GPAW09(g)/GPAW

GPAW 0.9 PAW dataset / GPAW 0.10.0 grid-based

`name and version of the code:` GPAW 0.10.0
`type of basis set:` grid-based
`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`	real-space grid spacing = 0.08 Å
`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

none

ADDITIONAL COMMENTS

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

REFERENCES

`PAW dataset`

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

`code`

[2] J. J. Mortensen, L. B. Hansen, K. W. Jacobsen, *Phys. Rev. B* **71**, 035109 (2005).
[3] J. Enkovaara, C. Rostgaard, J. J. Mortensen, J. Chen, M. Dułak, L. Ferrighi, J. Gavnholt, C. Glinsvad, V. Haikola, H. A. Hansen, H. H. Kristoffersen, M. Kuisma, A. H. Larsen, L. Lehtovaara, M. Ljungberg, O. Lopez-Acevedo, P. G. Moses, J. Ojanen, T. Olsen, V. Petzold, N. A. Romero, J. Stausholm, M. Strange, G. A. Tritsaris, M. Vanin, M. Walter, B. Hammer, H. Häkkinen, G. K. H. Madsen, R. M. Nieminen, J. K. Nørskov, M. Puska, T. T. Rantala, J. Schiøtz, K. S. Thygesen, and K. W. Jacobsen, *J. Phys.: Condens. Matter* **22**, 253202 (2010).

`scalar relativity`

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

GPAW09(g)/GPAW

GPAW 0.9 PAW dataset / GPAW 0.10.0 grid-based

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	17.462	10.329	2.743
He	2	40×40×22	8 800	17.583	0.869	4.874
Li	1	38×38×38	27 436	20.240	14.133	3.452
Be	2	52×52×28	18 928	8.008	123.420	3.367
B	3	26×26×24	8 112	7.243	237.629	3.306
C	4	48×48×12	6 912	11.648	208.928	3.570
N	5	16×16×16	176	28.820	53.862	3.438
O	6	26×24×24	3 744	18.544	51.859	4.098
F	7	16×28×14	3 136	19.248	34.060	3.942
Ne	8	22×22×22	286	26.239	1.341	−11.527
Na	7	32×32×32	16 384	37.233	7.596	3.726
Mg	10	36×36×20	6 480	22.890	36.229	4.019
Al	3	24×24×24	364	16.515	78.213	4.950
Si	4	32×32×32	16 384	20.521	88.720	4.275
P	5	30×8×22	1 320	21.528	68.008	4.475
S	6	38×38×38	27 436	17.223	82.420	3.634
Cl	7	12×24×12	864	38.957	19.009	4.331
Ar	8	16×16×16	120	52.308	0.729	11.207
K	9	20×20×20	220	73.556	3.692	3.519
Ca	10	18×18×18	165	42.402	17.543	3.633
Sc	11	34×34×20	5 780	24.576	54.970	3.340
Ti	12	40×40×22	8 800	17.430	111.636	4.185
V	13	34×34×34	969	13.541	182.380	3.743
Cr	6	36×36×36	1 140	11.851	163.951	7.354
Mn	15	28×28×28	2 744	11.476	121.042	0.651
Fe	8	36×36×36	1 140	11.480	193.333	4.672
Co	9	46×46×24	12 696	10.920	216.119	4.685
Ni	16	28×28×28	560	10.985	205.177	4.946
Cu	11	28×28×28	560	12.088	136.753	5.333
Zn	12	44×44×20	9 680	15.137	75.919	5.174
Ga	3	22×12×22	1 452	20.519	50.233	5.498
Ge	4	30×30×30	13 500	23.980	60.822	4.653
As	5	30×30×10	4 500	22.635	68.909	4.327
Se	6	26×26×20	6 760	29.745	46.787	4.714
Br	7	12×24×12	432	39.762	22.245	4.937
Kr	8	16×16×16	120	66.011	0.630	7.875
Rb	9	18×18×18	165	90.856	2.902	4.385
Sr	10	16×16×16	120	54.894	10.763	5.398
Y	11	32×32×18	4 608	32.831	41.332	3.153
Zr	12	36×36×20	6 480	23.330	93.859	3.469
Nb	13	30×30×30	680	18.047	170.104	3.770
Mo	14	32×32×32	816	15.771	257.627	4.177
Tc	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Ru	16	42×42×24	10 584	13.737	309.051	4.941
Rh	15	26×26×26	455	14.140	253.090	5.323
Pd	16	26×26×26	455	15.303	170.356	5.671
Ag	17	24×24×24	364	17.819	92.070	5.907
Cd	12	38×38×18	6 498	22.586	46.655	6.882

