# Some Defect Calculation results for $V_{Cd}$

In this example, `vac_1_Cd_0` wasn't generated with `doped`, and so the defect site is not the `doped`-generated one (0,0,0), but (0.5, 0.5, 0.5). Often `doped` can parse the defect fine, automatically determining the defect type and site from the relaxed structure.

However, sometimes it needs a little help (if the defect undergoes significant relaxation for example). This is why we have the `vasp_input.prepare_vasp_defect_dict()` function, to generate `transformation.json` files so that `doped` always knows what type of defect is present. In this case, we don't have (initially) have a `transformation.json` file because we didn't generate the defect with `doped`, but what we *can* do is get a `transformation.json` file from a different defect (that we have generated with `doped`), and modify to give the correct info for `vac_1_Cd_0` (i.e. defect charge, initial site etc.). That's what we've done here.

In [1]:

```
from doped import dope_stuff
```

In [2]:

```
from doped.pycdt.utils import parse_calculations
import numpy as np
import os
```

In [345]:

```
bulk_file_path = "Bulk_Supercell/vasp_ncl"
dielectric = np.array([[9.13, 0, 0],[0., 9.13, 0],[0, 0, 9.13]])
parsed_vac_Cd_dict = {}

for i in os.listdir():
    if 'vac_1_Cd' in i:
        print("\n",i)
        defect_file_path = f"{i}/vasp_ncl"
        defect_charge = int(i[-2:].replace("_",""))

        # Note that we've added the optional argument initial_defect_structure
to
        # SingleDefectParser.from_paths() in case PyCDT has trouble finding de
fect site
        # i.e. use the unrelaxed, PyCDT generated POSCAR in the local defect f
older
        sdp = parse_calculations.SingleDefectParser.from_paths(defect_file_pat
h, bulk_file_path,
                                                dielectric, defect_charge,
                                                initial_defect_structure=f"{i}/vas
p_gam/POSCAR")
        print("Uncorrected energy: {}".format(sdp.defect_entry))
        bo = sdp.freysoldt_loader()
        sdp.get_stdrd_metadata()

# We've also added actual_bulk optional argument to pycdt.utils.parse_calculat
ions.SingleDefectParser.get_bulk_gap_data(), in case we're
# looking at defect complexes (so the SingleDefectParser 'bulk' is actually th
e original point defect, with the wrong pmg-computed gap)
        sdp.get_bulk_gap_data(no_MP=True, actual_bulk_path=None)
# Pymatgen's bandfilling correction requires 'potalign' in defect_entry.parame
ters, which it calculates and puts there if potential
# alignment has been performed (freysoldt_loader() obviously doesn't do this f
or neutral defects), so in this case, if we think
# bandfilling's a possibility, we gotta set it ourselves
        #sdp.defect_entry.parameters['potalign'] = 0
        sdp.run_compatibility()
        print("Defect entry corrections:")
        print(sdp.defect_entry.corrections)
        print(f"Corrected energy: {sdp.defect_entry.energy} eV")
        parsed_vac_Cd_dict[i] = sdp.defect_entry # Keep dictionary of parsed d
efect entries
# Note that the PyCDT/pymatgen kumagai correction scheme uses the atomic core
potentials from the OUTCAR (which aren't printed
# if we set ICORELEVEL), so we ain't using that in this case. #itiswhatitis
```

```
 vac_1_Cd_-2
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge -2
Energy = 7.6608
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_-2/vasp_ncl
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
```

```
 [0.   0.   9.13]]
        mpid = None
Single mp-id found for bulk structure:mp-406.
WARNING: Mpid mp-406 was provided, but no bandstructure entry cu
rrently exists for it.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Mpid mp-406 was provided, but we've decided to say:
'Thanks, but no thanks' to the Materials Project.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Defect entry corrections:
{'charge_correction': 0.7376460317828045, 'bandfilling_correctio
n': -0.0, 'bandedgeshifting_correction': 0.0}
Corrected energy: 8.398484221782825 eV

 vac_1_Cd_0
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge 0
Energy = 4.6824
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_0/vasp_ncl
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
 [0.   0.   9.13]]
        mpid = None
Single mp-id found for bulk structure:mp-406.
WARNING: Mpid mp-406 was provided, but no bandstructure entry cu
rrently exists for it.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Mpid mp-406 was provided, but we've decided to say:
'Thanks, but no thanks' to the Materials Project.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Defect entry corrections:
{'bandedgeshifting_correction': 0.0}
Corrected energy: 4.682410190000013 eV

 vac_1_Cd_-1
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge -1
Energy = 6.1297
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_-1/vasp_ncl
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
 [0.   0.   9.13]]
        mpid = None
Single mp-id found for bulk structure:mp-406.
WARNING: Mpid mp-406 was provided, but no bandstructure entry cu
rrently exists for it.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Mpid mp-406 was provided, but we've decided to say:
```

```
'Thanks, but no thanks' to the Materials Project.
Reverting to use of bulk supercell calculation for band edge ext
rema.
Defect entry corrections:
{'charge_correction': 0.22517150393292082, 'bandfilling_correcti
on': -0.0, 'bandedgeshifting_correction': 0.0}
Corrected energy: 6.354858343932926 eV
```

