomic charges of 5'-deoxycytidylic acid molecule (dCMP) obtained in polarizable continuum dielectric envoronment by different methods: RESP (q_{RESP}), APT (q_{APT}) and the dipole-derived (q_d) charges proposed in this paper

Table ST1. Values of atomic charges of 5'-deoxycytidylic acid molecule (dCMP) in polarizable continuum dielectric envoronment calculated by envoronment by different methods: RESP (q_{RESP}), APT (q_{APT}) and the dipole-derived (q_d) charges proposed in this paper

Atom number	Atom Name	q d	q APT	q RESP
1	HO3'	0.374	0.333	0.436
2	O3'	-0.778	-0.800	-0.678
3	C3'	0.602	0.612	0.331
4	C4'	0.586	0.540	0.148
5	H3'	-0.124	-0.041	0.047
6	C2'	-0.141	-0.101	-0.353
7	C1'	1.087	1.144	0.378
8	O4'	-0.861	-1.027	-0.452
9	C5'	0.616	0.630	-0.008
10	H1'	-0.442	-0.068	0.068
11	H2'2	-0.117	0.031	0.121
12	H2'1	0.036	-0.002	0.130
13	H4'	-0.041	-0.075	0.082
14	O5'	-1.039	-1.279	-0.388
15	H5'1	-0.146	-0.012	0.111
16	H5'2	-0.145	-0.012	0.108
17	P	2.991	2.980	1.021
18	N1	-1.009	-1.057	-0.347
19	C2	1.833	1.692	0.930
20	N3	-1.228	-1.236	-0.906
21	C4	1.524	1.579	0.969
22	C5	-0.646	-0.646	-0.653
23	C6	0.703	0.663	0.195
24	H6	0.156	0.115	0.147
25	H5	0.059	0.067	0.245
26	N4	-1.356	-1.092	-0.970
27	H41	0.381	0.299	0.451
28	H42	0.458	0.308	0.446
29	O2	-0.910	-1.250	-0.681
30	OP1	-1.065	-1.088	-0.604
31	OP	-1.295	-1.090	-0.654
32	OP2	-1.044	-1.085	-0.603
33	HP2	0.484	0.482	0.466
34	HP1	0.495	0.487	0.466

sum(charges) = -0.0000