GBRV12/ABINIT

GBRV 1.2 PAW dataset / ABINIT 7.10.2

`name and version of the code:` ABINIT 7.10.2
`type of basis set:` plane waves
`method:` projector-augmented wave (GBRV 1.2)

GENERAL INFORMATION

`exchange-correlation functional`	PBE
`relativistic scheme`	core and valence scalar relativistic (Koelling-Harmon)
`assignment of core / valence states`	see table (Z_{val})
`basis set size`	plane-wave cutoff energy = 100 Ry
`k-mesh density`	see table (k-point mesh in the full 1st Brillouin zone of the conventional cell *kpts*, and number of irreducible k-points # *k*)
`reciprocal-space integration method`	Fermi-Dirac smearing with a fictitious temperature corresponding to 0.01 eV

METHOD-SPECIFIC INFORMATION

`energy cutoff for the double grid`	300 Ry

ADDITIONAL COMMENTS

The ASE script used to generate these data has been included after the table.

REFERENCES

`PAW dataset`

[1] http://www.physics.rutgers.edu/gbrv/
[2] K. F. Garrity, J. W. Bennett, K. M. Rabe and D. Vanderbilt, *Comput. Mater. Sci.* **81** 446–452 (2014).

`code`

[3] X. Gonze, J.-M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.-M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami, Ph. Ghosez, J.-Y. Raty and D. C. Allan, *Comput. Mater. Sci.* **25**, 478–492 (2002).
[4] X. Gonze, B. Amadon, P.-M. Anglade, J.-M. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, Ph. Ghosez, M. Giantomassi, S. Goedecker, D. R. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M. J. T. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G.-M. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M. J. Verstraete, G. Zerah and J. W. Zwanziger, *Comput. Phys. Commun.* **180**, 2582–2615 (2009).
[5] M. Torrent, F. Jollet, F. Bottin, G. Zérah, X. Gonze, *Comput. Mater. Sci.* **42**, 337–351, (2008).

`scalar relativity`

[6] D. D. Koelling and B. N. Harmon, *J. Phys. C: Solid State* **10**, 3107–3114 (1977).

GBRV12/ABINIT

GBRV 1.2 PAW dataset / ABINIT 7.10.2

Table I. Calculation settings and results per element: valence Z_{val}, k-point mesh in the full 1st Brillouin zone of the conventional cell *kpts* and number of irreducible k-points # k, equilibrium volume per atom V_0, bulk modulus B_0, pressure derivative of the bulk modulus B_1.

	Z_{val} [–]	*kpts* [–]	# k [–]	V_0 [Å^3/atom]	B_0 [GPa]	B_1 [–]
H	1	28×28×20	6 020	17.944	10.485	2.784
He	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Li	3	38×38×38	27 436	20.235	13.871	3.345
Be	4	52×52×28	9 828	7.915	122.838	3.315
B	3	26×26×24	12 324	7.243	237.546	3.469
C	4	48×48×12	10 512	11.654	209.140	3.570
N	5	16×16×16	688	29.089	54.218	3.763
O	6	26×24×24	7 488	18.692	50.450	3.872
F	7	16×28×14	3 136	19.543	34.056	4.301
Ne	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Na	9	32×32×32	16 384	36.992	7.784	3.691
Mg	10	36×36×20	9 900	22.988	36.026	4.396
Al	3	24×24×24	364	16.488	77.769	5.002
Si	4	32×32×32	8 448	20.454	88.933	4.329
P	5	30×8×22	1 320	21.731	67.222	4.345
S	6	38×38×38	27 436	17.193	84.050	4.072
Cl	7	12×24×12	864	38.784	18.961	4.376
Ar	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
K	9	20×20×20	220	73.821	3.569	3.774
Ca	10	18×18×18	165	42.227	17.617	3.379
Sc	11	34×34×20	8 840	24.651	54.467	3.382
Ti	12	40×40×22	4 620	17.413	111.832	3.600
V	13	34×34×34	969	13.433	183.055	3.890
Cr	14	36×36×36	2 280	11.821	181.629	7.079
Mn	15	28×28×28	5 488	11.391	110.008	0.632
Fe	16	36×36×36	2 280	11.400	191.717	5.167
Co	17	46×46×24	13 248	10.838	214.939	4.748
Ni	18	28×28×28	1 120	10.860	200.929	5.103
Cu	19	28×28×28	560	11.971	141.234	5.039
Zn	20	44×44×20	14 740	15.252	74.329	5.423
Ga	19	22×12×22	1 452	20.348	48.645	5.501
Ge	14	30×30×30	6 975	23.886	58.951	4.859
As	5	30×30×10	6 825	22.704	68.719	4.323
Se	6	26×26×20	3 510	29.742	47.284	4.475
Br	7	12×24×12	864	39.609	22.386	4.861
Kr	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Rb	9	18×18×18	165	91.082	2.790	3.681
Sr	10	16×16×16	120	54.474	11.248	5.167
Y	11	32×32×18	2 448	32.842	41.367	3.154
Zr	12	36×36×20	9 900	23.380	93.932	3.276
Nb	13	30×30×30	680	18.131	170.040	3.698
Mo	14	32×32×32	816	15.785	259.439	4.326
Tc	15	42×42×22	9 702	14.423	300.194	4.525
Ru	16	42×42×24	5 544	13.745	313.485	4.873
Rh	15	26×26×26	455	14.094	256.311	5.218
Pd	16	26×26×26	455	15.292	169.102	5.480
Ag	19	24×24×24	364	17.894	89.126	6.066
Cd	12	38×38×18	9 918	22.858	45.002	7.049