In [62]:

```
bulk_file_path = "Bulk_Supercell/vasp_ncl"
dielectric = np.array([[9.13, 0, 0],[0., 9.13, 0],[0, 0, 9.13]])
parsed_vac_Cd_dict = {}

for i in os.listdir():
    if 'vac_1_Cd_1' in i:
        print("\n",i)
        defect_file_path = f"{i}/vasp_ncl"
        defect_charge = int(i[-2:].replace("_",""))
        sdp = parse_calculations.SingleDefectParser.from_paths(defect_file_pat
h, bulk_file_path,
                                                dielectric, defect_charge)
        print("Uncorrected energy: {}".format(sdp.defect_entry))
        bo = sdp.freysoldt_loader()
        sdp.get_stdrd_metadata()
        sdp.get_bulk_gap_data(no_MP=True, actual_bulk_path=None)
        sdp.run_compatibility()
        print("Defect entry corrections:")
        print(sdp.defect_entry.corrections)
        print(f"Corrected energy: {sdp.defect_entry.energy} eV")
        parsed_vac_Cd_dict[i] = sdp.defect_entry
```

```
 vac_1_Cd_1
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge 1
Energy = 3.2425
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_1/vasp_ncl
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
 [0.   0.   9.13]]
        mpid = None
Single mp-id found for bulk structure:mp-406.
WARNING: Mpid mp-406 was provided, but no bandstructure entry curr
ently exists for it.
Reverting to use of bulk supercell calculation for band edge extre
ma.
Mpid mp-406 was provided, but we've decided to say:
'Thanks, but no thanks' to the Materials Project.
Reverting to use of bulk supercell calculation for band edge extre
ma.
Defect entry corrections:
{'charge_correction': 0.05155697496411549, 'bandfilling_correction
': -0.18376801255382666, 'bandedgeshifting_correction': 0.0}
Corrected energy: 3.110333182410302 eV
```

In [346]:

```
parsed_vac_Cd_dict['vac_1_Cd_0'].site
```

Out[346]:

```
PeriodicSite: Cd (6.5434, 6.5434, 6.5434) [0.5000, 0.5000, 0.5000]
```

**Verifying unstable $T_d$ $V_{Cd}^{-1}$:**

In [4]:

```
from doped.pycdt.utils import parse_calculations
import numpy as np
```

