Supplementary Information

Tutorial on Chemical Pressure Analysis: How Atomic Packing Drives Laves/Zintl Intergrowth in K_3Au_5Tl

Erdong Lu, Jonathan S. Van Buskirk, Jingxiang Cheng and Daniel C. Fredrickson *

Department of Chemistry, University of Wisconsin–Madison, 1101 University Avenue,
Madison, WI 53706, USA; elu4@wisc.edu (E.L.); jvanbuskirk@wisc.edu (J.S.V.B.);
jcheng67@wisc.edu (J.C.)
* Correspondence: danny@chem.wisc.edu

S1. Additional Computational Details and Results

Table S1. Computational parameters for ABINIT calculations.

Structure	XC Functional	Pseudopotentials	Energy Cut-off (Ha)[c]	Total Energy/atom (Ha/atom)[d]
KAu_2	LDA[a]	HGH[b]	50.0	-22.296
KTl	LDA[a]	HGH[b]	15.0	-1.19611
KAu_2 in $MgCu_2$ type	LDA[a]	HGH[b]	35.0	-22.29750
KTl in NaTl type	LDA[a]	HGH[b]	15.0	-1.20272
K_3Au_5Tl	LDA[a]	HGH[b]	35.0	-18.83456

[a] LDA parameterization of Goedecker, Teter, and Hutter.[1]
[b] Hartwigsen-Goedecker-Hutter norm-conserving pseudopotentials.[2]
[c] Energy Cut-off converges the total energy to within 3 meV per atom.
[d] Total energy with spin orbit coupling enabled.

Table S2. k-point and FFT meshes used in ABINIT calculations.

Structure[a]	K-Point Mesh[b]	K-Point Shift	FFT Grid for Calculation at V_{eq}	FFT Grid for Calculation at V_{con}	FFT Grid for Calculation at V_{exp}
KAu_2	7×7×4	0.0 0.0 0.0	96×96×144	96×96×120	96×96×216
KTl	3×3×4	0.0 0.0 0.0	180×180×60	144×144×48	192×192×60
KAu_2 in $MgCu_2$ type	5×5×5	0.0 0.0 0.0	96×96×96	60×60×60	96×96×96
KTl in NaTl type	5×5×5	0.0 0.0 0.0	72×72×72	60×60×60	96×96×96
K_3Au_5Tl	3×3×3	0.0 0.0 0.0	180×180×180	144×144×144	216×216×216

[a] All unit cells except for K_3Au_5Tl are kept as conventional unit cells, K_3Au_5Tl is converted to primitive cell.
[b] Expressed in terms of number of k points along each cell vector. The k-point meshes selected converge the total energy to within 5 meV per atom.

Table S3. Cell parameters for optimized geometries (LDA-DFT, ABINIT).

Structure	*a* (Å)	*b* (Å)	*c* (Å)	*α* (°)	*β* (°)	*γ* (°)
KAu_2	5.509	5.509	9.265	90.0	90.0	120.0
KTl	15.236	15.226	5.586	90.0	90.0	90.0
KAu_2 in $MgCu_2$ type	7.842	7.842	7.842	90.0	90.0	90.0
KTl in NaTl type	6.894	6.894	6.894	90.0	90.0	90.0
K_3Au_5Tl	5.352	18.772	8.335	90.0	90.0	90.0

Table S4. Optimized atomic coordinates for KAu_2 (LDA-DFT, ABINIT).

Element	*x*	*y*	*z*
Au	0.0000	0.0000	0.0000
Au	0.1687	0.3373	0.2500
Au	0.0000	0.0000	0.5000
Au	0.8313	0.1687	0.7500
Au	0.3373	0.1687	0.7500
Au	0.6627	0.8313	0.2500
Au	0.1687	0.8313	0.2500
Au	0.8313	0.6627	0.7500
K	0.3333	0.6667	0.9372
K	0.6667	0.3333	0.4372
K	0.6667	0.3333	0.0628
K	0.3333	0.6667	0.5628

Table S5. Optimized atomic coordinates for KTl (LDA-DFT, ABINIT).

Element	*x*	*y*	*z*
Tl	0.0000	0.0117	0.7778
Tl	0.0000	0.4883	0.2778
Tl	0.0000	0.5117	0.7222
Tl	0.0000	0.9883	0.2222
Tl	0.1113	0.1139	0.4737
Tl	0.1113	0.3861	0.9737
Tl	0.1113	0.6139	0.0263
Tl	0.1113	0.8861	0.5263
Tl	0.3887	0.1139	0.0263
Tl	0.3887	0.3861	0.5263
Tl	0.3887	0.6139	0.4737
Tl	0.3887	0.8861	0.9737
Tl	0.5000	0.0117	0.7222
Tl	0.5000	0.4883	0.2222
Tl	0.5000	0.5117	0.7778
Tl	0.5000	0.9883	0.2778
Tl	0.6113	0.1139	0.0263
Tl	0.6113	0.3861	0.5263
Tl	0.6113	0.6139	0.4737
Tl	0.6113	0.8861	0.9737

Tl	0.8887	0.1139	0.4737
Tl	0.8887	0.3861	0.9737
Tl	0.8887	0.6139	0.0263
Tl	0.8887	0.8861	0.5263
K	0.0000	0.1868	0.9578
K	0.0000	0.3132	0.4578
K	0.0000	0.6868	0.5422
K	0.0000	0.8132	0.0422
K	0.1984	0.0000	0.0000
K	0.1984	0.5000	0.5000
K	0.2500	0.2476	0.7500
K	0.2500	0.2524	0.2500
K	0.2500	0.7476	0.7500
K	0.2500	0.7524	0.2500
K	0.3016	0.0000	0.5000
K	0.3016	0.5000	0.0000
K	0.5000	0.1868	0.5422
K	0.5000	0.3132	0.0422
K	0.5000	0.6868	0.9578
K	0.5000	0.8132	0.4578
K	0.6984	0.0000	0.5000
K	0.6984	0.5000	0.0000
K	0.7500	0.2476	0.7500
K	0.7500	0.2524	0.2500
K	0.7500	0.7476	0.7500
K	0.7500	0.7524	0.2500
K	0.8016	0.0000	0.0000
K	0.8016	0.5000	0.5000

Table S6. Optimized atomic coordinates for KAu_2 in $MgCu_2$ type (LDA-DFT, ABINIT).