In	13	30×30×20	1 200	27.534	35.843	5.086
Sn	14	26×26×26	4 563	36.705	36.249	4.909
Sb	15	26×26×8	4 108	31.701	50.933	4.528
Te	6	26×26×16	8 216	34.926	45.262	4.745
I	7	12×22×10	660	50.138	18.684	5.066
Xe	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Cs	9	16×16×16	120	116.795	1.967	3.618
Ba	10	20×20×20	220	63.391	8.772	2.089
Lu	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Hf	12	36×36×20	3 420	22.579	107.548	3.402
Ta	13	30×30×30	680	18.297	196.061	3.576
W	14	32×32×32	816	16.123	304.147	4.187
Re	15	42×42×22	5 082	14.978	365.298	4.444
Os	16	42×42×24	5 544	14.251	398.704	4.811
Ir	15	26×26×26	455	14.498	349.186	5.150
Pt	16	26×26×26	455	15.615	248.578	5.494
Au	11	24×24×24	364	17.959	138.821	5.985
Hg	12	24×24×28	1 092	29.576	7.708	9.838
Tl	13	32×32×18	7 056	31.421	26.534	5.433
Pb	14	20×20×20	220	31.969	39.691	5.552
Bi	15	26×26×8	4 108	36.948	42.229	4.639
Po	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Rn	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*

GBRV12/ABINIT

GBRV 1.2 PAW dataset / ABINIT 7.10.2

script for the Atomic Simulation Environment (ASE)
S. R. Bahn, K. W. Jacobsen, *Comput. Sci. Eng.* **4**, 56–66 (2002)
https://wiki.fysik.dtu.dk/ase/

```
import os
import sys
import time

import numpy as np

import ase.db
from ase.units import Rydberg
from ase.utils import opencew
from ase.calculators.calculator import kpts2mp
from ase.io import Trajectory
from ase.calculators.abinit import Abinit
from ase.test.tasks.dcdft import DeltaCodesDFTCollection as Collection

collection = Collection()

if len(sys.argv) == 1:
    names = collection.names
else:
    names = [sys.argv[1]]

c = ase.db.connect('dcdft_abinit_paw.db')

ecut = 80
pawecutdg = 300

kptdensity = 16.0  # this is converged
width = 0.01
ecutsm = 0.0
fband = 1.5
tolsym = 1.e-12

linspace = (0.98, 1.02, 5)  # eos numpy's linspace
linspacestr = ''.join([str(t) + 'x' for t in linspace])[:-1]

code = 'abinit' + '-' + '_c' + str(ecut) + str(pawecutdg) + '_e' + linspacestr
code = code + '_k' + str(kptdensity) + '_w' + str(width) + '_s' + str(ecutsm) +
    '_t' + str(tolsym)

for name in names:
    # save all steps in one traj file in addition to the database
    # we should only used the database c.reserve, but here
    # traj file is used as another lock ...
    fd = opencew(name + '_' + code + '.traj')
    if fd is None:
        continue
    traj = Trajectory(name + '_' + code + '.traj', 'w')
    atoms = collection[name]
    if name == 'Mn':  # fails to find the right magnetic state
        atoms.set_initial_magnetic_moments([10., 20., -10., -20.])
```

```
if name == 'Co':  # fails to find the right magnetic state
    atoms.set_initial_magnetic_moments([10., 10.])
if name == 'Ni':  # fails to find the right magnetic state
    atoms.set_initial_magnetic_moments([10., 10., 10., 10.])
cell = atoms.get_cell()
kpts = tuple(kpts2mp(atoms, kptdensity, even=True))
kwargs = {}
# loop over EOS linspace
for n, x in enumerate(np.linspace(linspace[0], linspace[1], linspace[2])):
    id = c.reserve(name=name, ecut=ecut, pawecutdg=pawecutdg,
                   linspacestr=linspacestr,
                   kptdensity=kptdensity, width=width, ecutsm=ecutsm,
                   fband=fband, tolsym=tolsym,
                   x=x)
    if id is None:
        continue
    # perform EOS step
    atoms.set_cell(cell * x, scale_atoms=True)
    # set calculator
    atoms.calc = Abinit(
        pps='paw',  # uses highest valence
        label=name + '_' + code + '_' + str(n),
        xc='PBE',
        kpts=kpts,
        ecut=ecut*Rydberg,
        pawecutdg=pawecutdg*Rydberg,
        occopt=3,
        tsmear=width,
        ecutsm=ecutsm,
        toldfe=1.0e-6,
        nstep=900,
        pawovlp=-1,  # bypass overlap check
        fband=fband,
        # http://forum.abinit.org/viewtopic.php?f=8&t=35
        chksymbreak=0,
        tolsym=tolsym,
        prtwf=0,
        prtden=0,
    )
    atoms.calc.set(**kwargs)  # remaining calc keywords
    t = time.time()
    atoms.get_potential_energy()
    c.write(atoms,
            name=name, ecut=ecut, pawecutdg=pawecutdg,
            linspacestr=linspacestr,
            kptdensity=kptdensity, width=width, ecutsm=ecutsm,
            fband=fband, tolsym=tolsym,
            x=x,
            time=time.time()-t)
    traj.write(atoms)
    wfk = name + '_' + code + '_' + str(n) + 'o_WFK'
    if os.path.exists(wfk): os.remove(wfk)
    del c[id]
```