In [2]:
```
#%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
```

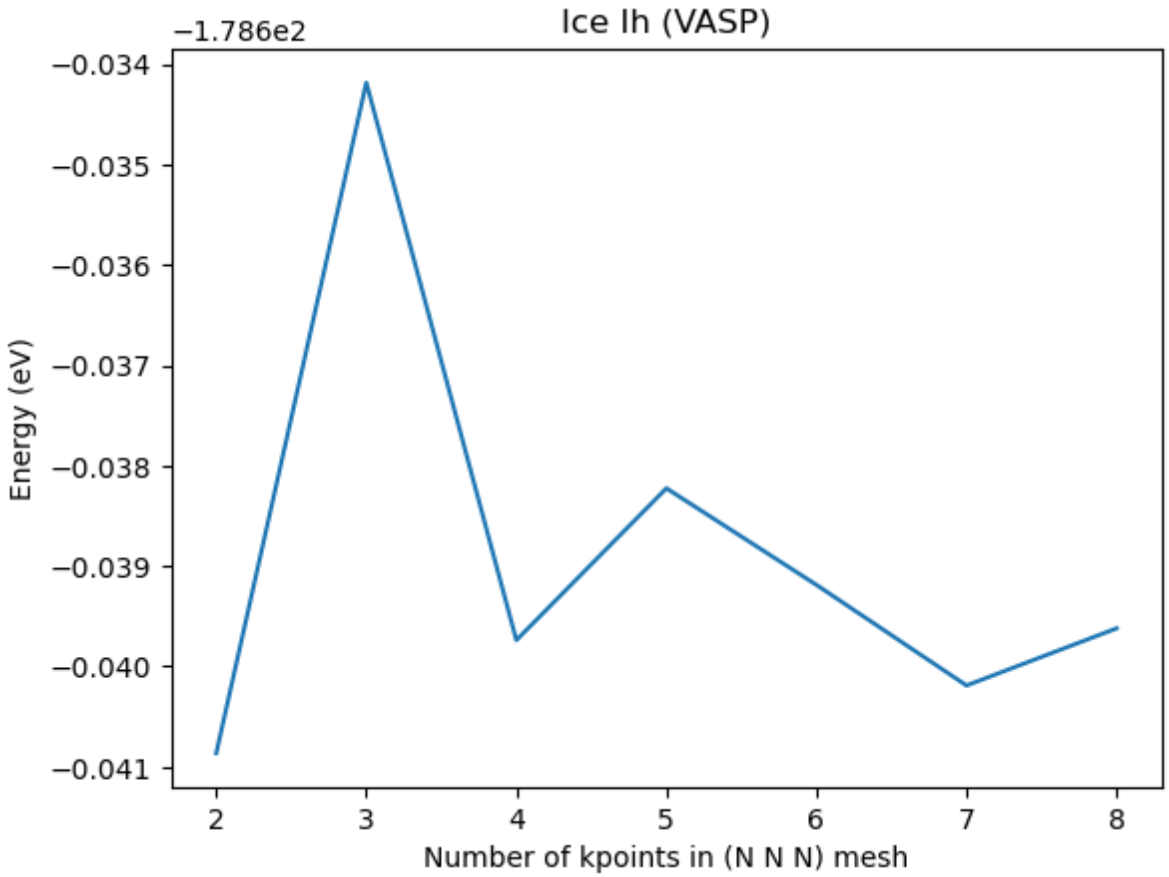
In [2]:
```
Requirement already satisfied: colorblind in c:\users\tabac\anaconda3\lib\sit
e-packages (0.0.9)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

In [4]:

In [5]:
```
plt.plot(cells_per_direction,energies)
plt.xlabel('Number of kpoints in (N N N) mesh')
plt.ylabel('Energy (eV)')
```

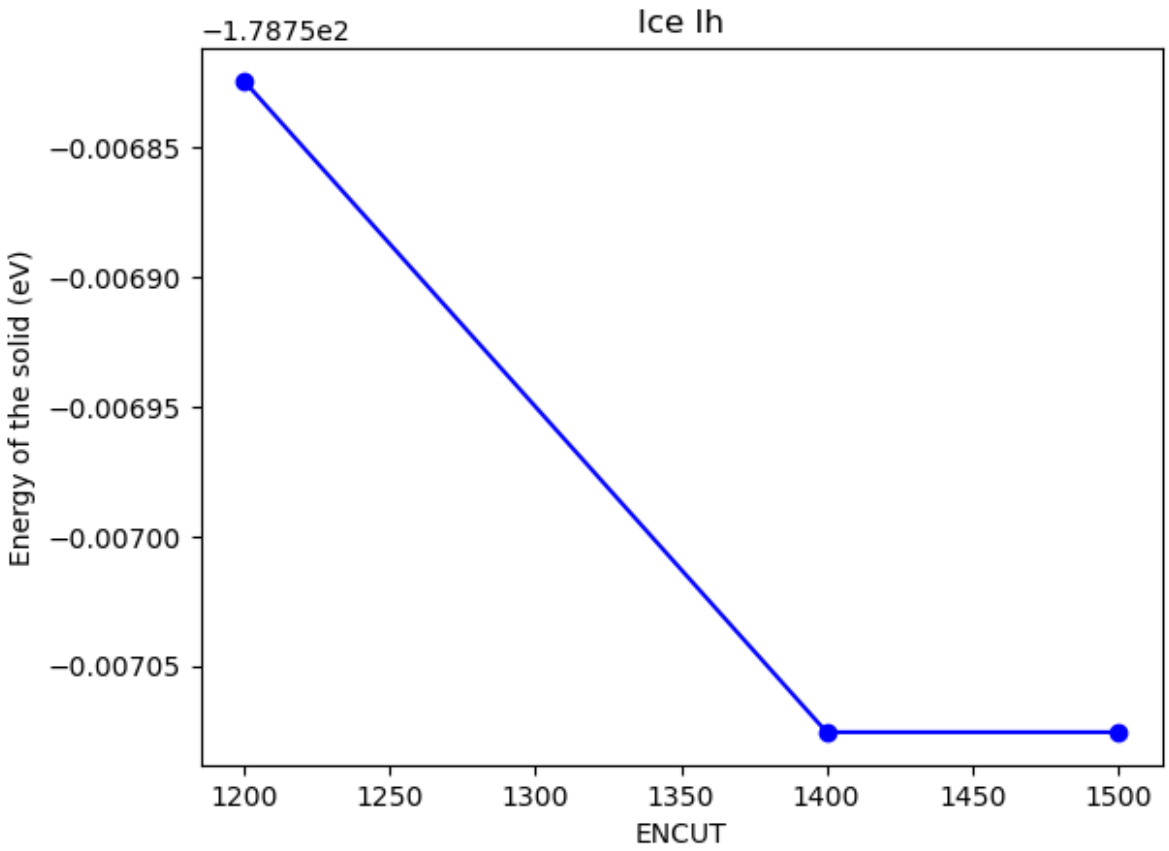
Out[5]: Text(0.5, 1.0, 'Ice Ih (VASP)')



In [75]:
```
cutoffs=np.array([300, 340, 380, 400, 420, 460, 500, 540, 580, 600, 800, 900,
energies_cutoff=np.array([-179.12687662,-178.65969720,-178.64669292, -178.6712
,-178.62559785
,-178.64503259,-178.74768068,-178.74890536,
-178.75139001,
-178.75399287,
-178.75682434,
-178.75707554,
-178.75707554
 ])
```

In [80]:
```
plt.plot(cutoffs[-3:], energies_cutoff[-3:], '-bo')
plt.title("Ice Ih")
plt.xlabel('ENCUT')
```

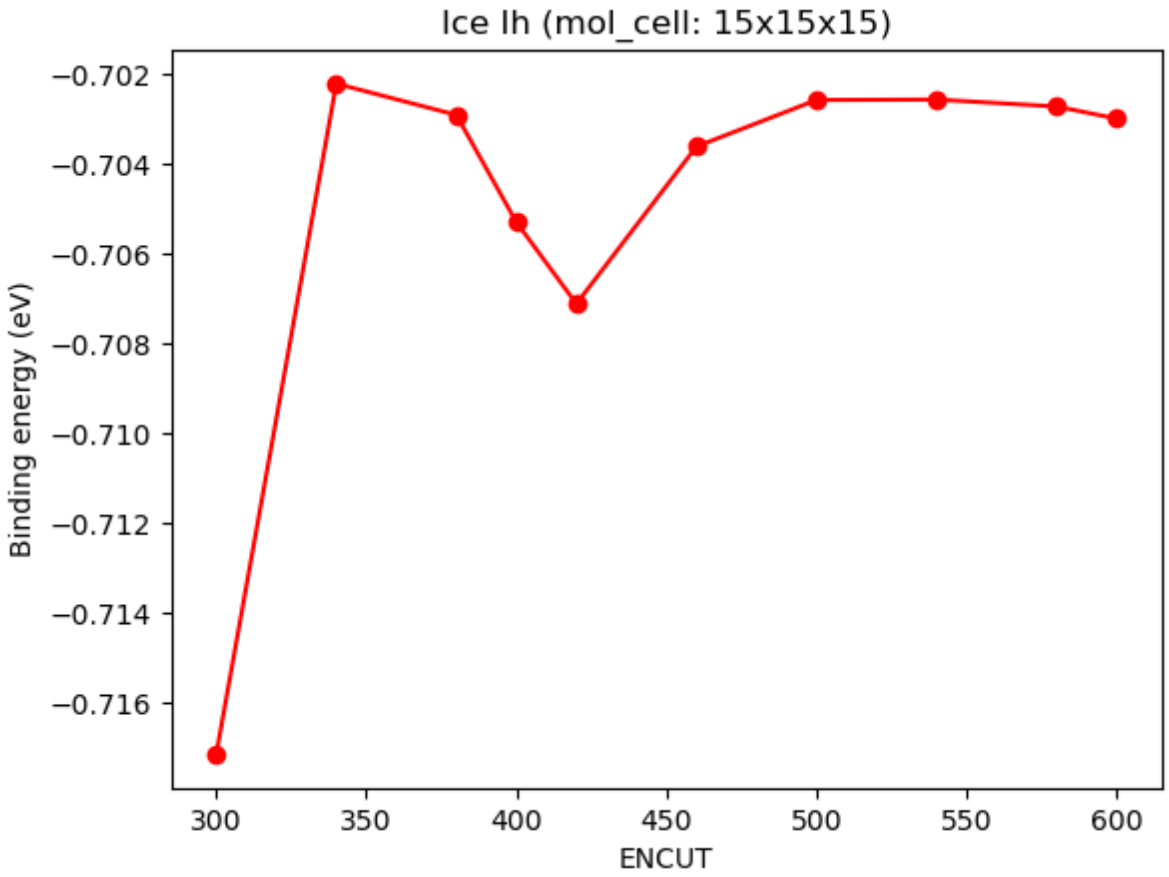
Out[80]: Text(0, 0.5, 'Energy of the solid (eV)')



In [8]:
```
bin_energy=np.array([-.71713691833333333333,
-.70220628000000000000,
-.70290334000000000000,
-.70529186416666666666,
-.70710846500000000000,
-.70360733583333333333,
-.70257005416666666666,
-.70256012416666666666,
-.70271534750000000000,
-.70298507916666666666,
                     ])
```

In [9]:
```
plt.plot(cutoffs, bin_energy, '-ro')
plt.title("Ice Ih (mol_cell: 15x15x15)")
plt.ylabel("Binding energy (eV)")
```

Out[9]:
```
Text(0.5, 0, 'ENCUT')
```

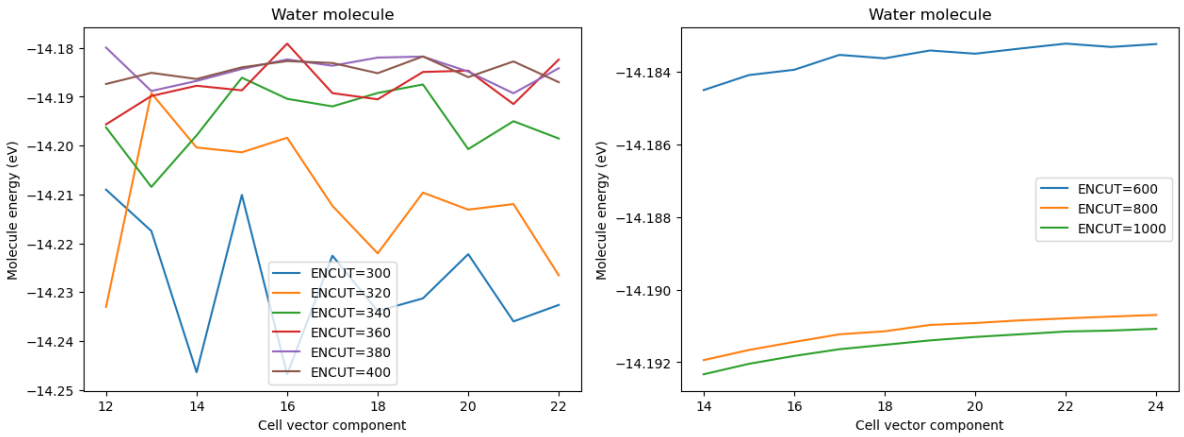


In [10]:
```
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(15, 5))