Element	*x*	*y*	*z*
Au	0.0000	0.0000	0.0000
Au	0.7500	0.2500	0.5000
Au	0.2500	0.5000	0.7500
Au	0.5000	0.7500	0.2500
Au	0.2500	0.7500	0.5000
Au	0.7500	0.5000	0.2500
Au	0.5000	0.2500	0.7500
Au	0.5000	0.5000	0.0000
Au	0.0000	0.7500	0.7500
Au	0.0000	0.2500	0.2500
Au	0.5000	0.0000	0.5000
Au	0.2500	0.0000	0.2500
Au	0.0000	0.5000	0.5000
Au	0.7500	0.0000	0.7500
Au	0.2500	0.2500	0.0000
Au	0.7500	0.7500	0.0000

K	0.3750	0.3750	0.3750
K	0.3750	0.8750	0.8750
K	0.8750	0.8750	0.3750
K	0.1250	0.6250	0.1250
K	0.6250	0.6250	0.6250
K	0.8750	0.3750	0.8750
K	0.1250	0.1250	0.6250
K	0.6250	0.1250	0.1250

Table S7. Optimized atomic coordinates for KTl in NaTl type (LDA-DFT, ABINIT).

Element	*x*	*y*	*z*
Tl	0.0000	0.0000	0.5000
Tl	0.0000	0.5000	0.0000
Tl	0.2500	0.2500	0.7500
Tl	0.2500	0.7500	0.2500
Tl	0.5000	0.0000	0.0000
Tl	0.5000	0.5000	0.5000
Tl	0.7500	0.2500	0.2500
Tl	0.7500	0.7500	0.7500
K	0.0000	0.0000	0.0000
K	0.0000	0.5000	0.5000
K	0.2500	0.2500	0.2500
K	0.2500	0.7500	0.7500
K	0.5000	0.0000	0.5000
K	0.5000	0.5000	0.0000
K	0.7500	0.2500	0.7500
K	0.7500	0.7500	0.2500

Table S8. Optimized atomic coordinates for K_3Au_5Tl (LDA-DFT, ABINIT).

Element	*x*	*y*	*z*
Au	0.0000	0.0000	0.0000
Au	0.1754	0.0268	0.1487
Au	0.3155	0.0000	0.8155
Au	0.3155	0.5000	0.3155
Au	0.3781	0.0268	0.3514
Au	0.5000	0.0000	0.5000
Au	0.6219	0.9732	0.6487
Au	0.6845	0.0000	0.1845
Au	0.6845	0.5000	0.6845
Au	0.8246	0.9732	0.8514
Tl	0.0660	0.8160	0.2500
Tl	0.9340	0.1840	0.7500
K	0.0324	0.6156	0.4168
K	0.3012	0.3844	0.9168
K	0.3593	0.6093	0.7500
K	0.6407	0.3907	0.2500

K	0.6988	0.6156	0.0832
K	0.9676	0.3844	0.5832

Table S9. Bader charges obtained from the LDA-DFT electron density of KAu_2 based on ABINIT calculation.

Element	Wyckoff Position	Charge
K	4*f*	0.4742
Au	2*a*	-0.2391
Au	6*h*	-0.2365

Table S10. Mulliken and Löwdin orbital occupations and charges obtained from the LOBSTER program[3-8] of KAu_2 based on VASP PAW-GGA calculation[9-14].

Element	Wyckoff Position	Orbital	*Mulliken*	*Löwdin*
K	4*f*	3s	1.96	1.95
		3p	6.00	6.00
		4s	0.57	0.71
		overall charge	0.48	0.35
Au	2*a*	6s	1.52	1.42
		5d	9.85	9.90
		overall charge	-0.39	-0.31
Au	6*h*	6s	1.39	1.29
		5d	9.81	9.83
		overall charge	-0.19	-0.13

Table S11. Bader charges obtained from the LDA-DFT electron density of KTl based on ABINIT calculation.

Element	Wyckoff Position	Charge
K	8*e*	0.4605
K	8*d*	0.4747
K	8*f*	0.4527
Tl	16*g*	-0.5681
Tl	8*f*	-0.2489

Table S12. Mulliken and Löwdin orbital occupations and charges obtained from the LOBSTER program of KTl based on VASP PAW-GGA calculation.

Element	Wyckoff Position	Orbital	*Mulliken*	*Löwdin*
K	8*e*	3s	1.97	1.95
		3p	6.00	5.97
		4s	0.44	0.48
		overall charge	0.59	0.59
K	8*d*	3s	1.98	1.95
		3p	6.00	5.97
		4s	0.29	0.40
		overall charge	0.74	0.67
K	8*f*	3s	1.98	1.95
		3p	6.00	5.97

		4s	0.33	0.41
		overall charge	0.69	0.65
Tl	16*g*	6s	1.86	1.61
		6p	1.88	2.11
		5d	9.96	9.96
		overall charge	-0.72	-0.70
Tl	8*f*	6s	1.76	1.47
		6p	1.85	2.07
		5d	9.96	9.96
		overall charge	-0.58	-0.51

Table S13. Bader charges obtained from the LDA-DFT electron density of KAu_2 in $MgCu_2$ type based on ABINIT calculation.

Element	Wyckoff Position	Charge
K	8*b*	0.4688
Au	16*c*	-0.2344

Table S14. Mulliken and Löwdin orbital occupations and charges obtained from the LOBSTER program of KAu_2 in $MgCu_2$ type based on VASP PAW-GGA calculation.

Element	Wyckoff Position	Orbital	*Mulliken* Charge	*Löwdin* Charge
K	8*b*	3s	0.05	0.05
		3p	0.00	0.00
		4s	0.30	0.19
		overall charge	0.35	0.24
Au	16*c*	6s	-0.38	-0.29
		5d	0.20	0.17
		overall charge	-0.18	-0.12

Table S15. Bader charges obtained from the LDA-DFT electron density of KTl in NaTl type based on ABINIT calculation.

Element	Wyckoff Position	Charge
K	8*a*	0.3325
Tl	8*b*	-0.3325

Table S16. Mulliken and Löwdin orbital occupations and charges obtained from the LOBSTER program of KTl in NaTl type based on VASP PAW-GGA calculation.

