GPAW09(pw)/ABINIT

GPAW 0.9 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 (GPAW 0.9)

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

none

REFERENCES

PAW dataset

[1] https://wiki.fysik.dtu.dk/gpaw/setups/setups.html

[2] F. Jollet, M. Torrent and N. Holzwarth, *Comput. Phys. Commun.* **185**, 1246–1254 (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).

GPAW09(pw)/ABINIT

GPAW 0.9 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	7 840	17.458	10.341	2.685
He	2	40×40×22	8 800	17.788	0.865	6.239
Li	1	38×38×38	27 436	20.256	14.042	3.381
Be	2	52×52×28	18 928	8.002	123.654	3.345
B	3	26×26×24	8 112	7.231	237.929	3.472
C	4	48×48×12	6 912	11.626	209.404	3.555
N	5	16×16×16	176	28.780	54.296	3.732
O	6	26×24×24	3 744	18.404	52.120	3.892
F	7	16×28×14	3 136	19.155	34.243	3.637
Ne	8	22×22×22	286	24.355	1.188	5.750
Na	7	32×32×32	16 384	37.065	7.701	3.691
Mg	10	36×36×20	6 480	22.898	36.138	4.069
Al	3	24×24×24	364	16.513	78.189	4.943
Si	4	32×32×32	16 384	20.523	88.382	4.324
P	5	30×8×22	1 320	21.521	68.100	4.347
S	6	38×38×38	27 436	17.236	84.023	4.073
Cl	7	12×24×12	864	38.861	19.035	4.348
Ar	8	16×16×16	120	52.281	0.764	7.317
K	9	20×20×20	220	73.581	3.582	3.263
Ca	10	18×18×18	165	42.406	17.823	3.114
Sc	11	34×34×20	5 780	24.623	54.640	3.463
Ti	12	40×40×22	8 800	17.437	112.133	3.590
V	13	34×34×34	969	13.546	181.724	3.741
Cr	6	36×36×36	1 140	11.807	163.069	7.502
Mn	15	28×28×28	2 744	11.389	113.075	2.438
Fe	8	36×36×36	1 140	11.504	190.093	5.333
Co	9	46×46×24	12 696	10.920	215.299	4.613
Ni	16	28×28×28	560	10.980	203.358	5.003
Cu	11	28×28×28	560	12.091	137.197	5.006
Zn	12	44×44×20	9 680	15.165	75.779	5.348
Ga	3	22×12×22	1 452	20.439	50.032	5.445
Ge	4	30×30×30	13 500	23.959	60.081	4.912
As	5	30×30×10	4 500	22.633	68.924	4.326
Se	6	26×26×20	6 760	29.748	47.391	4.484
Br	7	12×24×12	432	39.717	22.387	4.849
Kr	8	16×16×16	120	66.017	0.648	7.967
Rb	9	18×18×18	165	91.041	2.794	3.712
Sr	10	16×16×16	120	54.868	10.859	5.279
Y	11	32×32×18	4 608	32.860	41.314	3.161
Zr	12	36×36×20	6 480	23.395	93.592	3.236
Nb	13	30×30×30	680	18.084	169.770	3.667
Mo	14	32×32×32	816	15.797	256.468	4.277
Tc	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Ru	16	42×42×24	10 584	13.720	309.854	4.850
Rh	15	26×26×26	455	14.106	253.846	5.285
Pd	16	26×26×26	455	15.257	170.297	5.559
Ag	17	24×24×24	364	17.811	92.108	6.431
Cd	12	38×38×18	6 498	22.595	46.201	7.041

GPAW09(pw)/ABINIT

GPAW 0.9 PAW dataset / ABINIT 7.10.2

In	13	30×30×20	1 200	27.382	36.297	4.991
Sn	14	26×26×26	8 788	36.852	35.770	4.918
Sb	15	26×26×8	2 704	31.745	50.334	4.539
Te	6	26×26×16	5 408	34.843	45.097	4.717
I	7	12×22×10	330	50.547	18.595	5.079
Xe	8	14×14×14	84	86.765	0.542	7.216
Cs	9	16×16×16	120	116.931	1.965	3.482
Ba	10	20×20×20	220	63.561	8.967	2.102
Lu	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Hf	12	36×36×20	6 480	22.540	108.654	3.432
Ta	13	30×30×30	680	18.281	195.086	3.689
W	14	32×32×32	816	16.098	304.135	4.117
Re	13	42×42×22	9 702	14.934	364.161	4.401
Os	14	42×42×24	10 584	14.210	399.271	4.821
Ir	15	26×26×26	455	14.457	348.721	5.208
Pt	16	26×26×26	455	15.600	246.477	5.552
Au	11	24×24×24	364	18.178	137.671	5.898
Hg	18	24×24×28	1 092	29.547	8.559	10.946
Tl	13	32×32×18	4 608	31.159	27.221	5.427
Pb	14	20×20×20	220	31.889	40.089	5.587
Bi	15	26×26×8	2 704	36.989	42.457	4.651
Po	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Rn	8	14×14×14	84	92.788	0.544	6.684

GPAW09(pw)/ABINIT

GPAW 0.9 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]
```