In	13	30×30×20	1 200	27.300	36.377	5.114
Sn	14	26×26×26	8 788	36.611	35.531	4.797
Sb	15	26×26×8	2 704	31.594	50.515	4.618
Te	6	26×26×16	5 408	34.707	45.463	4.746
I	7	12×22×10	330	50.586	18.572	5.013
Xe	8	14×14×14	84	89.606	0.566	−10.282
Cs	9	16×16×16	120	117.021	1.977	3.783
Ba	10	20×20×20	220	63.616	8.932	2.515
Lu	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Hf	12	36×36×20	6 480	22.593	108.853	3.468
Ta	13	30×30×30	680	18.305	195.375	3.704
W	14	32×32×32	816	16.119	304.069	4.135
Re	13	42×42×22	9 702	14.927	364.389	4.430
Os	14	42×42×24	10 584	14.196	399.494	4.818
Ir	15	26×26×26	455	14.453	348.437	5.056
Pt	16	26×26×26	455	15.624	245.066	5.231
Au	11	24×24×24	364	18.222	136.482	5.312
Hg	18	24×24×28	1 092	29.087	8.844	8.576
Tl	13	32×32×18	4 608	31.129	27.726	5.192
Pb	14	20×20×20	220	31.774	40.658	5.247
Bi	15	26×26×8	2 704	36.831	42.685	4.415
Po	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Rn	8	14×14×14	84	92.855	0.681	9.916

GPAW09(g)/GPAW

GPAW 0.9 PAW dataset / GPAW 0.10.0 grid-based

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.trajectory import PickleTrajectory
from ase.test.tasks.dcdft import DeltaCodesDFTCollection as Collection
from gpaw import GPAW, PW
from gpaw.mixer import Mixer
from gpaw.utilities import h2gpts

collection = Collection()

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

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

mode = 'fd'

e = 0.08  # h -> gpts

kptdensity = 16.0  # this is converged
width = 0.01

relativistic = True
constant_basis = True

if relativistic:
    linspace = (0.98, 1.02, 5)  # eos numpy's linspace
else:
    linspace = (0.92, 1.08, 7)  # eos numpy's linspace
linspacestr = ''.join([str(t) + 'x' for t in linspace])[:-1]

code = 'gpaw' + '-' + mode + str(e) + '_c' + str(constant_basis) + '_e' + linspacestr
code = code + '_k' + str(kptdensity) + '_w' + str(width)
code = code + '_r' + str(relativistic)

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 = PickleTrajectory(name + '_' + code + '.traj', 'w')
atoms = collection[name]
cell = atoms.get_cell()
kpts = tuple(kpts2mp(atoms, kptdensity, even=True))
kwargs = {}
if mode in ['fd', 'lcao']:
    if constant_basis:
        # gives more smooth EOS in fd mode
        kwargs.update({'gpts': h2gpts(e, cell)})
    else:
        kwargs.update({'h': e})
elif mode == 'pw':
    if constant_basis:
        kwargs.update({'mode': PW(e, cell=cell)})
        kwargs.update({'gpts': h2gpts(0.10, cell)})
    else:
        kwargs.update({'mode': PW(e)})
if mode == 'pw':
    if name in ['Li', 'Na']:
        # https://listserv.fysik.dtu.dk/pipermail/gpaw-developers/2012-May/
        #   002870.html
        if constant_basis:
            kwargs.update({'gpts': h2gpts(0.05, cell)})
        else:
            kwargs.update({'h': 0.05})
if mode == 'lcao':
    kwargs.update({'mode': 'lcao'})
    kwargs.update({'basis': 'dzp'})
if name in ['He', 'Ne', 'Ar', 'Kr', 'Xe', 'Rn', 'Ca', 'Sr', 'Ba', 'Be']:
    # results wrong / scf slow with minimal basis
    kwargs.update({'basis': 'dzp'})
    kwargs.update({'nbands': -5})
# loop over EOS linspace
for n, x in enumerate(np.linspace(linspace[0], linspace[1], linspace[2])):
    id = c.reserve(name=name, mode=mode, e=e, linspacestr=linspacestr,
                   kptdensity=kptdensity, width=width,
                   relativistic=relativistic,
                   constant_basis=constant_basis,
                   x=x)
    if id is None:
        continue
    # perform EOS step
    atoms.set_cell(cell * x, scale_atoms=True)
    # set calculator
    atoms.calc = GPAW(
        txt=name + '_' + code + '_' + str(n) + '.txt',
        xc='PBE',
        kpts=kpts,
        width=width,
        parallel={'band': 1},
        maxiter=777,
```

```
    idiotproof=False)
atoms.calc.set(**kwargs)  # remaining calc keywords
t = time.time()
atoms.get_potential_energy()
c.write(atoms,
        name=name, mode=mode, e=e, linspacestr=linspacestr,
        kptdensity=kptdensity, width=width,
        relativistic=relativistic,
        constant_basis=constant_basis,
        x=x,
        niter=atoms.calc.get_number_of_iterations(),
        time=time.time()-t)
traj.write(atoms)
del c[id]
```