x=np.arange(12,23,1)
x2=np.arange(14,25,1)
y_400=np.array([-14.18739497,-14.18511328, -14.18637344,
-14.18397867, -14.18270360, -14.18309332,
 -14.18518881,
 -14.18175028,
 -14.18599548,
-14.18277823,
-14.18702443
])
y_300=np.array([-14.20902779, -14.21748356
, -14.24641462
, -14.21010189
, -14.24682772
, -14.22254668
, -14.23387548
, -14.23128286
, -14.22223243
, -14.23600063
, -14.23263078
])
y_320=np.array([-14.23300898, -14.18926801,-14.20040239, -14.20137595, -14.198
, -14.21312053
, -14.21197370
, -14.22656506
])
y_340=np.array([ -14.19628346
, -14.20846742
, -14.19792125
, -14.18610185
, -14.19042612
, -14.19198528
, -14.18923567
, -14.18749820
, -14.20072978
, -14.19501469
,-14.19856022
])
y_360=np.array([ -14.19566085
, -14.18985714
, -14.18775874
, -14.18869850
, -14.17912967
, -14.18926684
, -14.19053131
, -14.18493373
, -14.18466589
, -14.19150993
,-14.18239711
])
```

```
y_380=np.array([-14.17994373
,-14.18883189
,-14.18677409
,-14.18432259
,-14.18236608
,-14.18365712
,-14.18199032
,-14.18178349
,-14.18485347
,-14.18929358
,-14.18417092
])

y_1000=np.array([ -14.19233135
, -14.19204332
, -14.19182678
, -14.19164142
, -14.19152050
, -14.19139919
, -14.19130365
, -14.19123031
, -14.19115494
, -14.19112864
, -14.19108007
])
y_800=np.array([ -14.19194124, -14.19166290
, -14.19144108
, -14.19123118
, -14.19114828
, -14.19097354
, -14.19092048
, -14.19084640
, -14.19079299
,-14.19074318,-14.19069858

])

y_600=np.array([
 -14.18451344
,-14.18410064
, -14.18395459
, -14.18354529
, -14.18363919
, -14.18342406
, -14.18351227
, -14.18336920
, -14.18323506
, -14.18332669
,-14.18324717
])


ax1.plot(x,y_300, label='ENCUT=300')
ax1.plot(x,y_320,label='ENCUT=320')
ax1.plot(x,y_340, label='ENCUT=340')
ax1.plot(x,y_360, label='ENCUT=360')
```