Element	Wyckoff Position	Orbital	*Mulliken*	*Löwdin*
K	8*a*	3s	1.95	1.93
		3p	6.00	5.97
		4s	0.30	0.38
		overall charge	0.76	0.71
Tl	8*b*	6s	1.86	1.61

		6p	1.92	2.13
		5d	9.97	9.97
		overall charge	-0.76	-0.72

Table S17. Bader charges obtained from the LDA-DFT electron density of K_3Au_5Tl based on ABINIT calculation.

Element	Wyckoff Position	Charge
K	8*h*	0.5332
K	4*e*	0.5935
Tl	4*e*	-0.1601
Au	8*h*	-0.3997
Au	8*g*	-0.2315
Au	4*a*	-0.2354

Table S18. Mulliken and Löwdin orbital occupations and charges obtained from the LOBSTER program of K_3Au_5Tl based on VASP PAW-GGA calculation.

Element	Wyckoff Position	Orbital	*Mulliken* Charge	*Löwdin* Charge
K	8*h*	3s	0.03	0.04
		3p	0.00	0.00
		4s	0.58	0.42
		overall charge	0.61	0.46
K	4*e*	3s	0.03	0.04
		3p	0.00	0.01
		4s	0.71	0.50
		overall charge	0.74	0.55
Au	8*h*	6s	-0.38	-0.18
		5d	0.19	0.17
		overall charge	-0.19	-0.01
Au	8*g*	6s	-0.40	-0.28
		5d	0.23	0.20
		overall charge	-0.17	-0.08
Au	4*a*	6s	-0.51	-0.38
		5d	0.21	0.17
		overall charge	-0.30	-0.21
Tl	4*e*	6s	0.28	0.48
		6p	-1.28	-1.60
		5d	0.03	0.02
		overall charge	-0.97	-1.10

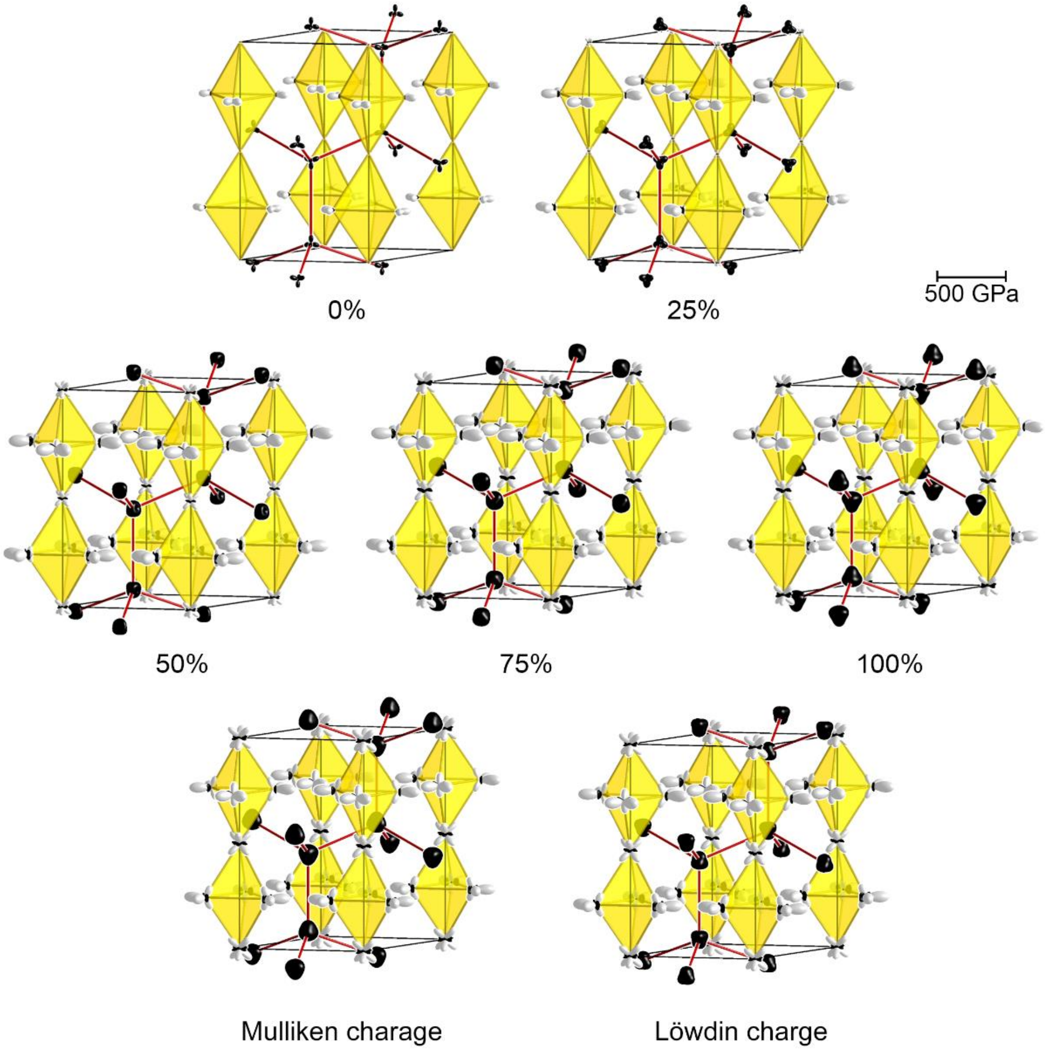


Figure S1. CP schemes for KAu_2 calculated assuming different ionicities (percentages of the Bader charges) and based on other charge definitions.