In [9]:

```
bulk_file_path = "Bulk_Supercell/vasp_ncl"
dielectric = np.array([[9.13, 0, 0],[0., 9.13, 0],[0, 0, 9.13]])
vac_1_Cd_vgam_dict = {}

for i in os.listdir():
    if 'vac_1_Cd_-1' in i:
        print("\n",i)
        defect_file_path = f"{i}/vasp_gam"
        defect_charge = int(i[-2:].replace("_",""))
        sdp = parse_calculations.SingleDefectParser.from_paths(defect_file_pat
h, bulk_file_path,
                                                dielectric, defect_charge,
                                                initial_defect_structure=f"{i}/vas
p_gam/POSCAR")
        print("Uncorrected energy: {}".format(sdp.defect_entry))
        bo = sdp.freysoldt_loader()
        sdp.get_stdrd_metadata()
        sdp.get_bulk_gap_data(no_MP=True, actual_bulk_path=None)
        sdp.run_compatibility()
        print("Defect entry corrections:")
        print(sdp.defect_entry.corrections)
        print(f"Corrected energy: {sdp.defect_entry.energy} eV")
        vac_1_Cd_vgam_dict["-1_C3v"] = sdp.defect_entry

        print("\n",i)
        defect_file_path = f"{i}/Td_sym_vasp_gam"
        defect_charge = int(i[-2:].replace("_",""))
        sdp = parse_calculations.SingleDefectParser.from_paths(defect_file_pat
h, bulk_file_path,
                                                dielectric, defect_charge,
                                                initial_defect_structure=f"{i}/vas
p_gam/POSCAR")
        print("Uncorrected energy: {}".format(sdp.defect_entry))
        bo = sdp.freysoldt_loader()
        sdp.get_stdrd_metadata()
        sdp.get_bulk_gap_data(no_MP=True, actual_bulk_path=None)
        sdp.run_compatibility()
        print("Defect entry corrections:")
        print(sdp.defect_entry.corrections)
        print(f"Corrected energy: {sdp.defect_entry.energy} eV")
        vac_1_Cd_vgam_dict["-1_Td"] = sdp.defect_entry
```

```
 vac_1_Cd_-1
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge -1
Energy = 11.2396
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_-1/vasp_gam
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
 [0.   0.   9.13]]
        mpid = None
Manually fed mpid = None
WARNING: No mp-id provided, will fetch CBM/VBM details from the bu
lk calculation.
Note that it would be better to perform real band structure calcul
ation...
Defect entry corrections:
{'charge_correction': 0.23206376535223944, 'bandfilling_correction
': -0.0, 'bandedgeshifting_correction': 0.0}
Corrected energy: 11.471684855352253 eV

 vac_1_Cd_-1
No transformation file exists at vac_1_Cd_-1/Td_sym_vasp_gam/trans
formation.json.
Calculating defect index manually (proceed with caution)
Uncorrected energy: DefectEntry None - Vac_Cd_mult32 - charge -1
Energy = 11.3186
Correction = 0.0000
Parameters:
        bulk_path = Bulk_Supercell/vasp_ncl
        defect_path = vac_1_Cd_-1/Td_sym_vasp_gam
        dielectric = [[9.13 0.   0.  ]
 [0.   9.13 0.  ]
 [0.   0.   9.13]]
        mpid = None
Manually fed mpid = None
WARNING: No mp-id provided, will fetch CBM/VBM details from the bu
lk calculation.
Note that it would be better to perform real band structure calcul
ation...
Defect entry corrections:
{'charge_correction': -0.31133855462593796, 'bandfilling_correctio
n': -0.0, 'bandedgeshifting_correction': 0.0}
Corrected energy: 11.007306245374078 eV

/Users/kavanase/anaconda3/lib/python3.7/site-packages/ipykernel_la
uncher.py:33: UserWarning: Delocalization analysis has indicated t
hat Vac_Cd_mult32 with charge -1 may not be compatible with the ch
osen charge correction scheme, and may require a larger supercell
for accurate calculation of the energy. Recommended to look at the
correction plots (i.e. run `get_correction_freysoldt(DefectEntry,.
..,plot=True)` from `DefectsWithTheBoys.pycdt.corrections.finite_s
ize_charge_correction` to visually determine if charge correction
scheme still appropriate.
```

In [10]:

```
from doped.pycdt.corrections.finite_size_charge_correction import get_correcti
on_freysoldt
```

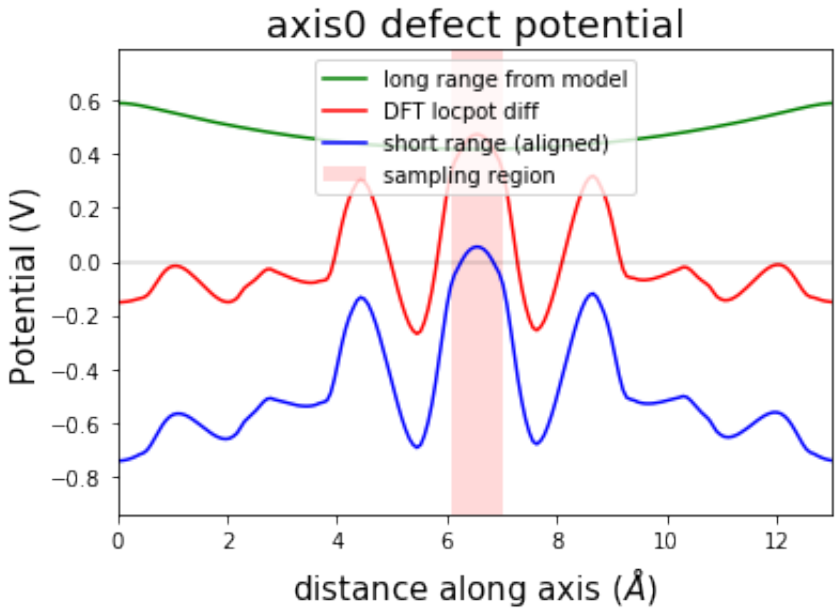
In [12]:

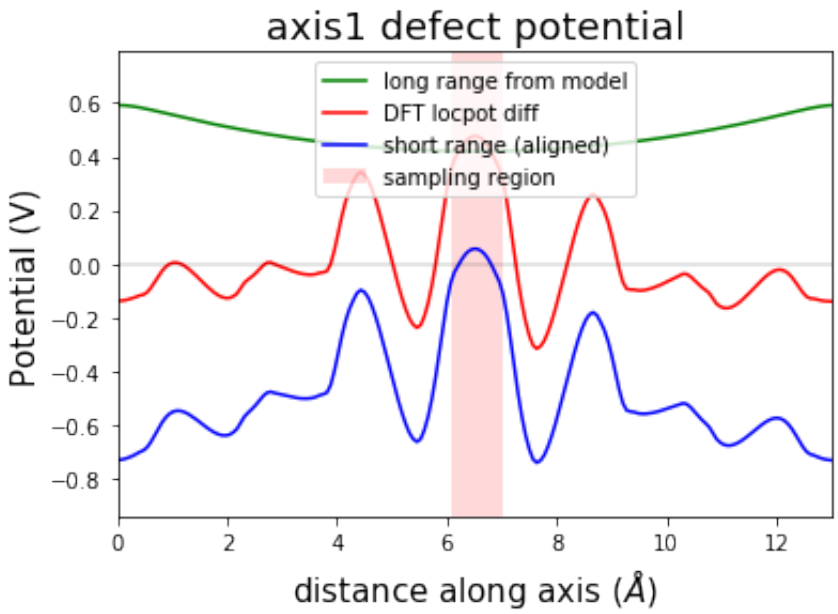
```
get_correction_freysoldt(vac_1_Cd_vgam_dict["-1_Td"], dielectric, plot = True)
```

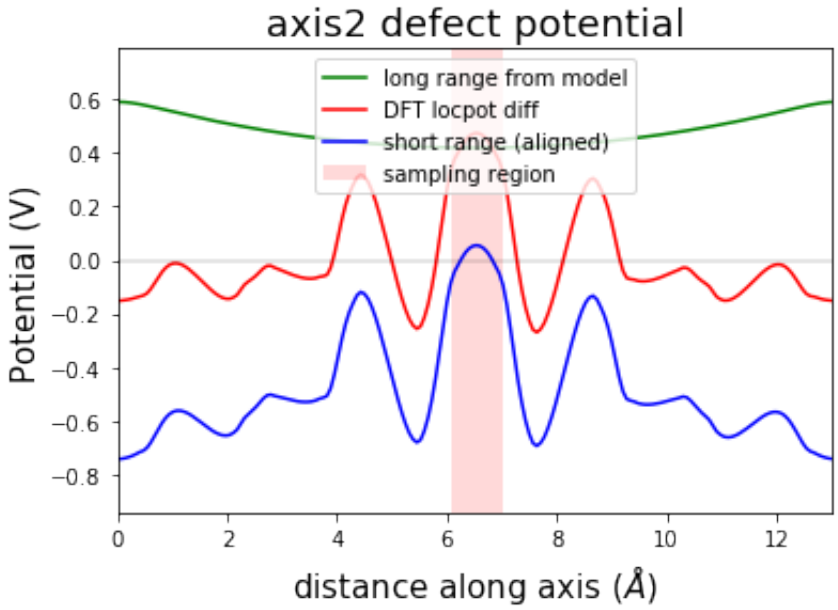
```
Final Freysoldt correction is -0.31133855462593796
```

Out[12]:

```
-0.31133855462593796
```






Yep, looks like it's unstable (by about 80 meV).

Can save python objects using `dope_stuff.save_to_pickle`, then reload them later with `dope_stuff.load_from_pickle`, so we can restart the Python kernel and not have to re-parse everything.

In [2]:

```
cdte_chempots = dope_stuff.load_from_pickle('Pickled_stuff/cdte_chempot_limits.pickle')
```

In [3]:

```
parsed_vac_Cd_dict = dope_stuff.load_from_pickle('Pickled_stuff/parsed_vac_Cd_dict.pickle')
```

In [4]:

```
vac_Cd_dpd = dope_stuff.load_from_pickle('vac_Cd_dpd.pickle')
```

In [8]:

```
parsed_CdTe_defects_dict = dope_stuff.load_from_pickle('Pickled_stuff/parsed_CdTe_defects_dict.pickle')
```

In [9]:

```
parsed_CdTe_defects_dict.update(parsed_vac_Cd_dict)
```

In [10]:

```
dope_stuff.save_to_pickle(parsed_CdTe_defects_dict, 'Pickled_stuff/parsed_CdTe_defects_dict.pickle')
```

In [ ]:

```
```

In [ ]:

```
```

## The Main Event: Transition Level Diagrams

In [110]:

```
vac_Cd_defects_dict = dope_stuff.load_from_pickle("Pickled_Shit/parsed_vac_Cd_
dict.pickle") # Frey corrected
```