```
ax1.plot(x,y_380,label='ENCUT=380')
ax1.plot(x,y_400,label='ENCUT=400')
ax1.set_ylabel('Molecule energy (eV)')
ax1.set_xlabel('Cell vector component')
ax1.set_title('Water molecule')
ax1.legend()
plt.xlabel("Cell vector component")
plt.ylabel("Molecule energy (eV)")
plt.title("Water molecule")
ax2.plot(x2,y_600,label='ENCUT=600')
ax2.plot(x2,y_800,label='ENCUT=800')
ax2.plot(x2,y_1000,label='ENCUT=1000')
```

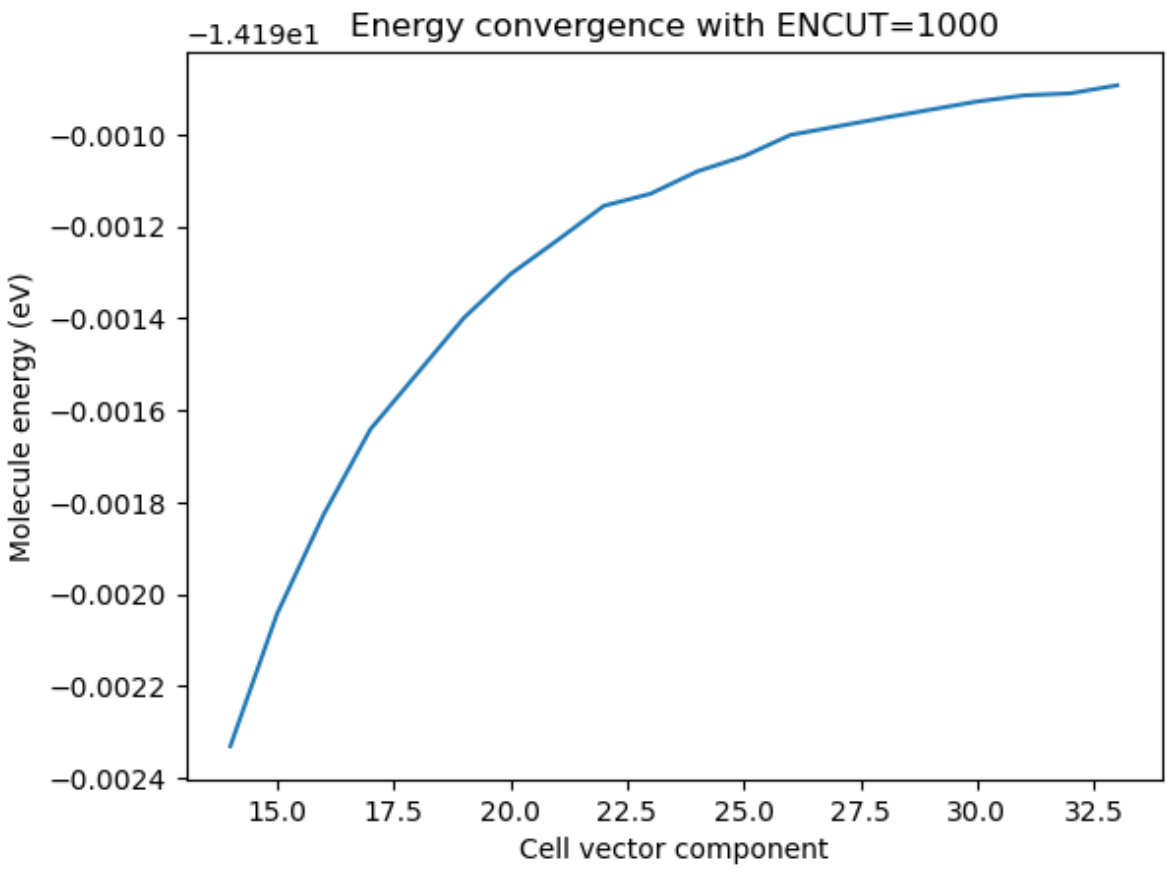
Out[10]: <matplotlib.legend.Legend at 0x268288afc10>



In [11]:
```
y_1000_1=np.array([-14.19104703
,-14.19100054
, -14.19096341
, -14.19092769
, -14.19091450
, -14.19090994
, -14.19089241])

y_1000_3=np.concatenate((y_1000,y_1000_1))
print(y_1000_3)
x_1000=np.arange(14,34,1)

x_1000=np.delete(x_1000, np.where(x_1000==29))
x_1000=np.delete(x_1000, np.where(x_1000==27))
print(x_1000)
plt.title('Energy convergence with ENCUT=1000')
plt.xlabel("Cell vector component")
plt.ylabel("Molecule energy (eV)")
```

```
[-14.19233135 -14.19204332 -14.19182678 -14.19164142 -14.1915205
 -14.19139919 -14.19130365 -14.19123031 -14.19115494 -14.19112864
 -14.19108007 -14.19104703 -14.19100054 -14.19096341 -14.19092769
 -14.1909145  -14.19090994 -14.19089241]
[14 15 16 17 18 19 20 21 22 23 24 25 26 28 30 31 32 33]
```

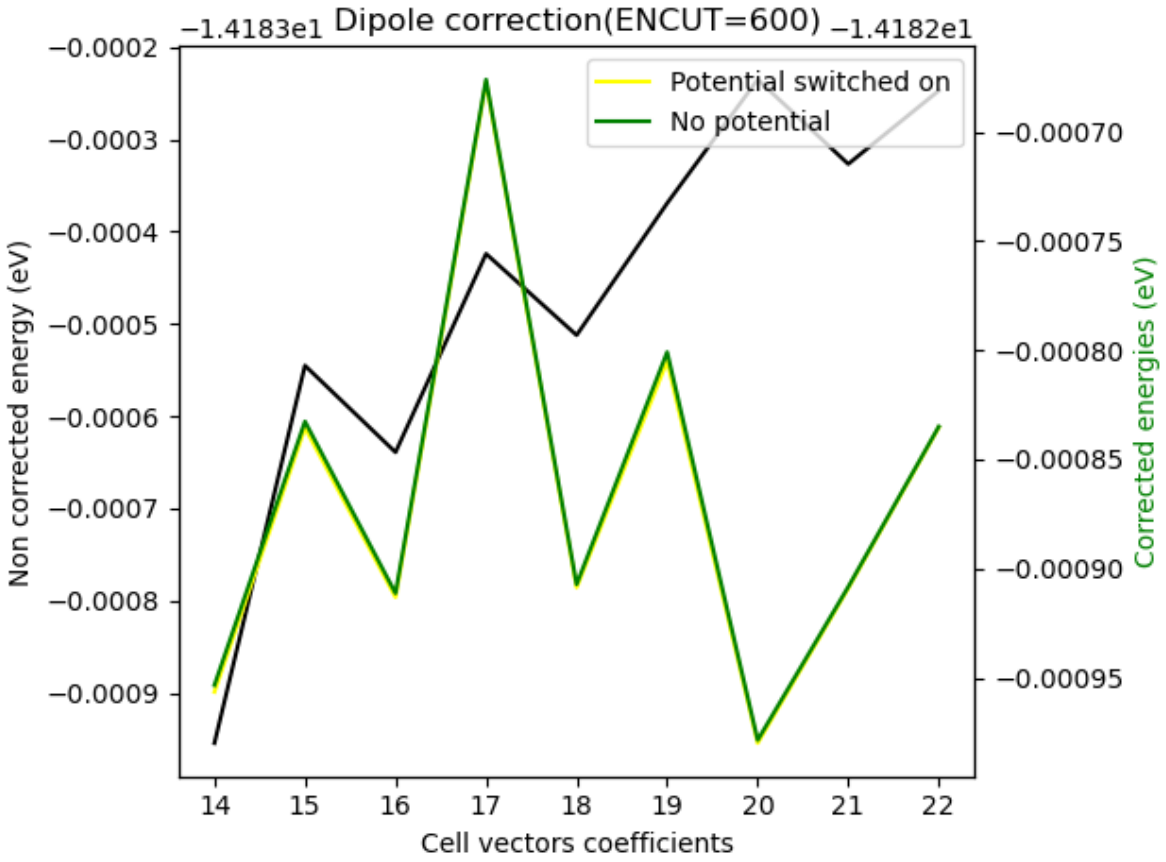
Out[11]:
```
[<matplotlib.lines.Line2D at 0x26828c83c40>]
```



In [12]:
```
en_dip=np.array([-14.18295337,
-14.18283262,
-14.18291129,
-14.18267595,
-14.18290718,
-14.18280081,
-14.18297853,
-14.18290836 ,
-14.18283493 ])
```

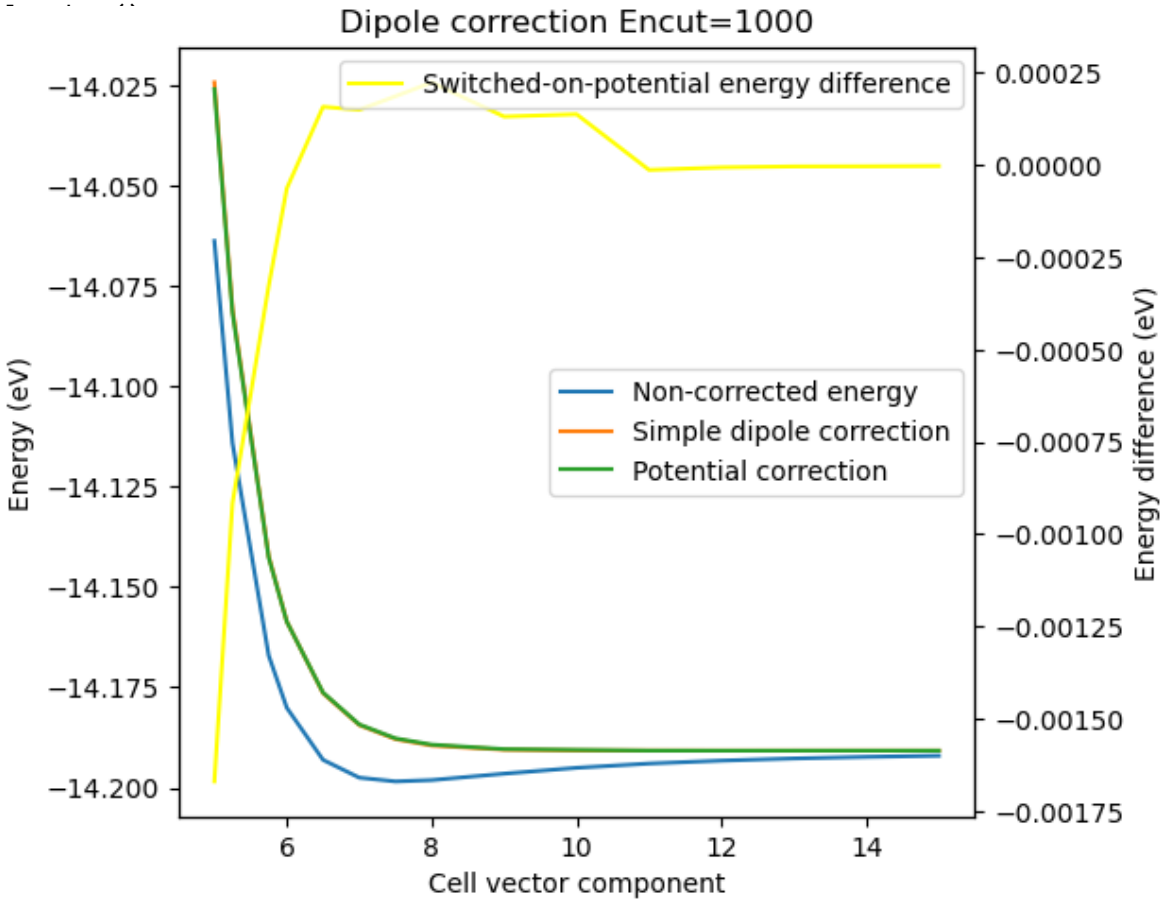
In [13]:
```
en_pot=np.array([-14.18295650
 ,-14.18283553
 ,-14.18291312
,-14.18267722
 ,-14.18290868
 ,-14.18280499
 ,-14.18297996
 ,-14.18290909
,-14.18283538])
```

In [14]:
```
fig, ax1 = plt.subplots()
x_cell=np.arange(14,23,1)