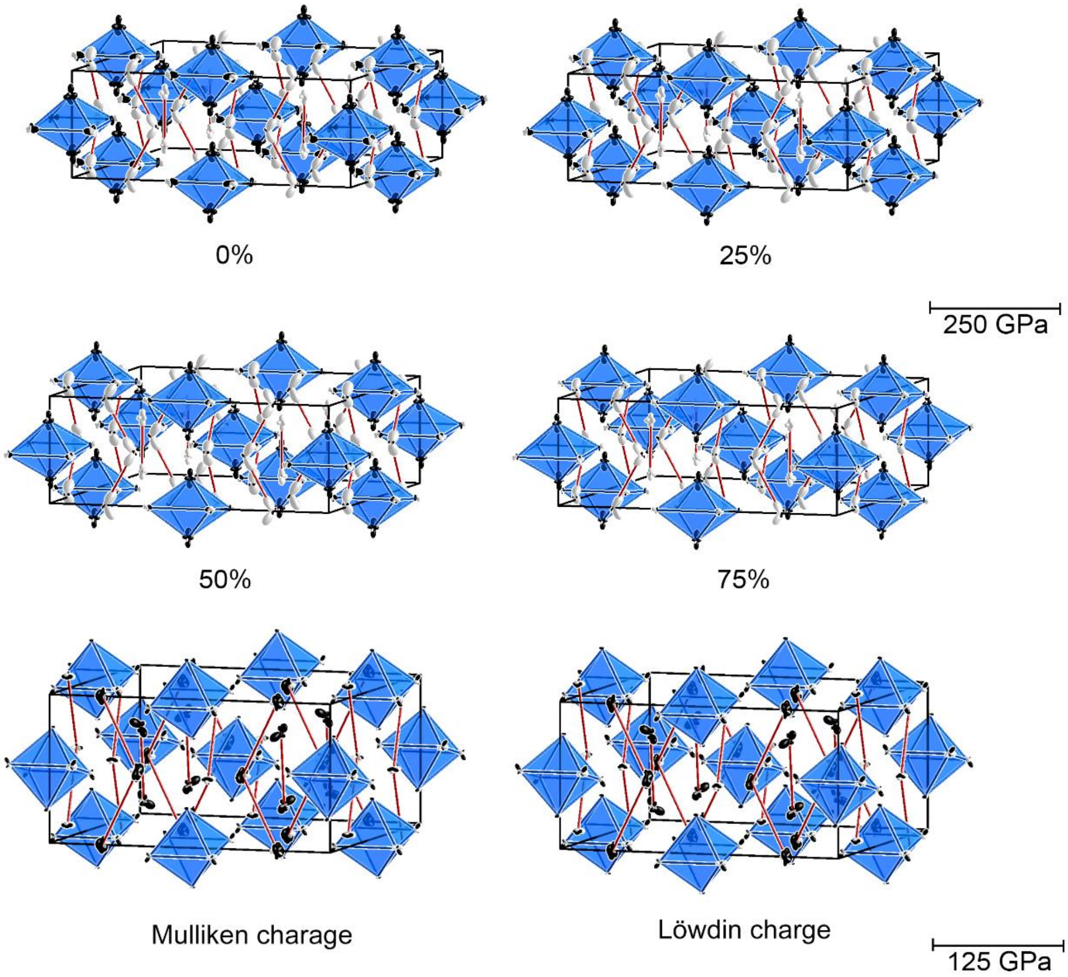
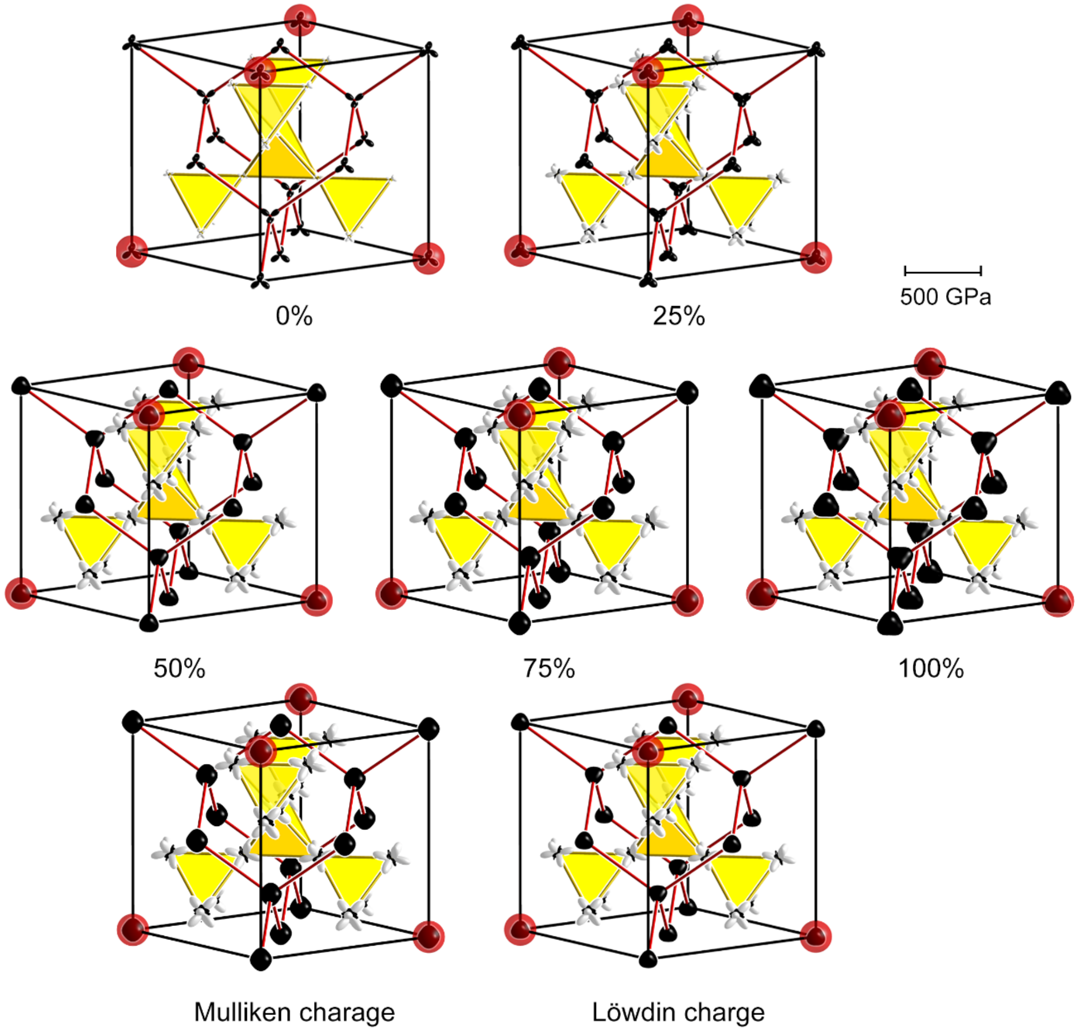


Figure S2. CP schemes for KTl at constructed assuming different ionicity levels and other charges. The 100% CP scheme was not obtainable here due to convergence issues.