In [111]:

```
lz_corrected_vac_Cd_dict = dope_stuff.lany_zunger_corrected_defect_dict_from_f
reysoldt(vac_Cd_defects_dict)
```

In [112]:

```
lz_corrected_vac_Cd_dpd = dope_stuff.dpd_from_parsed_defect_dict(lz_corrected_
vac_Cd_dict)
```

In [113]:

```
import copy
adjusted_lz_vac_Cd_dict = copy.deepcopy(lz_corrected_vac_Cd_dict)
adjusted_lz_vac_Cd_dict.pop("vac_1_Cd_1");
adjusted_lz_vac_Cd_dpd = dope_stuff.dpd_from_parsed_defect_dict(adjusted_lz_va
c_Cd_dict)
```

In [114]:

```
vac_Cd_dpd = dope_stuff.dpd_from_parsed_defect_dict(vac_Cd_defects_dict)
dope_stuff.save_to_pickle(vac_Cd_dpd, "Pickled_Shit/vac_Cd_dpd.pickle")
vac_Cd_dpd = dope_stuff.load_from_pickle("Pickled_Shit/vac_Cd_dpd.pickle")
```

In [17]:

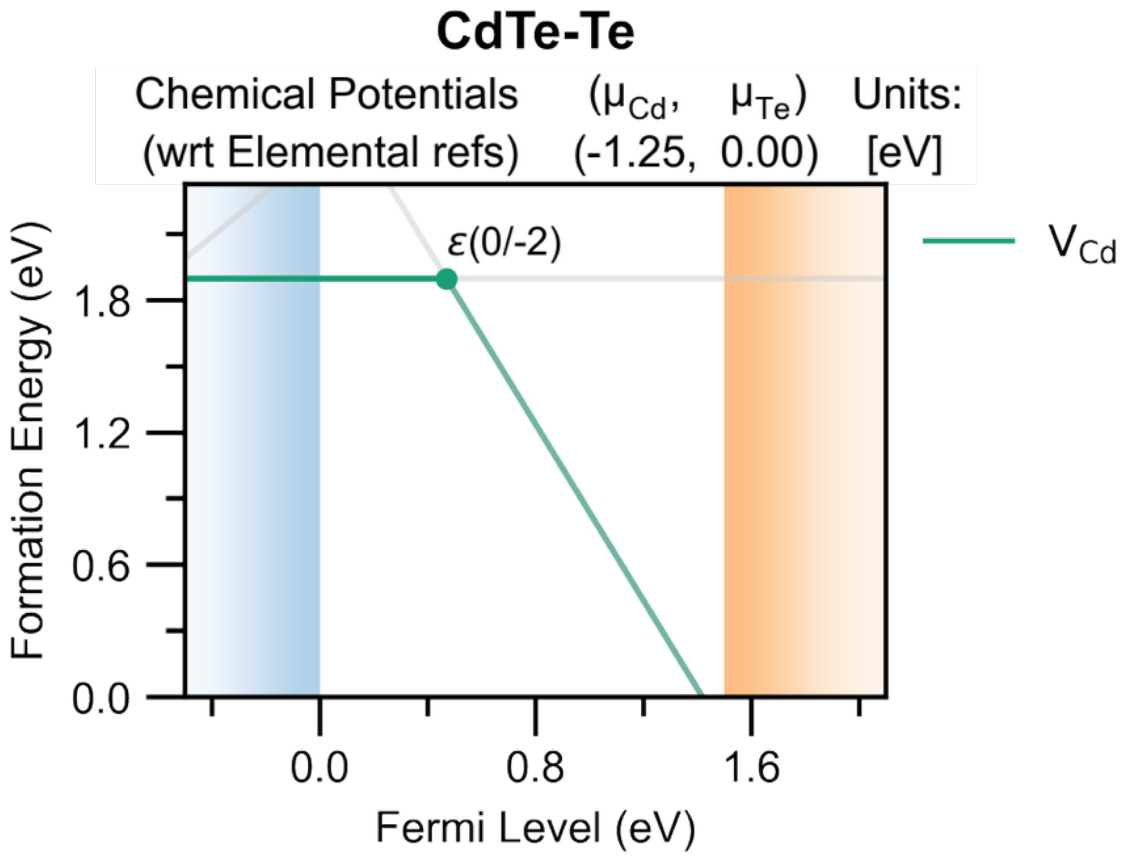
```
cdte_chempots = dope_stuff.load_from_pickle("Pickled_Shit/cdte_chempot_limits.
pickle")
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
<ipython-input-17-ac580b6cc0cf> in <module>
----> 1 cdte_chempots = dope_stuff.load_from_pickle("Pickled_Shit/
cdte_chempot_limits.pickle")

NameError: name 'dope_stuff' is not defined
```