ax1.set_xlabel('Cell vectors coefficients')
ax1.set_ylabel('Non corrected energy (eV)')
ax1.plot(x_cell,y_600[2:], 'black')
ax1.tick_params(axis='y')

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

ax2.set_ylabel('Corrected energies (eV)',color='green')  # we already handled
plt.plot(x_cell,en_pot,label='Potential switched on',color='yellow')
plt.plot(x_cell,en_dip,label='No potential', color='green')
ax2.tick_params(axis='y')
plt.legend()
plt.title('Dipole correction(ENCUT=600)')
fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



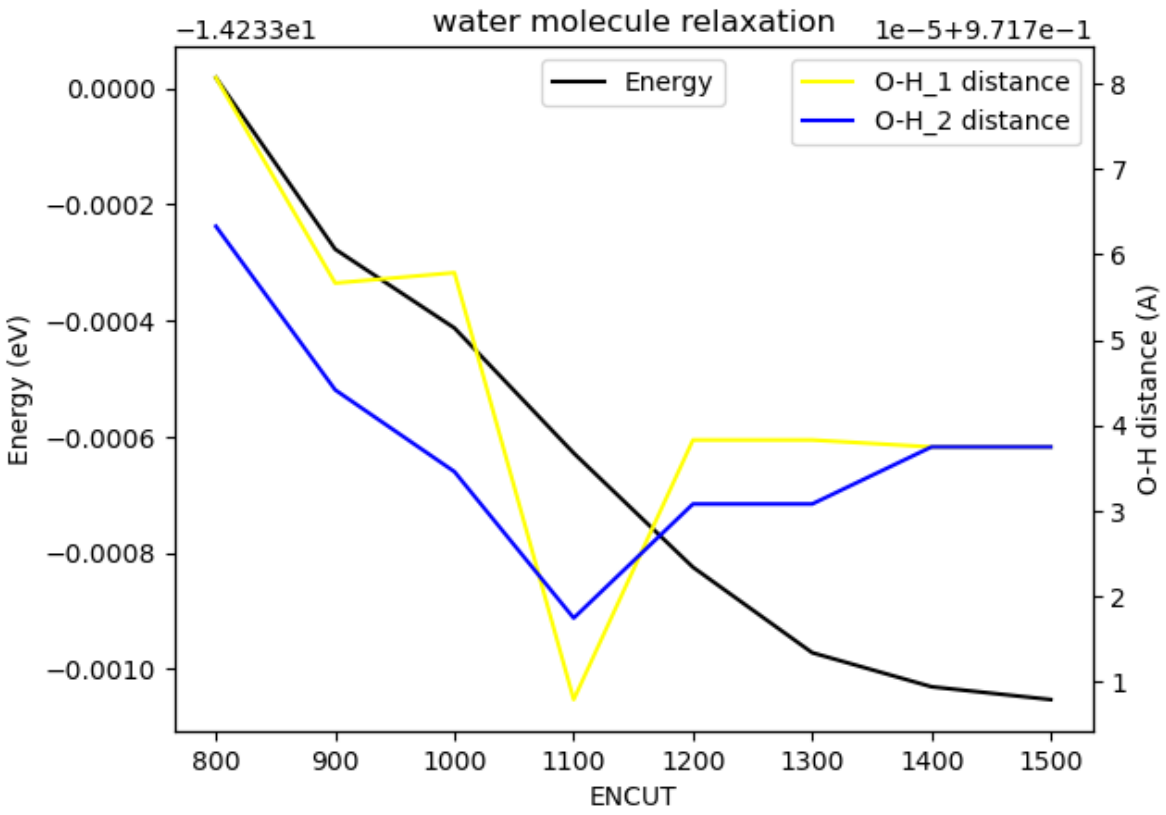
In [15]:
```
fig, ax1 = plt.subplots()
dp_1000=np.array([-14.02441881, -14.08077779,-14.14238098,
-14.15863502,-14.17647718,
-14.18440415,-14.18790911,
-14.18950297,
-14.19053277 ,
-14.19072655 ,
-14.19075371 ,
-14.19077506 ,
-14.19077252 ,
-14.19076971 ,
-14.19077465
])
no_corr=np.array([-14.06388021,-14.11421133,-14.16710044
,-14.18011805 ,-14.19306949 ,
-14.19750441 ,-14.19844428,
 -14.19811330 ,
-14.19651173 ,
-14.19505426 ,
-14.19399128 ,
-14.19326266 ,
-14.19272564 ,
-14.19233135 ,
-14.19204294,
])


pot_1000=np.array([-14.02608877,-14.08169349, -14.14270806,
-14.15869782,-14.176318942
,-14.18425408,-14.18772242
,-14.18927880
,-14.19040115
, -14.19058861
, -14.19076663
,-14.19078115
,-14.19077595
,-14.19077281
,-14.19077695])
x_dp=np.array([5,5.25,5.75,6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15])



ax1.tick_params(axis='y')
ax1.set_xlabel('Cell vector component')
ax1.set_ylabel('Energy (eV)')
ax1.plot(x_dp,no_corr,label='Non-corrected energy')
ax1.plot(x_dp,dp_1000,label='Simple dipole correction')
ax1.plot(x_dp,pot_1000,label='Potential correction')

ax2 = ax1.twinx()
ax2.tick_params(axis='y')
ax2.set_ylabel('Energy difference (eV)')
ax2.plot(x_dp,pot_1000-dp_1000,color='yellow',label='Switched-on-potential ene
ax1.legend(loc='center right')
```

```
ax2.legend(loc='upper right')
fig.tight_layout()
plt.title('Dipole correction Encut=1000')
```



In [71]:
```
a=np.arange(800,1600,100)
en=np.array([-14.23298219 ,
-14.2332774 ,-14.23341305 ,
-14.23362835 ,
-14.23382474 ,
-14.23397193,
-14.23403052 ,
-14.23405259 ,
])
OH_0=np.array([0.9717806718082017 ,
0.9717566348114121, 0.9717578497753441 ,
0.9717078908293376 ,
0.9717382713982196 ,
0.9717382713982196,
0.9717374597081251 ,
0.9717374597081251 ])
OH_1=np.array([0.9717632791992089 ,
0.971744135871166 ,0.9717345642200858 ,
0.9717174321787166 ,
0.9717307837050346 ,
0.9717307837050346,
0.9717374597081251 ,
0.9717374597081251 ])
fig, ax1 = plt.subplots()
ax2=ax1.twinx()
ax1.plot(a,en,label='Energy',color='black')
ax2.plot(a,OH_0,label='O-H_1 distance',color='yellow')
ax2.plot(a,OH_1,label='O-H_2 distance',color='blue')
ax1.set_xlabel('ENCUT')
ax1.set_ylabel('Energy (eV)')
ax2.set_ylabel('O-H distance (A)')
plt.legend()
ax1.legend(loc='upper center')
```