0%

25%

50%

75%

100%

Mulliken charage

Löwdin charge

Figure S3. CP schemes for KAu_2 in $MgCu_2$ type generated assuming different levels of ionicity and using other charges.

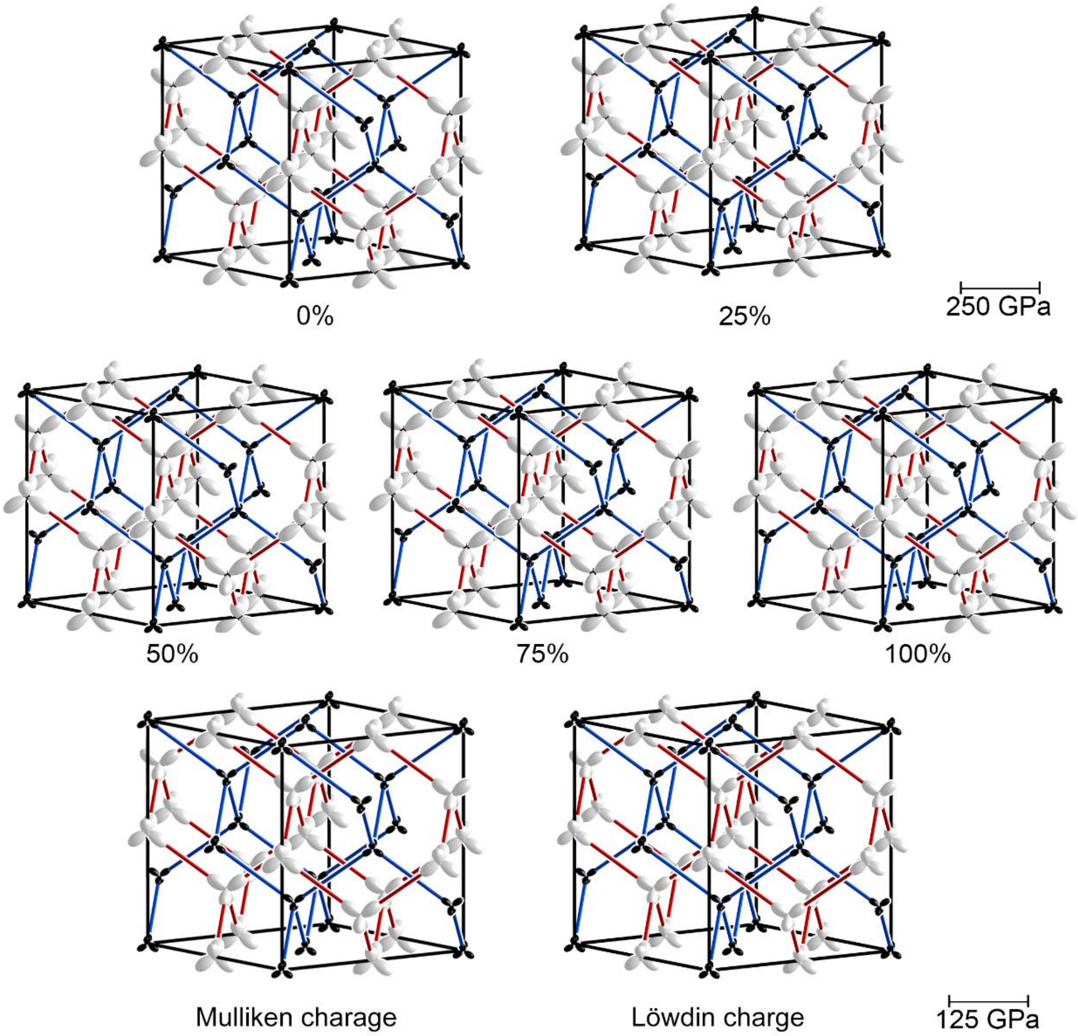
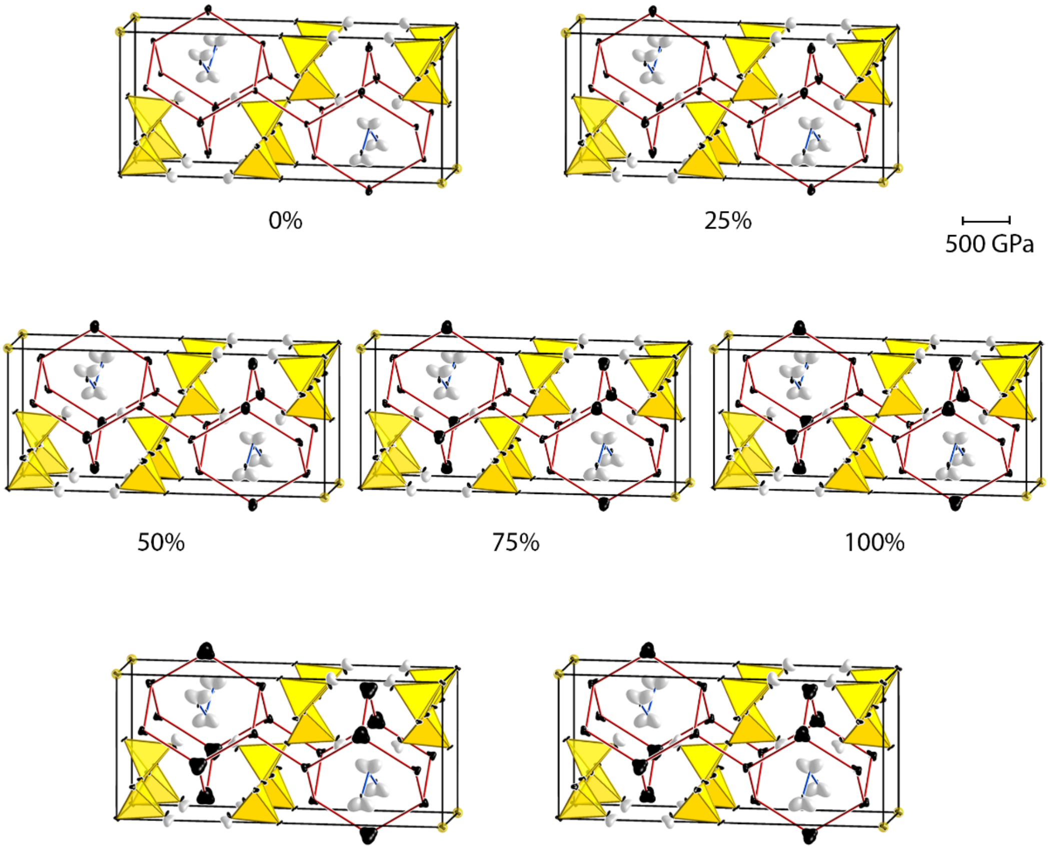


Figure S4. CP schemes for KTl in NaTl type generated assuming different levels of ionicity and using other charges.