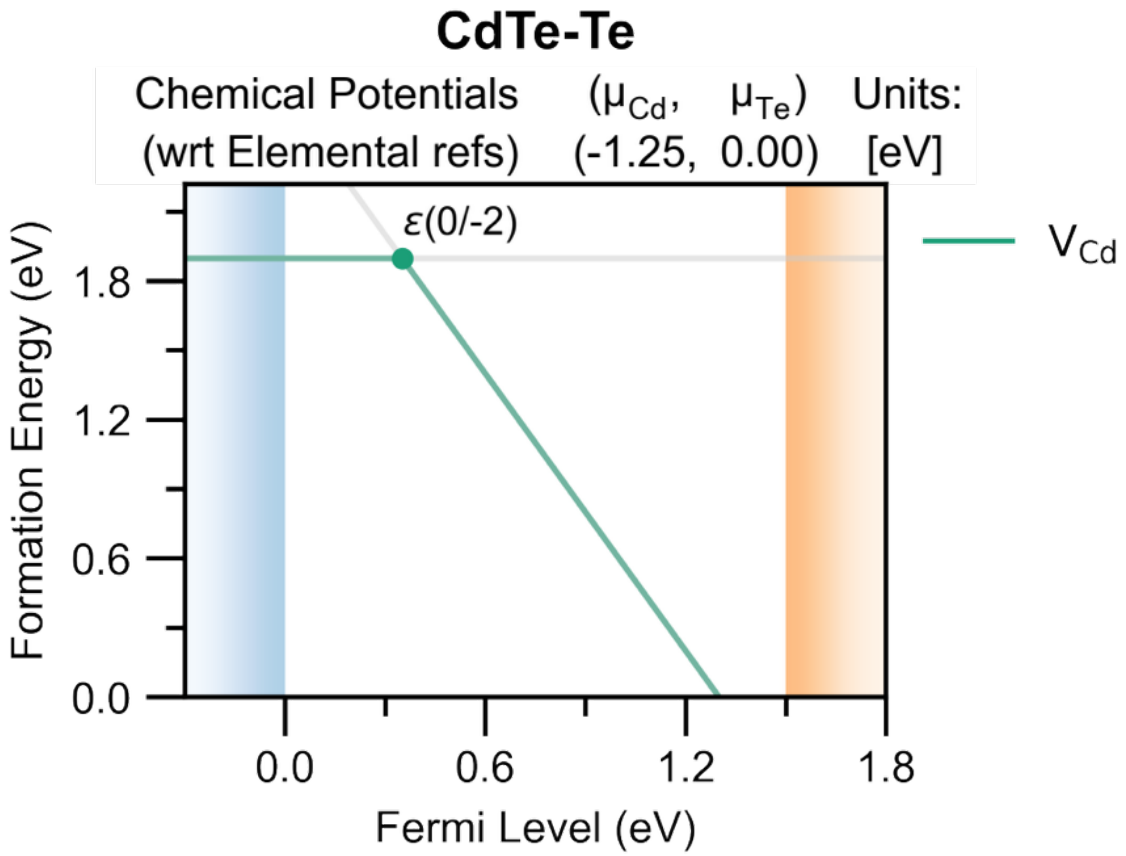
In [39]:

```
def_plot = dope_stuff.formation_energy_plot(
    vac_Cd_dpd,
    cdte_chempots,
    auto_labels=True,
    xlim=(-0.5, 2.1),
    emphasis=True,
    pd_facets=["CdTe-Te"],
    filename="V_Cd_Frey_Te-Rich.pdf"
)
```