Out[71]:
```
Text(0.5, 1.0, 'water molecule relaxation')
```


water molecule relaxation
−1.4233e1
1e−5+9.717e−1
Energy
O-H_1 distance
O-H_2 distance
Energy (eV)
O-H distance (A)
ENCUT
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
800
900
1000
1100
1200
1300
1400
1500
1
2
3
4
5
6
7
8

In [36]:
```
#BBSE and "MOLECULE" calculations from CRYSTALS for all the monomers in the un
BSSE_i=np.array([-7.6385783503105E+01,
-7.6385780467958E+01,
-7.6385785980236E+01,
-7.6385780469425E+01,
-7.6385740501031E+01,
-7.6385741108006E+01,
-7.6385740375840E+01,
-7.6385740473011E+01,
-7.6385783502989E+01,
-7.6385783926020E+01,
-7.6385740473393E+01,
-7.6385780467779E+01])

BSSE_vi=np.array([-7.6386179057642E+01,
-7.6386769033594E+01,
-7.6386769487429E+01,
-7.6386178995764E+01,
-7.6386257458105E+01,
-7.6386697404311E+01,
-7.6386755974769E+01,
-7.6386257758761E+01,
-7.6386697257124E+01,
-7.6386178995512E+01])

BSSE_xv=np.array([-7.6386827629818E+01,
-7.6386356505834E+01,
-7.6386849497127E+01,
-7.6386274111644E+01,
-7.6386827634392E+01,
-7.6386356529247E+01,
-7.6386849417353E+01,
-7.6386827629829E+01,
-7.6386827634239E+01,
-7.6386849418142E+01])

BSSE_iii=np.array([-7.6386827629818E+01,
-7.6386356505834E+01,
-7.6386849497127E+01,
-7.6386274111644E+01,
-7.6386827634392E+01,
-7.6386356529247E+01,
-7.6386849417353E+01,
-7.6386827629829E+01,
-7.6386827634239E+01,
-7.6386849418142E+01])

av_i=np.mean(BSSE_i)
av_vi=np.mean(BSSE_vi)
av_xv=np.mean(BSSE_xv)
av_iii=np.mean(BSSE_iii)

mol_I = np.array([
-7.6384945740878E+01,
-7.6384945827747E+01,
```

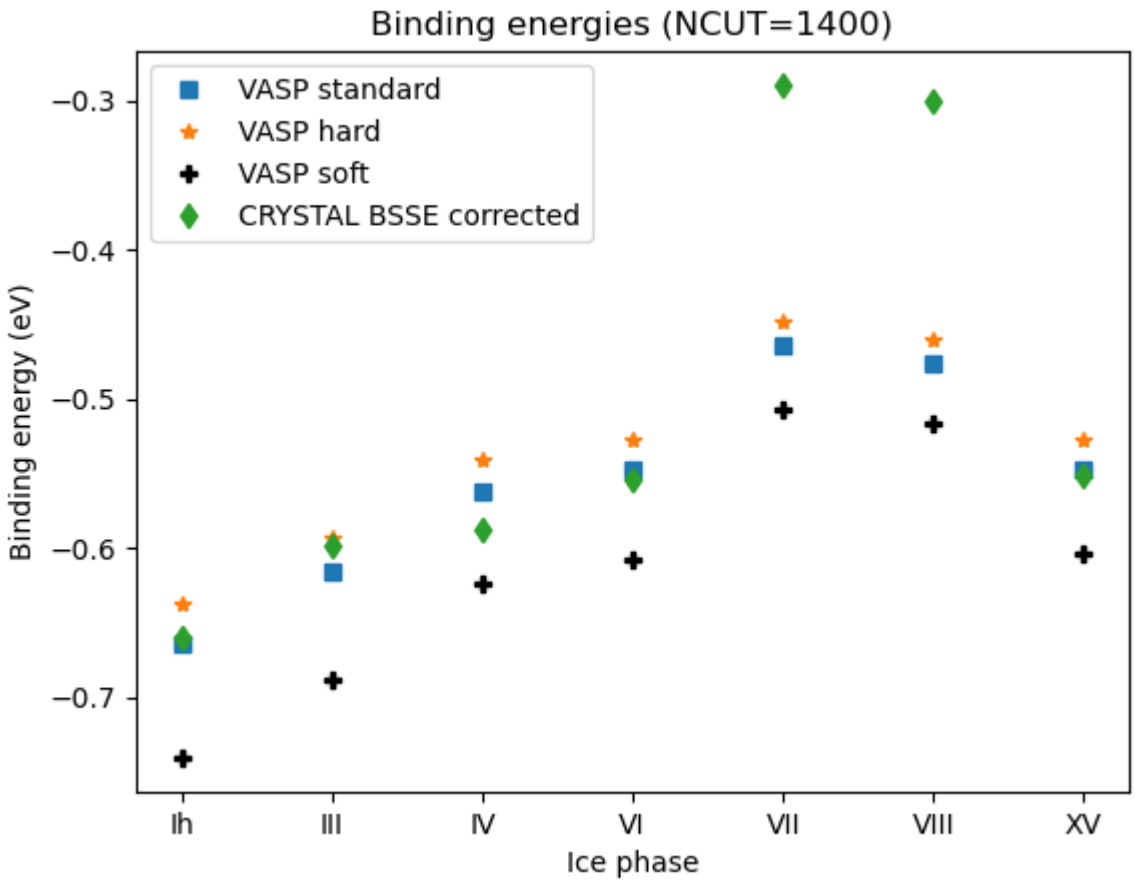
```
-7.6384945925046E+01,
-7.6384945827929E+01,
-7.6384911798649E+01,
-7.6384912424154E+01,
-7.6384911704601E+01,
-7.6384911599857E+01,
-7.6384945741600E+01,
-7.6384946205832E+01,
-7.6384911599790E+01,
-7.6384945827702E+01
])

mol_VI = np.array([
-7.6385491263581E+01,
-7.6385972519900E+01,
-7.6385972977010E+01,
-7.6385491247158E+01,
-7.6385549637020E+01,
-7.6385863898342E+01,
-7.6385929410543E+01,
-7.6385549959048E+01,
-7.6385863765507E+01,
-7.6385491246788E+01
])
mol_XV = np.array([
-7.6386031952841E+01,
-7.6385656066415E+01,
-7.6386026101350E+01,
-7.6385591768506E+01,
-7.6386031956859E+01,
-7.6385656090063E+01])

mol_III = np.array([
-7.6385320818423E+01,
-7.6385305772489E+01,
-7.6385305772313E+01,
-7.6385320818586E+01,
-7.6384574528443E+01,
-7.6384786861399E+01,
-7.6385346158616E+01,
-7.6385294289254E+01,
-7.6385320818347E+01,
-7.6385465005102E+01,
-7.6385337682753E+01,
-7.6385294289344E+01,
])


av_mol_I=np.mean(mol_I)
av_mol_VI= np.mean(mol_VI)
av_mol_XV=np.mean(mol_XV)
-76.38583232267233
```