0%

25%

50%

75%

100%

Mulliken charage

Löwdin charge

Figure S5. CP schemes for K_3Au_5Tl calculated assuming different ionicity levels and using other charges.

S2. Programs included in the DFT-CP package

qabinit-k

Language: Shell script.

Command line input: 2 inputs: ABINIT *.files* file, base name.

Required files/other inputs: CIF file, options for calculation, ABINIT *.in* file as specified by the *.files* file.

Function: Upon execution, the program checks for the existence of the *.files* file and *.in* file. If needed, it generates input files based on basename.cif for an ABINIT calculation (configured for a k-point mesh convergence test) by invoking *prepareABINIT*. If the necessary files already exist, it submits the k-point mesh convergence calculation using *qabinit*, and then calculates the convergence of each k-point mesh by with *ngkpt_convergence*.

Invoked by: None.

Invokes: *prepareABINIT*, *qabinit*, *ngkpt_convergence*.

prepareABINIT

Language: C.

Command line input: Minimum 3 input parameters: CIF file name, base name and cluster username; for certain modes, 5 input parameters: CIF file, base name, cluster username, ABINIT *.out* output file, and previous ABINIT *.in* input file.

Required files/other inputs: Options for calculation.

Function: The program generates ABINIT *.files* and *.in* files based on the selected mode of calculation from CIF file. Certain modes require previous output and input files because those calculations should be based on optimized geometries.

Invoked by: *qabinit-k*, *qabinit-opt*.

Invokes: None.

qabinit

Language: Shell script.

Command line input: 1 input: ABINIT *.files* file.

Required files/other inputs: ABINIT in file as specified by *.files* file.

Function: Submits the ABINIT calculation based on the *.files* file.

Invoked by: *qabinit-k*, *qabinit-opt*, *qabinit-ngfft*.

Invokes: None.

Credit: This script was written by Paul McGuire of the UW-Madison Department of Chemistry's High Performance Computing Center.

ngkpt_convergence

Language: C.

Command line input: 3 inputs: total energy file, ABINIT *.in* file, ABINIT *.out* output file.

Required files/other inputs: None.

Function: This calculates the energy differences between calculations involving different k-point meshes when they are specified with *ngkpt* and *shiftk*, and provides information about the level of the convergence of the total energy for the k-point meshes.

Invoked by: *qabinit-k*.

Invokes: None.

Additional comment: The total energies determined in a calculation can be extracted with the command:

```
grep etotal basename.out
```

qabinit-opt

Language: Shell script.

Command line input: 2 inputs: ABINIT *.files* file, base name.

Required files/other inputs: CIF file, options for calculation, ABINIT *.in* file as specified by the *.files* file.

Function: This program checks for the existence of the *.files* and *.in* files. If they do not exist, it will generate them based on the basename.cif file configured for two-step ABINIT geometry optimization using *prepareABINIT*. If they already exist, it will submit the optimization jobs with *qabinit*. After the geometry optimization is completed, it will process the output files to create input files for single point calculations at expanded, equilibrium, and contracted volumes with *prepareABINIT*, as described in the main manuscript.

Invoked by: None.

Invokes: *prepareABINIT*, *qabinit.*

qabinit-tryngfft

Language: Shell script.

Command line input: 4 inputs: base name, three integers specifying the ngfft parameters.

Required files/other inputs: ABINIT *.in* file as specified by the *.files* file.

Function: *qabinit-tryngfft* updates the ngfft parameters in the ABINIT *.in* file to the specified values and then submits the ABINIT job to the queue with *qabinit*. After ABINIT sets the FFT grid for the calculation, the script terminates the calculation and determines if the grid meets the certain specifications by invoking *read_NGFFT_info*. If it does, the script submits the calculation again by invoking *qabinit-runngfft*. Otherwise, the script informs the user that the FFT grid is not accepted. If 1 1 1 is entered for ngfft, the ngfft line in the *.in* file will be commented out to let ABINIT determine the FFT grids on its own.

Invoked by: None.

Invokes: *getJobID*, *updateNGFFT*, *qabinit-runngfft*, *read_NGFFT_info*.

getJobID

Language: C.

Command line input: None.

Required files/other inputs: *JobID_Info*

Function: This program reads the Job ID based on submission info stored in a file named *JobID_Info* created by *qabinit-tryngfft*.

Invoked by: *qabinit-tryngfft*.

Invokes: None.

updateNGFFT

Language: C.

Command line input: 4 inputs: ABINIT *.files* file name, 3 integers specifying the ngfft parameters.

Required files/other inputs: existing ABINIT *.in* file as specified by the *.files* file.

Function: This updates the ngfft parameters in the ABINIT .in file according to the specified values. If 1's are entered for the ngfft parameters, the ngfft line in the .in file will be commented out to let ABINIT determine the ngfft parameters automatically.

Invoked by: *qabinit-tryngfft*.

Invokes: None.

read_NGFFT_info

Language: C.

Command line input: None.

Required files/other inputs: ngfft_Info.

Function: This program reads the ngfft parameters from a file named *ngfft_Info*, which is created by *qabinit-tryngfft*, and determines if ngfft parameters meet the standards for CP analysis (identical for the three datasets and all three integers being multiples of twelve) have been adopted by the ABINIT program.

Invoked by: *qabinit-tryngfft*.

Invokes: None.

qabinit-runngfft

Language: Shell script.

Command line input: 4 inputs: base name, three integers giving the *ngfft* parameters.

Required files/other inputs: old ABINIT *.in* file as specified by the *.files* file.

Function: *qabinit-runngfft* updates the ABINIT *.in* file based on the known, accepted FFT grid settings and then submits the calculation with *qabinit-ngfft*. This step ensures the value of ngfft used by the ABINIT program indeed meets our requirement.