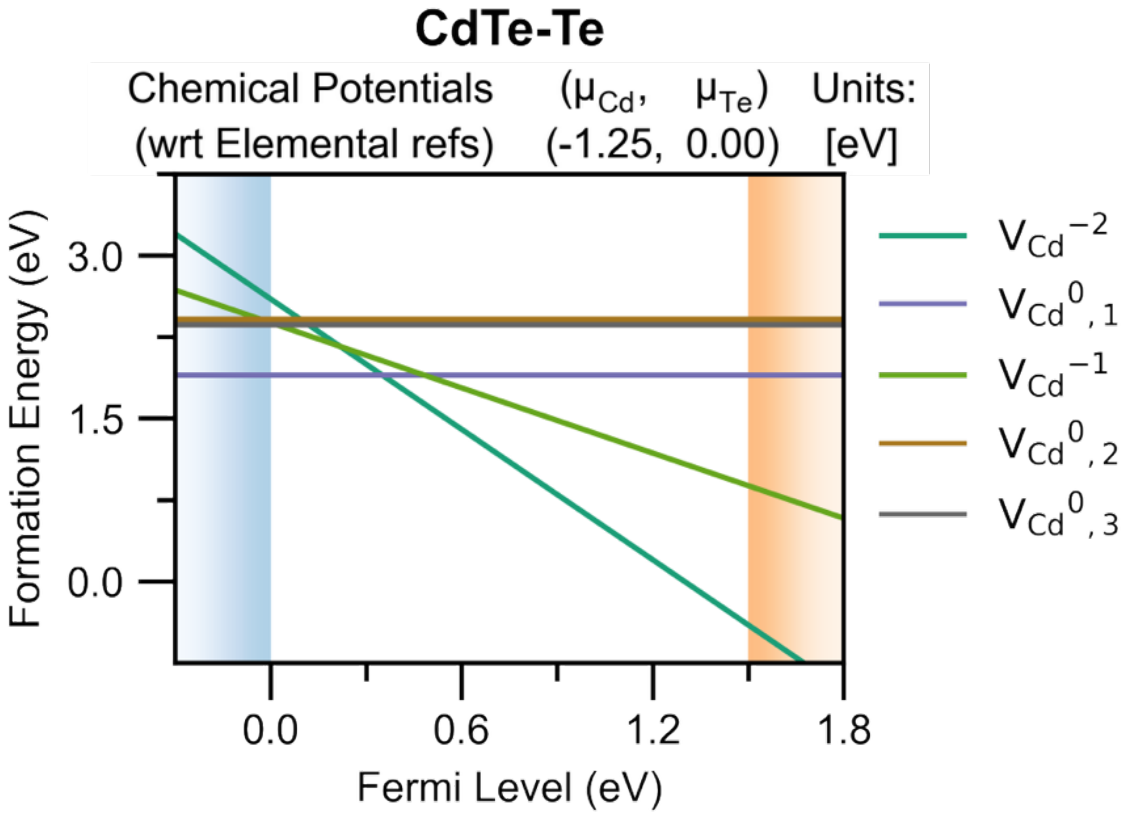
In [40]:

```
lzdef_plot = dope_stuff.formation_energy_plot(
    adjusted_lz_vac_Cd_dpd,
    cdte_chempots,
    auto_labels=True,
    xlim=(-0.3, 1.8),
    emphasis=True,
    pd_facets=["CdTe-Te"],
    filename="V_Cd_LZ_Te-Rich.pdf"
)
```



In [42]:

```
def_plot = dope_stuff.all_lines_formation_energy_plot(
    adjusted_lz_vac_Cd_dpd,
    cdte_chempots,
    auto_labels=False,
    xlim=(-0.3, 1.8),
    ylim=(-0.75,3.75),
    pd_facets=["CdTe-Te"],
    filename="V_Cd_LZ_Te-Rich_All_Lines.pdf"
)
```




# $V_{Cd}^{0}$ Concentrations

In [66]:

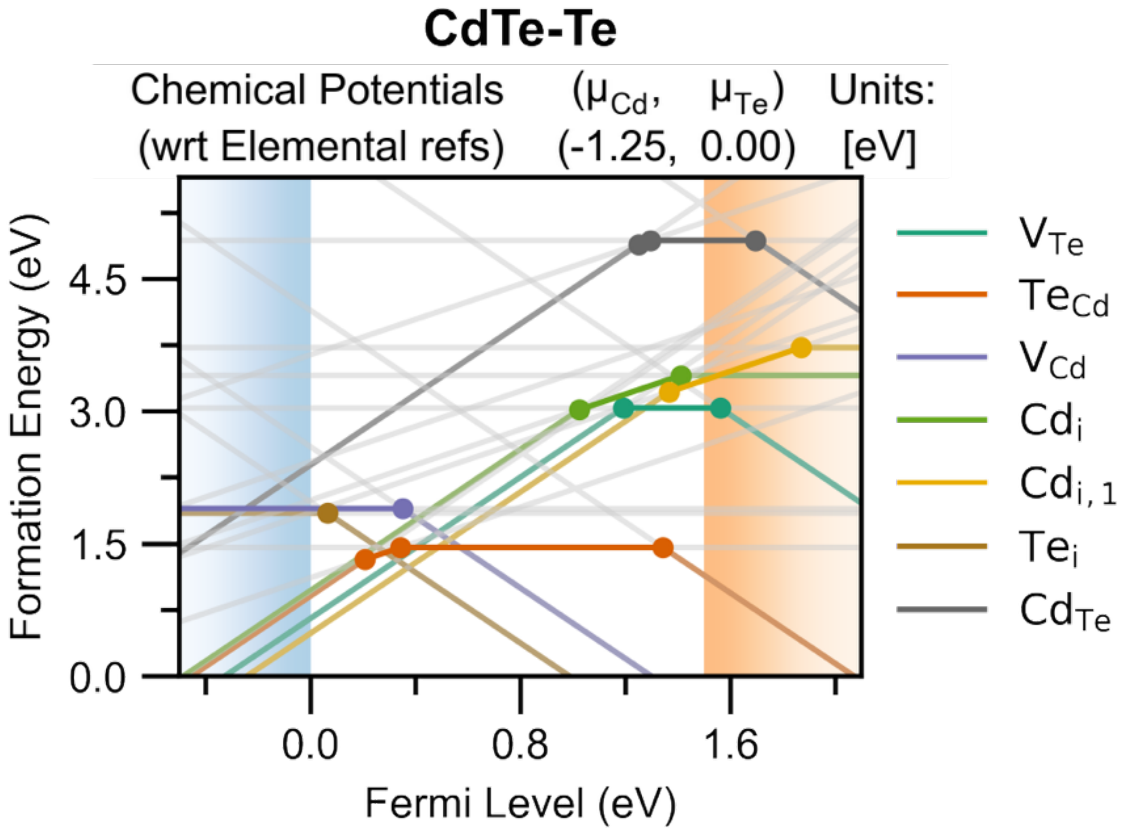
```
lz_Cd_defects_dpd = dope_stuff.load_from_pickle("Pickled_stuff/lz_CdTe_defects_dpd.pickle")
```

In [67]:

```
cdte_chempots = dope_stuff.load_from_pickle("Pickled_stuff/cdte_chempot_limits.pickle")
```

In [21]:

```
def_plot = dope_stuff.formation_energy_plot(
    lz_Cd_defects_dpd,
    cdte_chempots,
    #auto_labels=True,
    xlim=(-0.5, 2.1),
    emphasis=True,
    pd_facets=["CdTe-Te"],
)
```




In [76]:

```
[defect.energy for defect in lz_Cd_defects_dpd.entries if defect.name == "Vac_
Cd_mult32"]
```

Out[76]:

```
[8.159487929384449,
 4.165717260000008,
 6.295109020833332,
 3.050583859310708,
 4.682410190000013]
```

In [77]:

```
vac_Cd_0_sorted = sorted([defect for defect in lz_Cd_defects_dpd.entries if de
fect.name == "Vac_Cd_mult32" and defect.charge == 0], key=lambda x: x.energy)
vac_Cd_0_Te_dimer = vac_Cd_0_sorted[0]
```

In [78]:

```
def_energy = lambda x: x.energy
def_energy(vac_Cd_0_sorted[0])
```

Out[78]:

```
4.165717260000008
```

In [47]:

```
print(f"V_Cd^0 concentration in p-type CdTe at T=1000K: {vac_Cd_0_Te_dimer.def
ect_concentration(cdte_chempots['facets']['CdTe-Te'], temperature=1000, fermi_
level = 0.3):.2E} cm^3")
```

```
V_Cd^0 concentration in p-type CdTe at T=1000K: 3.86E+12 cm^3
```

In [46]:

```
print(f"V_Cd^0 formation energy in p-type CdTe: {vac_Cd_0_Te_dimer.formation_e
nergy(cdte_chempots['facets']['CdTe-Te'], fermi_level = 0.3):.3f} eV")
```

```
V_Cd^0 formation energy in p-type CdTe: 1.899 eV
```

In [ ]:

```

```

In [70]:

```
from doped.pycdt.utils.parse_calculations import get_vasprun
```

In [71]:

```
CdTe_dos_vr = get_vasprun("prim_cell_DOS/vasprun.xml.gz")
```

In [75]:

```
T = 273.15 + 750 # 750 degrees C
lz_Cd_defects_dpd.solve_for_fermi_energy(T, cdte_chempots["facets"]["CdTe-Te"]
, CdTe_dos_vr.complete_dos)
```

Out[75]:

```
0.5158090324034222
```

In [79]:

```
print(f"V_Cd^0 concentration in p-type CdTe at T=1000K: {vac_Cd_0_Te_dimer.def
ect_concentration(cdte_chempots['facets']['CdTe-Te'], temperature=T, fermi_lev
el = 0.5158):.2E} cm^3")
```

```
V_Cd^0 concentration in p-type CdTe at T=1000K: 6.35E+12 cm^3
```