In [128]:
```
#the followings have ENCUT=1400
sol_en_vasp_st=np.array([-178.75683367,-178.18900048
,-236.71325878,
-147.80488309
,-176.35769502,
-117.66876624,
-147.79891335])

sol_en_vasp_hard=np.array([-179.07394826
,-178.53006005
,-237.21142940
,-148.12479698
,-176.79325888
,-117.95948727
,-148.12308535
])

sol_en_vasp_soft=np.array([
-177.31633795
,-176.68568354
,-234.54958171
,-116.43324602-30
,-174.52143799
,-146.41561331+30
,-146.39028846
])

#phases I, III, VI and XV  energies have been computed with def2-QZVP basis se

sol_en_cry=np.array([-9.1693338073488E+02   , -9.169137356764E+02,-1.222521966

mol_en_cry = np.array([-7.6371201858219*10, -7.6371604638737*10, -7.6371104710
ghost_en_cry = np.array([ -7.6375995883096E+01
, -7.6376692516403E+01
, -7.6375804950590E+01
, -7.6375637566095E+01
, -7.6376103257614E+01
,-7.6376073263530*10
,-7.6376728953341E+01
])
bsse=ghost_en_cry-mol_en_cry
#bind_cry= (sol_en_cry/(np.array([36,36,48,30,36,24,30])/3)-(ghost_en_cry))*27
bind_vasp_st = sol_en_vasp_st/(np.array([36,36,48,30,36,24,30])/3)-(-14.232700
bind_vasp_soft = sol_en_vasp_soft/(np.array([36,36,48,30,36,24,30])/3)-( -14.0
bind_vasp_hard = sol_en_vasp_hard/(np.array([36,36,48,30,36,24,30])/3)-(-14.28

#I have computed a mean value of the bsse of all the monomers in the unit cell
bsse_1=np.array([av_i-av_mol_I,av_iii-av_mol_III,0,av_vi-av_mol_VI,0,0,av_xv-a
#the water molecule structure has been taken from VASP after relaxing one clus
bind_cry_1= (sol_en_cry/(np.array([36,36,48,30,36,24,30])/3)-(-7.6386024531398



x=['Ih','III','IV','VI','VII','VIII','XV']
#plt.plot(x,bind_cry,'o', label = 'CRYSTAL 3D MOLECULE')
```

```
plt.plot(x,bind_vasp_st, 's', label= 'VASP standard')
plt.plot(x,bind_vasp_hard, '*', label= 'VASP hard')
plt.plot(x,bind_vasp_soft, 'P', label= 'VASP soft',color='black')
plt.plot(x,bind_cry_1,'d',label = 'CRYSTAL BSSE corrected')
plt.xlabel('Ice phase')
plt.ylabel('Binding energy (eV)')
plt.legend(loc='upper left')
print('bind_vasp_hard',bind_vasp_hard)
print('bind_cry_1',bind_cry_1)
print('bind_vasp_st',bind_vasp_st)

plt.title('Binding energies (NCUT=1400)')


 #all the molecules retrieved from the solids are higher in energy than the 0 
```

```
bind_vasp_hard [-0.63803266 -0.59270864 -0.54091798 -0.52768334 -0.44797521 -
0.46013955
 -0.52751218]
bind_cry_1 [-0.66007015 -0.59842173 -0.58772166 -0.55481515 -0.28982129 -0.30
006131
 -0.55213382]
bind_vasp_st [-0.66370248 -0.61638304 -0.56187834 -0.54778798 -0.46377425 -0.
47589545
 -0.547191  ]
```

Out[128]: Text(0.5, 1.0, 'Binding energies (NCUT=1400)')



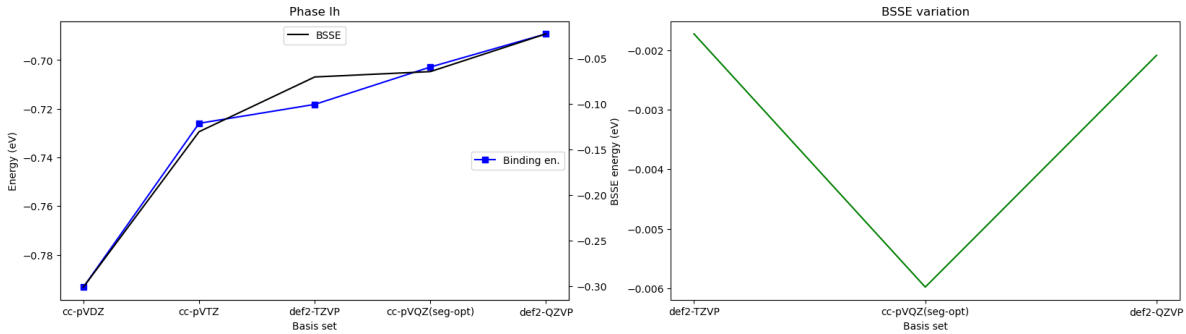
In [134]:

```
mean_f=['cc-pVDZ','cc-pVTZ','def2-TZVP', 'cc-pVQZ(seg-opt)', 'def2-QZVP']
sol_VIII=np.array([-6.1094645175128E+02,-6.1117641257992E+02,-6.1118502204528E
sol_I=np.array([-9.1647360365420E+02,-9.1683209831799E+02, -9.1684936534795E+0
BSSE_I= np.array([-7.6343649318262E+01,
-7.6375995883096E+01,
-7.6377720135428E+01,
-7.6383700408326E+01,-7.6385785978790E+01 ])
mol_I=([-7.6332575639978E+01,
-7.6371201858219E+01,
-7.6375138534484E+01,
-7.6381335231841E+01,-7.6384945924816E+01 ])
mol_VIII= np.array([-7.6333747376680E+01,
-7.6372621206597E+01,
-7.6376414625147E+01,
-7.6382766893083E+01])
BSSE_VIII=np.array([
-7.6341711084870E+01,
-7.6376073263530E+01,
-7.6377920067006E+01,
-7.6384448801219E+01
])
bind_I=((sol_I/12)-BSSE_I)*27.2114079527


bind_VIII=((sol_VIII/8)-BSSE_VIII)*27.2114079527




fig, (ax1,ax3) = plt.subplots(1,2,figsize=(20, 5))

ax1.tick_params(axis='y')
ax1.set_xlabel('Basis set')
ax1.set_ylabel('Energy (eV)')
ax1.plot(mean_f, bind_I,'-s',label='Binding en.',color='blue')

ax2 = ax1.twinx()
ax2.tick_params(axis='y')
ax2.set_ylabel('BSSE energy (eV)')
ax2.plot(mean_f,(BSSE_I-mol_I)*27.2114079527,label='BSSE',color='black')

diff=np.random.rand(len(BSSE_I))
print('The BSSE error is (eV) :',(BSSE_I-mol_I)*27.2114079527)
for i in np.arange(1,len(BSSE_I),1):
    diff[i]=BSSE_I[i]-BSSE_I[i-1]
  #  print(BSSE_I[i]-BSSE_I[i-1])
diff[0]=0




ax3.plot(mean_f[2:],diff[2:], label='Difference',color='green')
# ax4 = ax3.twinx()
```

```
# ax4.tick_params(axis='y')
# ax4.plot(mean_f,bind_VIII,label='Binding en.')
# ax4.set_ylabel('Energy (eV)')

ax1.legend(loc='center right')
ax2.legend(loc='upper center')
# ax3.legend(loc='upper center')
# ax4.legend(loc='lower center')
ax3.set_xlabel('Basis set')


ax3.set_title('BSSE variation')
ax1.set_title('Phase Ih')
# ax3.set_title('Phase VIII')
```

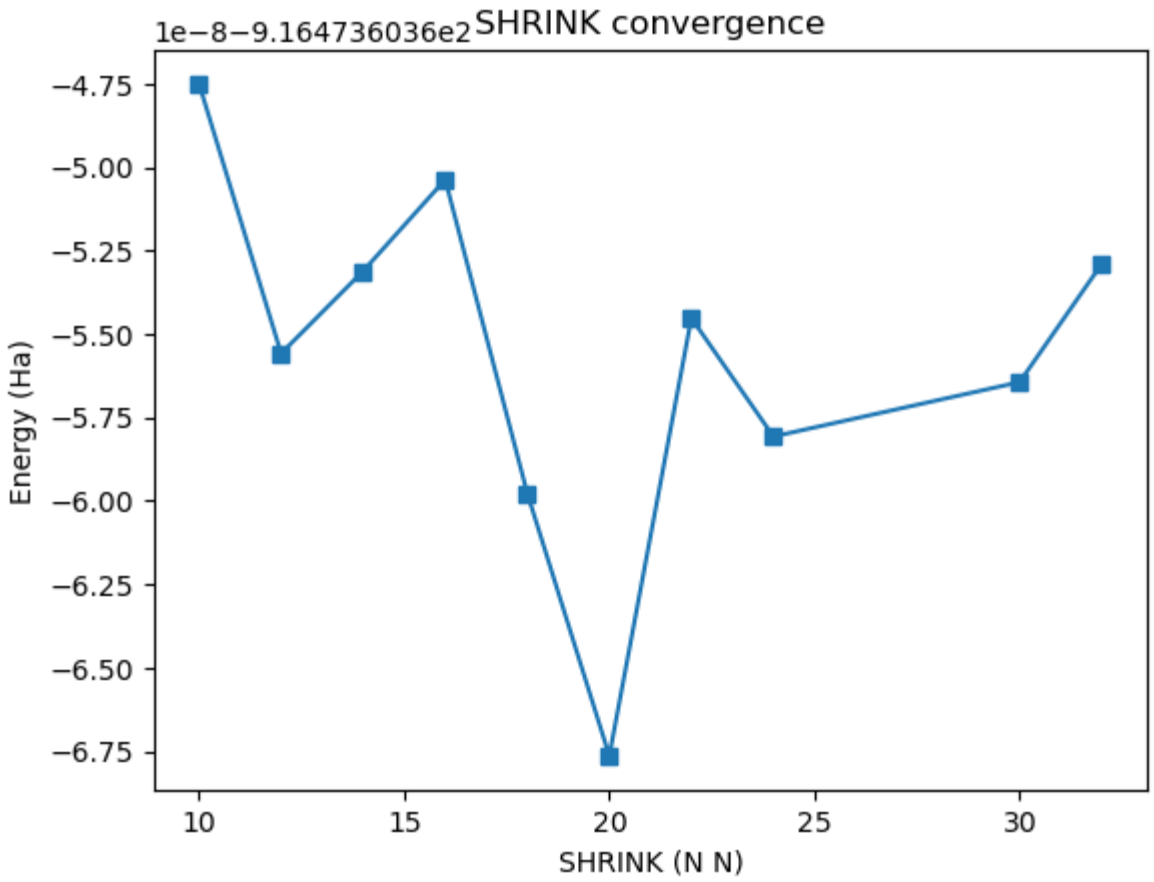
```
The BSSE error is (eV) : [-0.30133038 -0.13045217 -0.070249   -0.06435978 -0.
02285905]
```