Invoked by: *qabinit-tryngfft*.

Invokes: *qabinit-ngfft*.

qabinit-ngfft

Language: Shell script.

Command line input: 1 input: ABINIT *.files* file.

Required files/other inputs: ABINIT *.in* file as specified by *.files* file.

Function: This script submits an ABINIT single point calculation based on the *.files* file. Overall, it has the same functionality as *qabinit*, but with the same command-line usage as *qabinit-tryngfft*.

Invoked by: *qabinit-runngfft*.

Invokes: None.

qBader

Language: C.

Command line input: 2 inputs: ABINIT *.files* file, ABINIT base name.

Required files/other inputs: ABINIT calculation output files, the *apeDEN_files* directory in the user's home directory.

Function: This program converts an ABINIT valence electron density file to the XSF format and adds core electron densities to produce an approximate all-electron density. It then runs the *Bader* program to get Bader charges and Bader atomic volumes.

Invoked by: None.

Invokes: *bin2xsf*, *mkden_all_elements*, *AddCore*, the *Bader* program of the Henkelman et al. [15-18]

bin2xsf

Language: C.

Command line input: 1 input: ABINIT density output file or binary file containing grid data created by ABINIT.

Required files/other inputs: None.

Function: This program converts binary grid data files created by ABINIT to the XSF format.

Invoked by: *qBader*.

Invokes: None.

mkden_all_elements

Language: Shell script.

Command line input: 1 input: ABINIT *.files* file.

Required files/other inputs: atomic wave function files generated by *APE* (see *mkden* below for more details).

Function: This script generates *mkden* input files (type of element, charge on each element and electron configuration) using *mk_mkden_in* and then creates neutral and core electron density profiles based on those input files with *mkden*. The generated *mkden* input files have names such as *mkden_input1*, *mkden_input2*, etc. Each element in a crystal structure will result in two different *mkden_input* files, one corresponding to the generation of a neutral electron density radial profile and another corresponding to the generation of core electron density. *mkden* is responsible for producing the actual profiles based on the input files.

Invoked by: *qBader*.

Invokes: *mk_mkden_in*, *mkden*.

mk_mkden_in

Language: C.

Command line input: 1 input: ABINIT *.files* file.

Required files/other inputs: None.

Function: Based on the information of the ABINIT *.files* file, this program generates the input file for the *mkden* program to generate the necessary electron density radial profiles. For each element, there will be two inputs, one corresponding to the neutral profile and one corresponding to the core profile. The core electron density is dependent on the pseudopotential used. For example, an Au atom modeled with a semicore pseudopotential has 11 electrons that are handled explicitly by ABINIT. The core electron input file should then contain:

79	(Au atomic number)
11	(Charge on atom)
2	(1s)
2	(2s)
6	(2p)
2	(3s)
6	(3p)
2	(4s)
10	(3d)
6	(4p)
2	(5s)
10	(4d)
6	(5p)
0	(6s)
14	(4f)

Note that the contents in the parentheses are comments not included in the actual input file.

Invoked by: *mkden_all_elements*.

Invokes: None.

mkden

Language: C.

Command line input: None.

Required files/other inputs: This program prompts for the atomic number of the element, charge on the atom of the element, and electron configuration. It also requires atomic wave function files for each orbital of the element generated by APE. By default, the database of wavefunctions should be located in the *~/apeDEN_files* directory but can be changed easily in the *mkden_all_elements* script.

Function: This program generates the electron density radial profiles for free ions based on input files provided by *mk_mkden_in*. The name of the output file has format "Element_name-charge." For example, the Au atom with 11 charges has a core density profile named as Au-11.000000.

Invoked by: *mkden_all_elements*.

Invokes: None.

AddCore

Language: C.

Command line input: 2 inputs: XSF file name without extension, ABINIT output file.

Required files/other inputs: None.

Function: *AddCore* adds core electron densities created by APE and *mkden* to the density files in the XSF file. This program writes two output files, *CHGCAR* and *CHGCAR_sum* in the VASP CHGCAR format, which serve as input for the Bader program.

Invoked by: *qBader*.

Invokes: None.

qnonloc_k

Language: Shell script.

Command line input: 3 inputs: number of k-points, ABINIT output file, ABINIT base name.

Required files/other inputs: None.

Function: This script writes a set of *qnonlocal17_by_k* input files and feeds them one by one to the *qnonlocal17_by_k* shell script to submit the nonlocal energy partition calculation for each k-point.

Invoked by: None.

Invokes: *qnonlocal17_by_k.*

qnonlocal17_by_k

Language: Shell script.

Command line input: 4 inputs: ABINIT *.out* output file, ABINIT base name, dataset number, k-point number.

Required files/other inputs: ABINIT output wave function file.

Function: This program submits to the queue the *nonlocal17* calculation of the nonlocal energy contribution from each atom at the specified k-point.

Invoked by: *qnonloc_k.*

Invokes: *nonlocal17.*

nonlocal17

Language: C.

Command line input: 4-5 inputs: ABINIT *.out* output file, ABINIT base name, dataset number, minimum band occupancy and optionally k-point number.

Required files/other inputs: ABINIT output wave function file.

Function: *nonlocal17* calculates the nonlocal energy contribution from each atom. If a k-point number is provided, then the program will focus on that single k-point. If the data files already exist, then the program will instead use that data to generate a map of the nonlocal energy in the XSF format that can be used by *CPpackage*.

Invoked by: *qnonlocal_by_k*, *generate_XSF*.

Invokes: None.

generate_XSF

Language: Shell script.

Command line input: 4 inputs: ABINIT *.out* output file, ABINIT base name, dataset number, minimum band occupancy.

Required files/other inputs: merged data from series of nonlocal energy calculations for individual k-points.

Function: This script calls *nonlocal17* to generate a map of the nonlocal energy from a merged set of data from individual k-points.

Invoked by: *qnonloc_k.*

Invokes: *nonlocal17*.

qMerge

Language: Shell script.

Command line input: 3 inputs: number of k points, ABINIT *.out* output file, ABINIT base name

Required files/other inputs: *qnonloc_k* log files.