Out[134]: Text(0.5, 1.0, 'Phase Ih')



In [137]:
```
k_points = [10, 12,14,16,18,20,22,24,30,32
]
energy= [-9.1647360364750E+02,
-9.1647360365558E+02,
-9.1647360365311E+02,
-9.1647360365037E+02,
-9.1647360365977E+02,
-9.1647360366764E+02,-9.1647360365452E+02,-9.1647360365807E+02,-9.164736036564
]
plt.plot(k_points,energy,'-s')
plt.xlabel('SHRINK (N N)')
plt.ylabel('Energy (Ha)')
```

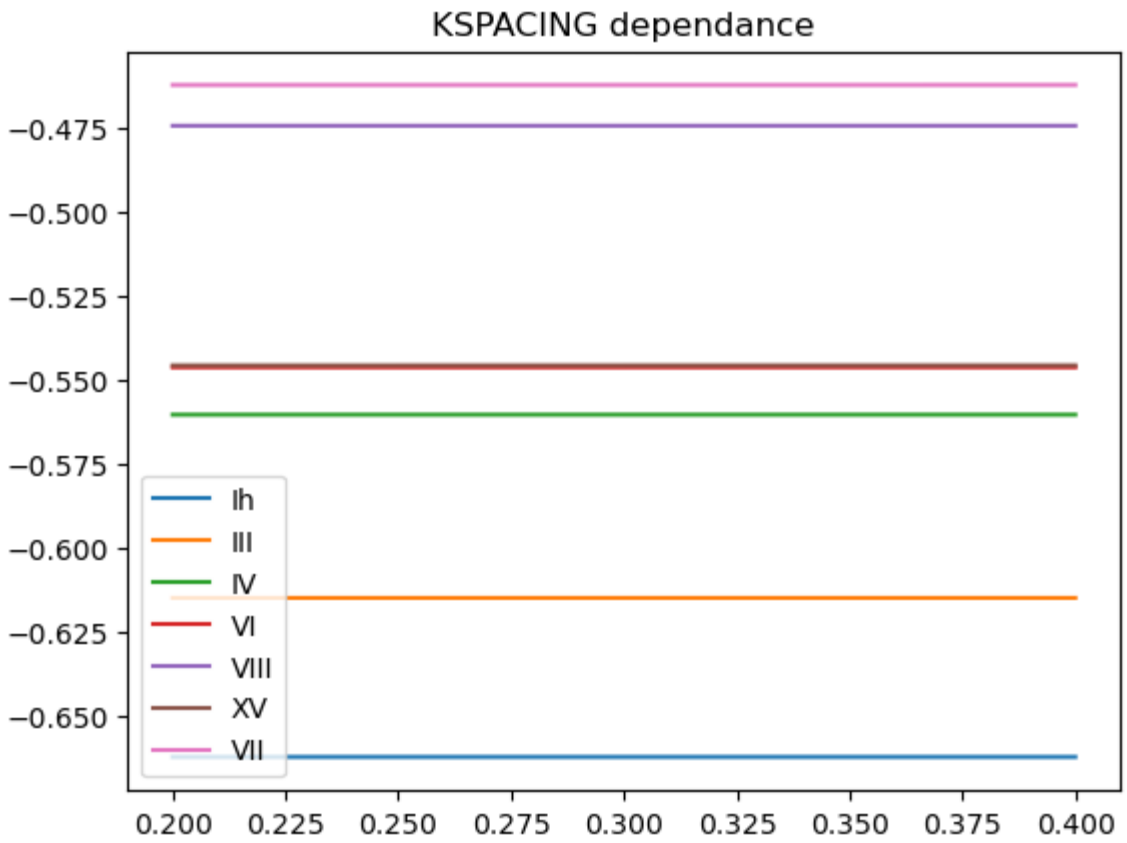
Out[137]: Text(0.5, 1.0, 'SHRINK convergence')



In [172]:
```
Ih=np.array([-178.75682125,
-178.75682449,
-178.75682120,
-178.75683367,
-178.75683363
])/12

III=np.array([-178.18902083,
-178.18902264,
-178.18902257,
-178.18900048,
-178.18900059
])/12

IV=np.array([
-236.71322976,
-236.71323121,
-236.71323212,
-236.71325878,
-236.71325864
])/16

VI=np.array([
-147.80493353,
-147.80492857,
-147.80491406,
-147.80488309,
-147.80488513
])/10

VII=np.array([
-176.35770604
,-176.35770003
,-176.35769512
,-176.35769502
,-176.35761245
])/12

VIII=np.array([-117.66879218,
-117.66879013,
-117.66878955,
-117.66876624,
-117.66876631
])/8

XV=np.array([-147.79896307,
-147.79895764,
-147.79894494,
-147.79891335,
-147.79892215,
])/10

k_space=np.arange(0.2,0.45, 0.05)
```

```
plt.plot(k_space,Ih--14.23403052,label='Ih')
plt.plot(k_space,III--14.23403052,label='III')
plt.plot(k_space,IV--14.23403052,label='IV')
plt.plot(k_space,VI--14.23403052,label='VI')
plt.plot(k_space,VIII--14.23403052,label='VIII')
plt.plot(k_space,XV--14.23403052,label='XV')
plt.plot(k_space, VII--14.23403052,label='VII')

plt.legend()
```

Out[172]: Text(0.5, 1.0, 'KSPACING dependance')