Function: This script merges the *.log* files generated by *qnonloc_k* for individual k-points into one single *.log* file for each dataset. The shell script will then save the old *.log* files in another directory named *nonlocal_byk_log* in case they are needed again. It then generates the XSF files containing maps of the nonlocal energy from the merged log files by invoking *generate_XSF*.

Invoked by: None.

Invokes: *mergelog_byk_17*, *generate_XSF*.

mergelog_byk_17

Language: C.

Command line input: 2 inputs: number of k-points, base name

Required files/other inputs: *qnonloc_k* output log files.

Function: This program merges the output log files for nonlocal energy calculations for individual k-points into a single log file for each dataset.

Invoked by: *qMerge*.

Invokes: None.

prepareApe_shell

Language: Shell script.

Command line input: 2-3 inputs: ABINIT *.files* file, ABINIT *.out* output file, the name of the Bader *ACF.dat* file, if it has been renamed.

Required files/other inputs: None.

Function: This script prepares the input files for APE calculations by running *prepareApe*.

Invoked by: None.

Invokes: *prepareApe*.

prepareApe

Language: C.

Command line input: 2-3 inputs: ABINIT *.files* file, ABINIT *.out* output file, the name of the Bader *ACF.dat* file if it has been renamed.

Required files/other inputs: None.

Function: This program calculates the values of 25%, 50%, 75% and 100% of Bader charges based on data in the *ACF.dat* file and then creates a directory named *Bader* populated with the APE input files for each atomic site at each ionicity.

Invoked by: *prepareApe_shell*

Invokes: None.

runApe

Language: Shell script.

Command line input: None.

Required files/other inputs: The *Bader* directory generated by *prepareApe_shell* that includes the APE input files for each site at each ionicity.

Function: This performs APE calculations using the input files written by *prepareApe_shell*. If a calculation does not converge for a given free ion, the shell script will recognize that issue and move on to the next site, as well as showing a message on the screen. After all the calculations are completed, the electron density profiles are then copied after some reformatting to the main directory of the CP calculation.

Invoked by: None.

Invokes: None.

CPpackage

Language: c.

Command line input: 2 inputs: ABINIT output file, CP calculation base name.

Required files/other inputs: See the main text.

Function: This program calculates the CP scheme of a structure according to the parameters given in the basename.ini file.

Invoked by: None.

Invokes: None.

FigureToolWeb

Languages: JavaScript, HTML, CSS

Command line input: Not applicable.

Required files/other inputs: Interface control and styling files: three.js, create2.js, TrackballControls.js, and style.css.

Function: *FigureToolWeb* is a library of functions for visualizing crystal structures and theoretical data with a web browser.

Invoked by: None.

Invokes: None.

References

1. Goedecker, S.; Teter, M.; Hutter, J., Separable Dual-space Gaussian Pseudopotentials. *Phys. Rev. B* **1996,** *54*, 1703-1710.
2. Hartwigsen, C.; Goedecker, S.; Hutter, J., Relativistic Separable Dual-space Gaussian Pseudopotentials from H to Rn. *Phys. Rev. B* **1998,** *58*, 3641-3662.
3. Deringer, V. L.; Tchougréeff, A. L.; Dronskowski, R., Crystal Orbital Hamilton Population (COHP) Analysis As Projected from Plane-Wave Basis Sets. *J. Phys. Chem. A* **2011,** *115* (21), 5461-5466.
4. Dronskowski, R.; Bloechl, P. E., Crystal orbital Hamilton populations (COHP): energy-resolved visualization of chemical bonding in solids based on density-functional calculations. *J. Phys. Chem.* **1993,** *97* (33), 8617-8624.
5. Maintz, S.; Deringer, V. L.; Tchougréeff, A. L.; Dronskowski, R., Analytic projection from plane-wave and PAW wavefunctions and application to chemical-bonding analysis in solids. *J. Comput. Chem.* **2013,** *34* (29), 2557-2567.
6. Maintz, S.; Deringer, V. L.; Tchougréeff, A. L.; Dronskowski, R., LOBSTER: A tool to extract chemical bonding from plane-wave based DFT. *J. Comput. Chem.* **2016,** *37* (11), 1030-1035.
7. Maintz, S.; Esser, M.; Dronskowski, R., Efficient Rotation of Local Basis Functions Using Real Spherical Harmonics. *Acta Phys. Pol. B* **2016,** *47*, 1165.
8. Nelson, R.; Ertural, C.; George, J.; Deringer, V. L.; Hautier, G.; Dronskowski, R., LOBSTER: Local orbital projections, atomic charges, and chemical-bonding analysis from projector-augmented-wave-based density-functional theory. *J. Comput. Chem.* **2020,** *41* (21), 1931-1940.
9. Blöchl, P. E., Projector augmented-wave method. *Phys. Rev. B* **1994,** *50* (24), 17953-17979.
10. Kresse, G.; Furthmüller, J., Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Comp. Mater. Sci.* **1996,** *6* (1), 15-50.
11. Kresse, G.; Furthmüller, J., Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **1996,** *54* (16), 11169-11186.

12. Kresse, G.; Hafner, J., Ab initio molecular dynamics for liquid metals. *Phys. Rev. B* **1993,** *47* (1), 558-561.

13. Kresse, G.; Hafner, J., Ab initio molecular-dynamics simulation of the liquid-metal--amorphous-semiconductor transition in germanium. *Phys. Rev. B* **1994,** *49* (20), 14251-14269.

14. Kresse, G.; Joubert, D., From ultrasoft pseudopotentials to the projector augmented-wave method. *Phys. Rev. B* **1999,** *59* (3), 1758-1775.

15. Henkelman, G.; Arnaldsson, A.; Jónsson, H., A fast and robust algorithm for Bader decomposition of charge density. *Comput. Mater. Sci.* **2006,** *36* (3), 354-360.

16. Sanville, E.; Kenny, S. D.; Smith, R.; Henkelman, G., Improved grid-based algorithm for Bader charge allocation. *J. Comput. Chem.* **2007,** *28* (5), 899-908.

17. Tang, W.; Sanville, E.; Henkelman, G., A grid-based Bader analysis algorithm without lattice bias. *J. Phys-Condens. Mat.* **2009,** *21* (8), 084204.

18. Yu, M.; Trinkle, D. R., Accurate and efficient algorithm for Bader charge integration. *J. Chem. Phys.* **2011,** *134* (6), 064111.