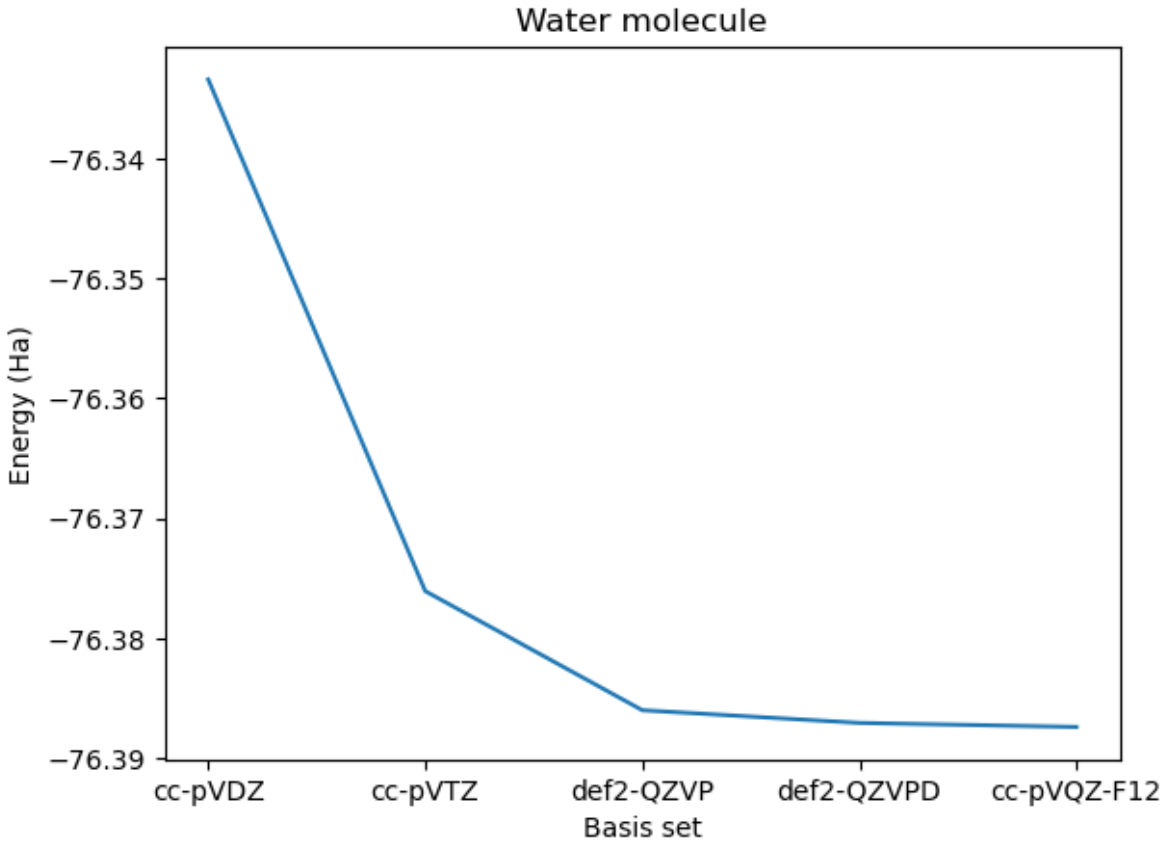
In [106]:
```
#I am checking the magnitude of variation between the differences of binding e
#(with different KSPACINGS)

#creation of a matrix with all the energies for each KSPACINGS value for each 
matrix=np.array([[-178.75682125,
-178.75682449,
-178.75682120,
-178.75683367,
-178.75683363
],[-178.18902083,
-178.18902264,
-178.18902257,
-178.18900048,
-178.18900059
],[
-236.71322976,
-236.71323121,
-236.71323212,
-236.71325878,
-236.71325864
],[
-147.80493353,
-147.80492857,
-147.80491406,
-147.80488309,
-147.80488513
],[
-176.35770604
,-176.35770003
,-176.35769512
,-176.35769502
,-176.35761245
],[-117.66879218,
-117.66879013,
-117.66878955,
-117.66876624,
-117.66876631
],[-147.79896307,
-147.79895764,
-147.79894494,
-147.79891335,
-147.79892215,
]])

# creation of the energy difference matrix between all the phases ([1][:] -> I
# [8][:] -> III-VI; ...)

diff_row=np.eye(np.sum(np.arange(matrix.shape[0])),matrix.shape[1])
c=0
matrix=matrix/(np.array([36,36,48,30,36,24,30])/3).reshape((7,1))

for r in np.arange(matrix.shape[0]):
       for i in np.arange(r+1,matrix.shape[0]):
           for j in np.arange(matrix.shape[1]):
               diff_row[c][j]=matrix[i][j]-matrix[r][j]
```

```
            c+=1
c=0

# creation of  matrix to check the magnitude of the variation of the differenc

diff_col=np.eye(diff_row.shape[0],diff_row.shape[1]-1)
for r in np.arange(diff_row.shape[0]):
            for j in np.arange(diff_row.shape[1]-1):
                diff_col[r][j]=diff_row[r][j+1]-diff_row[r][j]
print(diff_col)
```

```
[[ 1.19166669e-07 -2.68333336e-07  2.88000000e-06 -1.24999993e-08]
 [ 1.79375000e-07 -3.31041669e-07 -6.27083333e-07  5.41666623e-09]
 [ 7.66000003e-07  1.17683333e-06  4.13616667e-06 -2.07333333e-07]
 [ 7.70833335e-07  1.34999997e-07  1.04750000e-06  6.87750000e-06]
 [ 5.26250000e-07 -2.01666669e-07  3.95291667e-06 -1.20833334e-08]
 [ 8.13000003e-07  9.95833329e-07  4.19816667e-06 -8.83333335e-07]
 [ 6.02083308e-08 -6.27083327e-08 -3.50708333e-06  1.79166655e-08]
 [ 6.46833334e-07  1.44516667e-06  1.25616667e-06 -1.94833333e-07]
 [ 6.51666666e-07  4.03333333e-07 -1.83250000e-06  6.89000000e-06]
 [ 4.07083331e-07  6.66666669e-08  1.07291667e-06  4.16665813e-10]
 [ 6.93833334e-07  1.26416666e-06  1.31816667e-06 -8.70833336e-07]
 [ 5.86625003e-07  1.50787500e-06  4.76325000e-06 -2.12749999e-07]
 [ 5.91458335e-07  4.66041666e-07  1.67458333e-06  6.87208333e-06]
 [ 3.46875000e-07  1.29375000e-07  4.58000000e-06 -1.74999997e-08]
 [ 6.33625003e-07  1.32687500e-06  4.82525000e-06 -8.88750002e-07]
 [ 4.83333196e-09 -1.04183333e-06 -3.08866666e-06  7.08483333e-06]
 [-2.39750003e-07 -1.37850000e-06 -1.83249998e-07  1.95249999e-07]
 [ 4.70000003e-08 -1.81000001e-07  6.20000034e-08 -6.76000003e-07]
 [-2.44583335e-07 -3.36666666e-07  2.90541667e-06 -6.88958333e-06]
 [ 4.21666684e-08  8.60833332e-07  3.15066667e-06 -7.76083333e-06]
 [ 2.86750003e-07  1.19750000e-06  2.45250002e-07 -8.71250002e-07]]
```

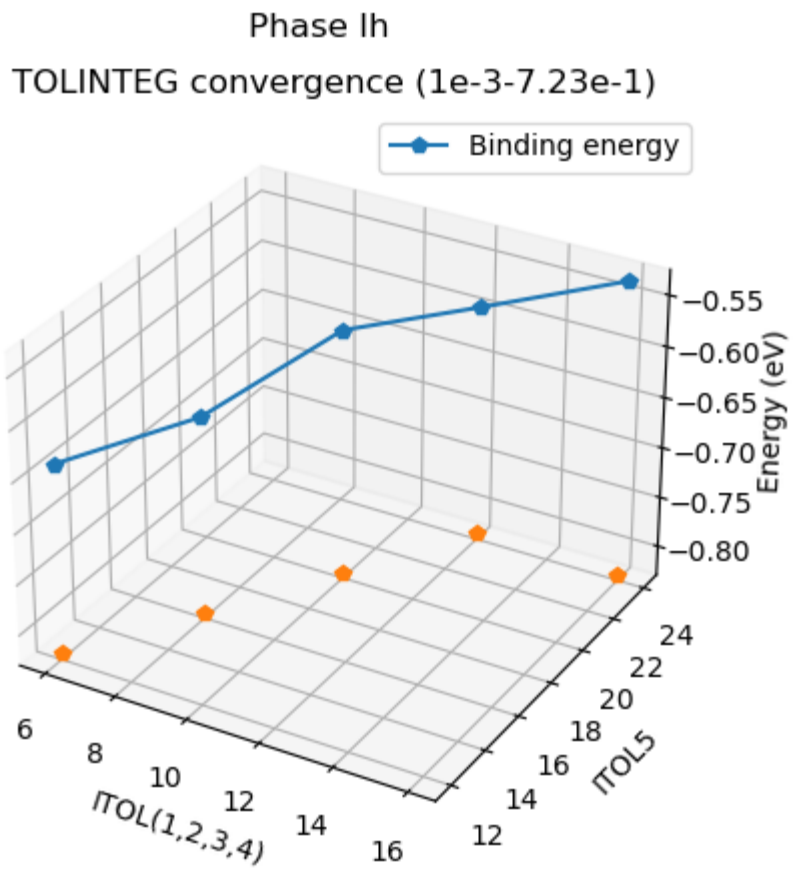
In [107]:
```
H2O=np.array([-7.6333364748786E+01,-7.6376081736518E+01,-7.6386024531398E+01,-

basis_set=['cc-pVDZ','cc-pVTZ','def2-QZVP','def2-QZVPD','cc-pVQZ-F12']
plt.plot(basis_set,H2O)
plt.title('Water molecule')
plt.xlabel('Basis set')
plt.ylabel('Energy (Ha)')
diff=np.random.rand(len(H2O))
for i in np.arange(1,len(H2O),1):
    diff[i]=H2O[i]-H2O[i-1]
diff[0]=0
print('the energy difference for each basis set with the previous one is:', di
```

```
the energy difference for each basis set with the previous one is: [ 0.
-0.04271699 -0.00994279 -0.00105856 -0.00033734]
```



In [127]:
```
fig= plt.figure()
ax=plt.axes(projection='3d')
tol=np.array([6,8,10,12,16],dtype=float)
tol5=np.array([12,16,20,24,24])
bin_ener=(np.array([-9.1683209831799E+02,-9.1683209500653E+02,-9.1683207390534

plt.plot(tol,tol5,bin_ener*1000+723,'-p',label='Binding energy')
plt.plot(tol,tol5,np.zeros(len(tol))+(-0.1+bin_ener[-1]),'p')

plt.title('TOLINTEG convergence (1e-3-7.23e-1)')
plt.suptitle('Phase Ih')
plt.xlabel('ITOL(1,2,3,4)')
plt.ylabel('ITOL5')
ax.set_zlabel('Energy (eV)')
plt.legend()
print('The binding eneries are',bin_ener)
```

```
The binding eneries are [-0.72363633 -0.72362882 -0.72358097 -0.72359387 -0.7
2353158]
```



In [47]: