Supporting information for: Investigating the Energetic Ordering of Stable and Metastable TiO_2 Polymorphs Using DFT+U and Hybrid Functionals

Matthew T. Curnan[†] and John R. Kitchin[*,‡]

Department of Materials Science and Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, and Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213

E-mail: jkitchin@andrew.cmu.edu

[*]To whom correspondence should be addressed

[†]Department of Materials Science and Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213

[‡]Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213

August 24, 2015

Contents

1 Introduction **S4**

2 Relational Schema for Database **S6**

2.1 Symmetry S7

2.2 Structure S8

2.3 Composition$N$ S8

2.4 Parameters S9

2.4.1 ParametersPosFinal S10

2.4.2 ParametersInput S10

2.4.3 ParametersResponse S11

2.5 CalcResult S12

2.5.1 CalcResultEnergetics S12

2.5.2 CalcResultLRCalc S13

2.6 FormE S14

2.6.1 FormEURange S14

2.6.2 FormEHybrid S15

2.6.3 FormEFigure S15

2.7 Metadata S15

3 Reproducing the Database **S17**

3.1 Structure table S17

3.2 Symmetry Table S18

3.3 Composition1 Table S19

3.4 Composition2 Table S19

3.5 Metadata Table S20
3.6 FormEURange Table S20
3.7 FormEHFRange Table S21
3.8 CalcResultLRCalc Table S21
3.9 CalcResultEnergetics Table S22
3.10 ParametersResponseVASP Table S23
3.11 ParametersInputVASP Table S23
3.12 ParametersInputQE Table S24
3.13 ParametersPosFinalVASP Table S25
3.14 ParametersPosFinalQE Table S27
3.15 FormEFigure Table S27
3.16 Reproducing the Calculation Input Files S28
3.16.1 Example: INCAR, KPOINTS, POTCAR Retrieval S29
3.16.2 Example: CONTCAR Retrieval S32

4 Results and Discussion **S40**
4.1 Hubbard U S40
4.1.1 Sample Input Files: Hubbard U S41
4.1.2 Plot Generation: Hubbard U S49
4.1.3 Validation of Columbite as an Upper Bound for the Metastable Range in TiO_2 . . . . . S74
4.1.4 Standard vs. Self-consistent Linear Response S91
4.2 Hybrid Functionals S128
4.2.1 Sample Input Files: Hybrid Functionals S129
4.2.2 Plot Generation: Hybrid Functionals S131
4.2.3 Structural Characterizations Relating to Energetic Calculations . . . S144
4.2.4 Energetic Comparisons: Hubbard U + Linear Response vs. Hybrid Functionals S165

1 Introduction

The Supporting Information document for the article, "Investigating the Energetic Ordering of Stable and Metastable TiO_2 Polymorphs Using DFT+U and Hybrid Functionals", provides all of the information needed to reproduce the results presented in this article. Additionally, this document will provide explanations and additional data for analyses completed in the article to substantiate several of the insights and claims made in the article itself. Both this Supporting Information document and the article explained by it were prepared using the org-mode mark-up language, which facilitates the formatting, drafting, organization, and quick PDF conversion of manuscripts prepared in LaTeX.[S1] The original source of this document can be found here: .

Sections of this Supporting Information document provide additional information regarding how the results contained within the article were achieved, illustrating how calculations yielding the results depicted were completed, detailing how results were aggregated and visualized as plotted data, and verifying the claims made by sections alluding to a supporting information document. Minimally, this document will contain information concerning the input files used to perform calculations, such that the replication of each different type of calculation performed in the Results and Discussion section of the article is possible. In addition, sections will also contain Python[S2] scripts capable of generating all plots within the article, accompanied by the numerical values of the formation energies depicted on those plots.

All of the direct and indirect allusions to a Supporting Information document in the article are addressed in this document. These allusions can be made for multiple reasons, including the inability to efficiently portray all combinations of data within the article itself or the need to more broadly contextualize results calculated for several TiO_2 polymorphs with respect to other BO_2 polymorphs (B = metal cation) covered in past research.[S3] For example, the number of combinations of different pseudopotentials, functionals, and software packages used to calculate formation energy orderings over ranges of U values cannot be effectively

portrayed in a number of static plots that could fit in the content of an article. Thus, the dynamical generation of plots featuring queried subsets of data is employed to allow users of this document the ability to superimpose data sets of their choice in individual plots, as is mentioned in Section 4.1. When assessing the effects of employing different fractions of exact exchange in hybrid functional formation energy calculations, the maximum energetic differences between a hybrid functional and either a pure PBE or HF calculation constitute an error that can be related to the energetic windows rendering the epitaxial stabilization of one polymorph on top of another. Analysis of this issue requires extensive reference to previous work[S3] and advancement upon it that requires calculations insufficient to be placed in another article on their own, thus this document includes those calculations and that analysis in Section 4.2.4. Several of these sections will also feature additional input files needed to generate necessary supporting information.

All of the results of calculations and input parameters responsible for these calculations are stored in a database consisting of relations that are largely consistent with either just Boyce-Codd Normal Form (BCNF)[S4] or both BCNF and 4th Normal Form (4NF).[S5] The information needed to generate and sort all input files and results of calculations will be stored in several files of the .csv file format, which can be imported into a wide variety of relational database management systems (RDBMS) such as MySQL[S6] or SQLite[S7] to appropriately process information. In the following section, the relational schema categorizing the data will be detailed, illustrating how the information can be recalled. To facilitate the recollection of information needed for generating plots contained within the article, relational algebraic expressions and matching SQL queries suitable for generating all of the plots contained in the article via the database are provided.

2 Relational Schema for Database

The relational hierarchy of the database detailed below can be illustrated using the class diagram generation feature of the PlantUML Java tool,[S8] representing each relation (R) as an object in the diagram, each attribute as an element listed within an object or relation, and each key as a morphism or link between two objects or relations.

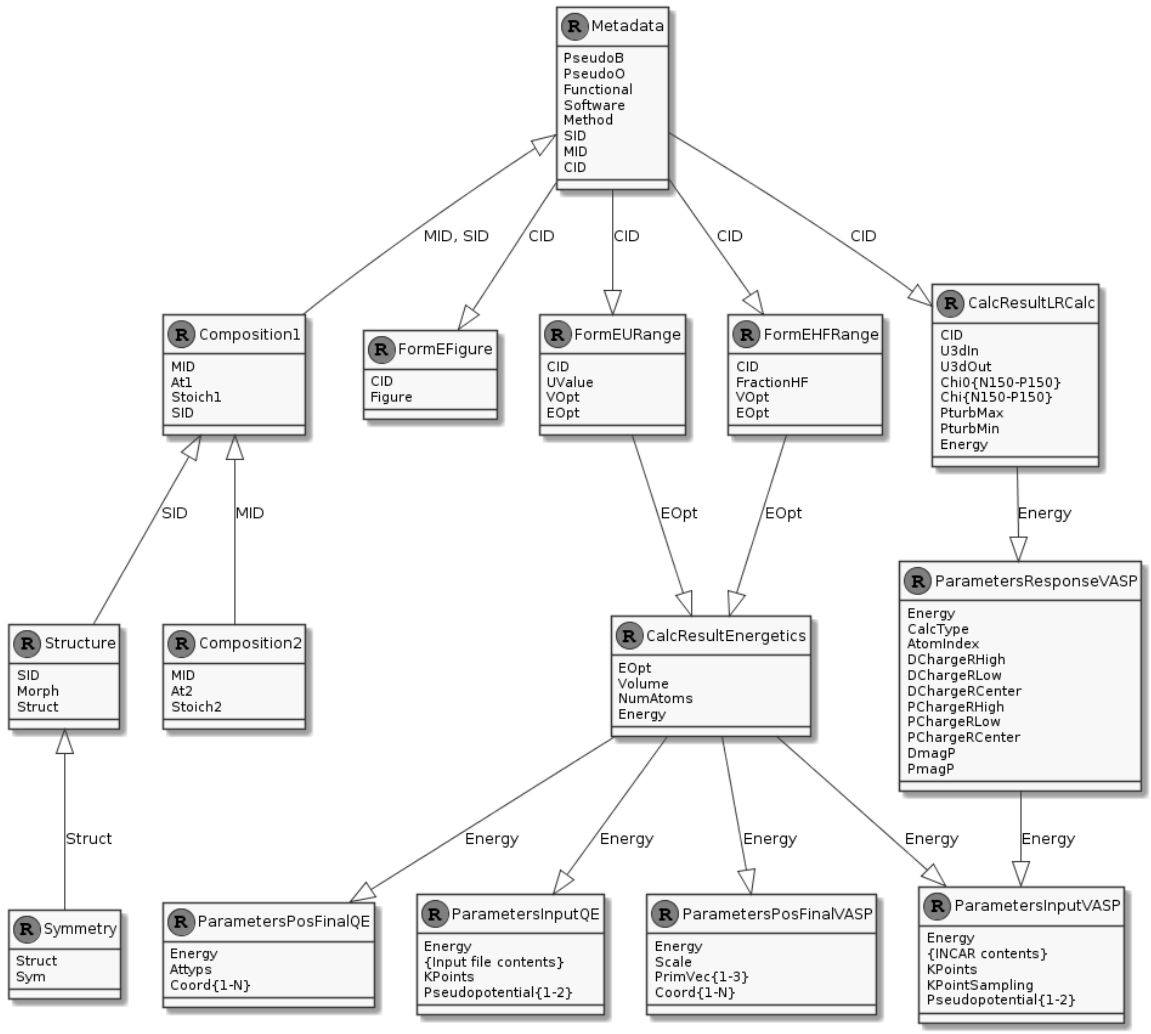

Figure S1: Visualization of the relational hierarchy of the database after division into the subrelations detailed above.

In the following sections, each feature of the data structure illustrated above is described in detail. Assembly of the relational schema dictating the structure of the database presented here proceeds from a decomposition directly corresponding to that shown in previ-

ous research,[S9] thus only the fully decomposed relations are characterized below with their constituent attributes explained. This schema, though inspired by the design principles of BCNF and 4NF, also has several attributes organized so as to ease the querying of necessary, polymorph-specific energetic and volumetric information needed to reproduce and verify presented results. Excluding the relations within this schema devoted to defining structures, this schema can also be considered a set of hierarchical depths of a tree diagram, in which each depth contains organized sets of parameters pertaining to a different step in the process of calculating results to be represented in plots within the article. Prior to reviewing these hiearchical depths, the relations within this schema relating to the definition of structures (located on the leftmost branch of the relation Metadata in Figure S1) are characterized in the following sections.

2.1 Symmetry

The Symmetry relation contains the attributes "Sym" and "Struct", the former of which is a key attribute and the latter of which relates Symmetry to Structure. The attribute "Sym", which features TEXT formatted instances, refers to the crystallographic point group of a particular periodic structure. The attribute "Struct", which also features TEXT formatted instances, refers to the Bravais lattice of a particular periodic, crystallographic structure. Values for "Sym" relating to calculations detailed in this article or its Supporting Information document sections include "P4—2/mnm", "I4—1/amd", "Pbcn", "Pbca", "Pnma", "Pa-3", "Fm-3m", and "P2—1/c". Corresponding values for "Struct" include "tetragonal", "orthorhombic", "cubic", and "monoclinic". Note that vertical bar characters (—) are used instead of underscore characters (_) to specify subscripts in symmetry groups recorded as data instances in this database, for underscore characters serve as string literals (i.e.: wildcards) in SQL-based string comparison commands (involving data instances, not attributes or relations).[S6]

2.2 Structure

The Structure relation contains the attributes ”Struct”, Morph”, and ”SID”, with the key attribute ”SID” relating Structure to Composition1. In this article, the attribute ”Morph”, which features TEXT formatted instances, refers to the structural polymorph represented by a particular system. The attribute ”SID”, which features INTEGER formatted instances, is a unique integer assigned to each lattice structure, serving as an identification of the lattice itself independent of the compositions of particular lattice sites (i.e.: a structural ID, or sID). Values for ”Morph” relating to calculations detailed in article or its Supporting Information document sections include ”Rutile”, ”Anatase”, ”Columbite”, ”Brookite”, ”Cotunnite”, ”Pyrite”, ”Fluorite”, and ”Baddeleyite”.

2.3 Composition*N*

The relation Composition1 contains the attributes ”MID”, ”At1”, ”Stoich1”, and ”SID”. The attribute ”MID”, which features INTEGER formatted instances, is a unique integer assigned to each set of atomic compositions ordered in a periodic lattice (also known as a motif identifier, or mID). Combination of the lattice and motif identifications allows identification of any periodic solid, thus in combination, ”SID” and ”MID” serve as an attribute key for the relation Composition1. These attributes are linked to the relation Metadata, which is the first relation in queries performed later that contains data explicitly involving calculations. Composition1 contains information concerning the portion of the motif represented by the first atomic type in a calculation, serving as the only information on motif directly linked to other relations in the schema. The composition of the first atomic type, namely ”At1”, features TEXT formatted instances. The attribute ”Stoich1”, which features INTEGER formatted instances, indicates the number of atoms of the first atomic type that are present in a system. Values for ”At1” relating to calculations detailed in article or its Supporting Information document sections include ”Ti”, ”V”, ”Ru”, and ”Ir”.

Each relation containing information concerning each subsequent atomic type is directly

linked to the relation directly preceding it via the key attribute ”MID”, which is shared for all atomic types represented by a single motif within a schema. Given that each system studied in this article contains two atomic types (i.e.: a metal cation, generally Ti, and O), Composition2 is the terminal relation in the structural branch of the schema, which represents information concerning the portion of the motif represented by the second atomic type in a calculation. Generally, the metal cation (usually Ti) in each calculation is represented in Composition1, while the O atom of each calculation is usually represented in Composition2. The attributes ”At2” and ”Stoich2” represent the second atomic type and the number of atoms of that type in a calculation, respectively. They possess the same formatting as corresponding instances of ”At1” and ”Stoich1”.

2.4 Parameters

The hierarchical depths within the schema not relating to the leftmost branch depicting structural information in Figure S1 are stated as prefixes in the relations constituting them, namely ”Parameters”, ”CalcResult”, ”FormE”, and the head node (relative to nodes containing results) ”Metadata”. Relations within the depth ”Parameters” describe the relaxed or final structural information of a system (PosFinal), necessary input file values needed to achieve those structures and related energetic properties (Input), or information needed to calculate response matrices (Response). These relations divide data based on the software platform employed to calculate them, using VASP or Quantum Espresso (abbreviated as QE) in this study. Relations of this form are linked to ”CalcResult” relations via the key attribute ”Energy” (contains FLOAT formatted data instances), the DFT total energy of a single calculation that serves as a unique identifier of each calculation. Unless otherwise noted, all instances contained within relations prefixed by ”Parameters” are TEXT formatted instances.

2.4.1 ParametersPosFinal

The relations ParametersPosFinalVASP and ParametersPosFinalQE store information concerning the final, relaxed structural information of a system of interest, given its software platform. For VASP calculations, this includes all information in a CONTCAR file, including primitive lattice vectors (”PrimVec1”, ”PrimVec2”, ”PrimVec3”) and the magnitude scale for those vectors (”Scale”). Additionally, the atomic coordinates of each atom are instances within the attributes ”Coord$N$”, where N represents a number between one and the total number of atoms featured in the largest system studied in this article. For QE calculations, this only includes the ”Scale” and ”Coord$N$” attributes due to differences in the formats of the input and output files presented in VASP and QE. The ”Scale” features FLOAT formatted instances, while ”PrimVec” and ”Coord$N$” attributes feature TEXT formatted instances. TEXT formatted instances are employed in this case, as atomic coordinates and primitive lattice vectors feature numbers separated by semicolons, in order to differentiate between different coordinates or indices, respectively. For both software platforms, atomic coordinates are ordered in accordance with atomic type instances, which always start with the metal cation species (usually Ti) and end with the O species. In linear response calculations, the first atomic coordinate present per system always represents the perturbed metal cation.

2.4.2 ParametersInput

The relations ParametersInputVASP and ParametersInputQE store information concerning the other input file data necessary for reproducing data in the article beyond relaxed structural information. For VASP calculations, this includes the tags associated with the use of particular pseudopotentials (”Pseudopotential1” and ”Pseudopotential2”) in the order that they are presented in a POTCAR file, the k-point sampling method (”KPointSampling”), the distribution of k-points in the x, y, and z directions (”KPoints”, respectively separated by semicolons) specified in a KPOINTS file, and the input parameters specified in an IN-

CAR file of a particular calculation. For QE calculations, this includes all parameters within the input file (ending in ".in"), including the distribution of k-points in the x, y, and z directions ("KPoints", respectively separated by semicolons) and the default names of the pseudopotentials used to calculate results ("Pseudopotential1" and "Pseudopotential2"). In all non-structural input file entries in QE and all non-structural INCAR entries in VASP, tags (VASP) or parameters (QE) are attributes matched to instances or values of data. Note that all ParametersInput relations feature TEXT formatted instances, even if the output for a specific attribute is generally numerical (including attributes relating to the distribution of KPoints).

2.4.3 ParametersResponse

The relation ParametersResponseVASP stores information concerning the outputted data received from calculations involving the linear response approach. Each atom, which is classified by its atomic index (INTEGER formatting, attribute "AtomIndex"), is mapped to its d-orbital ("DCharge") and p-orbital ("PCharge") decomposed charge occupation data, which is stored under the FLOAT formatted attribute sets "DChargeRHigh", "DChargeRLow", "DChargeRCenter" and "PChargeRHigh", "PChargeRLow", "PChargeRCenter", respectively. Attributes containing charge occupation data labelled with the suffix "RHigh" denote the occupation values produced by the highest applied perturbation levels per system ($\alpha = 0.15$ in this study), while those labelled "RLow" denote the corresponding occupation values produced by the lowest applied perturbations ($\alpha =$ -0.15 in this study). The "RCenter" suffix indicates occupation data at which the value of perturbation equals zero and the chi and chi_0 responses of a system are expected to intersect. Corresponding magnetic moment data for each atom is stored under FLOAT formatted attributes "DMagP" and "PMagP", respectively. Note that, for cases in which calculations are not spin polarized, an effective magnetization value of '0.000' is placed in each instance formed by the intersection of both "DMagP" or "PMagP" and "AtomIndex". The type of calculation contained by a particu-

lar tuple of data, which can represent either the initial or bare response ("Chi0") or final response ("Chi") used to yield a first-principles U value, is denoted by the TEXT formatted attribute "CalcType". In the case of PM TiO_2 Rutile calculations completed at $U = 0$ using standard pseudopotentials (Ti and O) and the PBE functional, note that "CalcType" instances can have a "NS-NS", "NS-S", "S-NS", and "S-S" appended after the "Chi" or "Chi0" labels. As shown in Subsection 4.1.4 later, these suffixes indicate the application or absence of spin polarization to different steps in the calculation of linear response U values. Unlike several other Parameters relations, which feature solely "Energy" as a key attribute, ParametersResponseVASP has both "Energy" and "CalcType" as key attributes.

2.5 CalcResult

Relations within the depth "CalcResult" represent individual calculation results achieved from the input information described in "Parameters". These relations contain data based on the types of calculation results placed within them, namely formation energy calculations ("Energetics" suffix) primarily applied to plots depicting incremental change of U parameters (Hubbard U calculations), incremental change of exact exchange fractions (hybrid functional calculations), or linear response calculations ("LRCalc" suffix) applied to determine the first-principles values of U for particular systems. Relations containing this content and possessing the "Energetics" suffix are linked to "FormE" relations via the key attribute "EOpt", the Birch-Murnaghan fitted[S10] total energy of a set of calculations completed for a system over displacement of a structural feature (e.g.: cell volume). Relations containing matching content that possess the "LRCalc" suffix are linked to Section 2.7 via the key attribute "CID" (both "Metadata" and "CID" are to be explained later).

2.5.1 CalcResultEnergetics

The relation CalcResultEnergetics contains the attributes "EOpt", "Energy", "Volume", and "NumAtoms". Attributes serving as keys to link one relation to another, namely "Energy"

and ”EOpt”, represent the total, final DFT energy associated with a calculation and the fitted energy associated with a set of calculations of a shared system performed over a structural displacement, respectively. The attribute ”Volume” represents the total relaxed system volume associated with a single calculation, whereas the attribute ”NumAtoms” represents the number of atoms within that calculation. Note that all of these attributes feature FLOAT formatted instances.

2.5.2 CalcResultLRCalc

The relation CalcResultLRCalc contains the attributes ”CID”, ”U3dIn”, ”U3dOut”, ”Energy”, ”PturbMin”, ”PturbMax”, Chi0*N150-P150* (or explicitly, ”Chi0N150”, ”Chi0N100”, ”Chi0N050”, ”Chi0P000”, ”Chi0P050”, ”Chi0P100”, ”Chi0P150”), and Chi*N150-P150* (or explicitly, ”ChiN150”, ”ChiN100”, ”ChiN050”, ”ChiP000”, ”ChiP050”, ”ChiP100”, ”ChiP150”). Attributes serving as keys to link one relation to another, namely ”Energy” and ”CID”, represent the total, final DFT energy associated with a calculation and the unique identifiers of particular plotted data points in Section 2.7 (to be explained later), respectively. The attribute ”U3dOut” represents the first-principles resolved value of U calculated using linear response theory at the GGA+U ground state of a system without considering off-diagonal terms, whereas ”U3dIn” represents the values of U serving as inputs in self-consistent U calculations (standard linear response always features a value of ”U3dIn” = 0). Self-consistent U values calculated at the GGA ground state are calculated within this document from queried data. Attributes with the prefix ”Chi0” represent initial values of the $3d$ orbital occupations (Ti, first atom) upon linear perturbation, while attributes with the prefix ”Chi” represent the final values of these perturbations after SCF electronic convergence. The suffixes of ”Chi0” and ”Chi” attributes represent the magnitudes of these perturbations, though the magnitudes can be discerned only after replacing instances of ”N” (within the suffix) with a negative sign or ”P” (within the suffix) with a positive sign and then dividing the number by a factor of 100. Attributes ”PturbMax” and ”PturbMin” represent the α values associated

with the maximum (”RHigh”) and minimum (”RLow”) values of the perturbations completed for each system. Note that, excluding ”CID”, all of these attributes feature FLOAT formatted instances.

2.6 FormE

Relations within the depth ”FormE” represent sets of calculation results taken over displacement of a structural parameter, generally cell volume or ”Volume”. Data within this set of relations can be directly manipulated by orders of operation to reproduce data visualized in plots contained within the article or its Supporting Information document. These relations are divided based on the type of data stored within them, whether this data is used to produce plots featuring incremental change of a U parameter (FormEURange), an exact exchange fraction (FormEHybrid), or used to categorize which calculated values are in which formation energy plots within the article (FormEFigure). Relations of this form are linked to the head relation, namely Section 2.7, using the key attribute ”CID”. The attribute ”CID”, which features TEXT formatted instances, consists of a unique integer identifier, an underscore, and a secondary unique identifier consisting of either parts of the DOI of this article (10.1021/acs.jpcc.5b05338) or the first characters of the first five words of the title of this article. Therefore, a sample calculation for this article would have a ”CID” of acs.jpcc.5b05338-1.

2.6.1 FormEURange

The relation FormEURange contains the attributes ”CID”, ”UValue”, ”VOpt”, and ”EOpt”. With the exception of ”CID”, all instances within these attributes are FLOAT formatted. Attributes ”CID” and ”EOpt” have already been respectively defined as attributes uniquely identifying sets of calculations within particular articles and uniquely identifying sets of calculations modeling a particular system over a structural displacement. The attribute ”VOpt” is the complement to the Birch-Murnaghan fitted energy ”EOpt” of a system, defining the

fitted volume of the system. The attribute "UValue" defines the constant value of U_{3d} at which a set of calculations plotted in a figure incrementally changing U is evaluated. This quantity is available in the input file data presented in either ParametersInputVASP as the first semicolon separted number in each TEXT formatted instance of attribute "LDAUU", or in ParametersInputQE as the instances of attribute "HubbardU1". However, for the purposes of easily retrieving common data uniformly over multiple software platforms, the "UValue" attribute has been reproduced in this relation.

2.6.2 FormEHybrid

The relation FormEHybrid contains the attributes "CID", "FractionHF", "VOpt", and "EOpt". With the exception of "CID", all instances within these attributes are FLOAT formatted. All attributes besides "FractionHF" have been formerly defined in either Section 2.6.1 or another section. Attribute "FractionHF" represents the fraction of exact exchange employed in a hybrid functional calculation using either the HSE06 or PBE0 functional. In VASP calculations, this quantity is available in the input file data presented in ParametersInputVASP as each TEXT formatted instance of attribute "AEXX". However, for the purposes of easily retrieving common data, the "FractionHF" attribute has been reproduced in this relation.

2.6.3 FormEFigure

The relation FormEFigure contains the attributes "CID" and "Figure". The attribute "Figure" links each cID to the figure (figure name minus file extension) in which it is represented in the article, containing TEXT formatted instances that give the name of each figure.

2.7 Metadata

The relation Metadata serves as the head node in the relational schema containing all of the data within this article and its supporting information section. Despite slight redundancy, it

contains attributes spanning all of the unique identifiers (cID, mID, sID) linking structures to calculated results, in addition to several attributes that conveniently delineate between calculations that are featured in different Figures and underscore different arguments within the article. These additional attributes of convenience contain all TEXT formatted attributes. With regard to these attributes, "PseudoB" details the type of metal cation (B, usually Ti) pseudopotential employed in a particular calculation, "PseudoO" details the type of O pseudopotential employed in the same calculation, "Functional" specifies the type of functional used in that calculation, "Software" indicates whether QE or VASP was used to complete the calculation, and "Method" indicates the type of calculation mapped to a particular cID value.

Values for "PseudoB" relating to calculations detailed in the article or its Supporting Information document sections include "PAW-B" (standard VASP PAW pseudopotential, where B = Ti, V, Ru, or Ir), "PAW-Ti-pv" (*p*-valence inclusive VASP PAW pseudopotential), "PAW-Ti-sv" (*s*-valence inclusive VASP PAW pseudopotential), and "US-Ti-pv-sv" (*p*-valence and *s*-valence inclusive Quantum Espresso Ultrasoft pseudopotential). Values for "PseudoO" relating to calculations detailed in the article or its Supporting Information document sections include "PAW-O" (standard VASP PAW pseudopotential) and "PAW-O-s" (soft VASP PAW pseudopotential). Values for "Functional" relating to calculations detailed in the article or its Supporting Information document sections include "PBE" (Perdew-Burke-Ernzerhof parameterization of the Generalized Gradient Approximation, or GGA, functional),[S11] "LDA" (Local Density Approximation), "PS" (PBEsol or Perdew-Burke-Ernzerhof parameterization of the GGA functional for solids),[S12] "PW91" (Perdew-Wang 1991 functional),[S13] and "AM05" (Armiento-Mattsson 2005 functional).[S14] Also, see the "GGA" tag entry in the VASP documentation to interpret these tuple values. Values for "Software" relating to calculations detailed in the article or its Supporting Information document sections include "VASP"[S15,S16] and "QE" (Quantum Espresso).[S17] Lastly, calculation types or "Method" values employed in this study include "E", or energetic incrementation

calculations, and ”LR”, or linear response calculations.

3 Reproducing the Database

The .csv files accompanying this article and its supporting information file can be imported into any RDBMS platform to reproduce results or perform other operations on aggregated data. However, in order to facilitate the easy integration of .csv files into org-mode and enable full reproducibility of all plots from initial data within this contained document, this article will assemble all .csv files into a single .sqlite file, which can be read into Python scripts within the ”Plot Generation” subsections of this document to generate all plots contained within the article. Plots and tables not associated with plots in the article can also be reproduced using this file. Upon being equipped with proper extensions, Mozilla Firefox[S18] can also import the .sqlite file, perform queries on its contained data, and provide a graphic-user interface suitable for browsing and examining the data.

The created database can be downloaded here. ITEOO_data.sqlite:

In order to generate a single .sqlite file containing all .csv file data needed to reproduce all plots listed in the following section, execute the Python scripts listed below in the order that they are presented. Note that the .csv files and this document should be located in the current working directory prior to executing these scripts, which upload tables that correspond to aforementioned RDBMS subrelations one at a time:

3.1 Structure table

Data corresponding to this relation is stored in Structure.csv:

```
import sqlite3
import os

# We start from scratch. Delete the database if it already exists.
if os.path.exists('ITEOO_data.sqlite'):
```

```
    os.remove('ITEOO_data.sqlite')

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table Structure(Morph TEXT, Struct TEXT, SID INTEGER PRIMARY KEY)''')

with open('Structure.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into Structure(Morph,Struct,SID)
                  VALUES(?,?,?)''', fields)

db.commit()
db.close()
```

3.2 Symmetry Table

Data corresponding to this relation is stored in Symmetry.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table Symmetry(Sym TEXT, Struct TEXT)''')

with open('Symmetry.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into Symmetry(Sym, Struct)
                  VALUES(?,?)''', fields)

db.commit()
db.close()
```

3.3 Composition1 Table

Data corresponding to this relation is stored in Composition1.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table Composition1(At1 TEXT, Stoich1 INTEGER, SID INTEGER, MID INTEGER)''')

with open('Composition1.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into Composition1(At1,Stoich1,SID,MID)
                  VALUES(?,?,?,?)''', fields)

db.commit()
db.close()
```

3.4 Composition2 Table

Data corresponding to this relation is stored in Composition2.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table Composition2(At2 TEXT, Stoich2 INTEGER, SID INTEGER, MID INTEGER)''')

with open('Composition2.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into Composition2(At2,Stoich2,MID,SID)
                  VALUES(?,?,?,?)''', fields)

```

```
db.commit()
db.close()
```

3.5 Metadata Table

Data corresponding to this relation is stored in Metadata.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table Metadata(PseudoB TEXT, PseudoO TEXT,
Functional TEXT, Software TEXT, Method TEXT, CID TEXT,
MID INTEGER, SID INTEGER)''')

with open('Metadata.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into Metadata(PseudoB, PseudoO,
Functional, Software, Method, CID, MID, SID)
                  VALUES(?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.6 FormEURange Table

Data corresponding to this relation is stored in FormEURange.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table FormEURange(UValue FLOAT, VOpt FLOAT, CID TEXT, EOpt FLOAT)''')

with open('FormEURange.csv') as f:
```

```
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into FormEURange(UValue, VOpt, CID, EOpt)
                  VALUES(?,?,?,?)''', fields)

db.commit()
db.close()
```

3.7 FormEHFRange Table

Data corresponding to this relation is stored in FormEHFRange.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table FormEHFRange(FractionHF FLOAT, VOpt FLOAT, CID TEXT, EOpt FLOAT)''')

with open('FormEHFRange.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into FormEHFRange(FractionHF,VOpt,CID,EOpt)
                  VALUES(?,?,?,?)''', fields)

db.commit()
db.close()
```

3.8 CalcResultLRCalc Table

Data corresponding to this relation is stored in CalcResultLRCalc.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')
```

```python

db.execute('''create table CalcResultLRCalc(CID TEXT, U3dIn FLOAT,
U3dOut FLOAT, Energy FLOAT, ChiON150 FLOAT, ChiON100 FLOAT,
ChiON050 FLOAT, ChiOP000 FLOAT, ChiOP050 FLOAT, ChiOP100 FLOAT,
ChiOP150 FLOAT, ChiN150 FLOAT, ChiN100 FLOAT, ChiN050 FLOAT,
ChiP000 FLOAT, ChiP050 FLOAT, ChiP100 FLOAT, ChiP150 FLOAT,
PturbMax FLOAT, PturbMin FLOAT)''')

with open('CalcResultLRCalc.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into CalcResultLRCalc(CID, U3dIn, U3dOut,
Energy, ChiON150, ChiON100, ChiON050, ChiOP000, ChiOP050, ChiOP100,
ChiOP150, ChiN150, ChiN100, ChiN050, ChiP000, ChiP050, ChiP100,
ChiP150, PturbMax, PturbMin)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.9 CalcResultEnergetics Table

Data corresponding to this relation is stored in CalcResultEnergetics.csv:

```python
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table CalcResultEnergetics(Volume FLOAT, NumAtoms FLOAT, EOpt FLOAT, Energy FLOAT)''')

with open('CalcResultEnergetics.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into CalcResultEnergetics(Volume, NumAtoms, EOpt, Energy)
                VALUES(?,?,?,?)''', fields)

```

```
db.commit()
db.close()
```

3.10 ParametersResponseVASP Table

Data corresponding to this relation is stored in ParametersResponseVASP.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table ParametersResponseVASP(Energy FLOAT,
CalcType TEXT, AtomIndex INTEGER, DChargeRHigh FLOAT, DChargeRLow FLOAT,
DChargeRCenter FLOAT, PChargeRHigh FLOAT, PChargeRLow FLOAT,
PChargeRCenter FLOAT, DMagP FLOAT, PMagP FLOAT)''')

with open('ParametersResponseVASP.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into ParametersResponseVASP(Energy,
CalcType, AtomIndex, DChargeRHigh, DChargeRLow, DChargeRCenter,
PChargeRHigh, PChargeRLow, PChargeRCenter, DMagP, PMagP)
                  VALUES(?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.11 ParametersInputVASP Table

Data corresponding to this relation is stored in ParametersInputVASP.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table ParametersInputVASP(Energy FLOAT, NELMDL TEXT,
```

```
SYMPREC TEXT, ISYM TEXT, IBRION TEXT, SIGMA TEXT, NELMIN TEXT,
KPointSampling TEXT, KPoints TEXT, LDAUL TEXT, LDAUJ TEXT, ENCUT TEXT,
ISIF TEXT, ICHARG TEXT, GGA TEXT, LDAUPRINT TEXT, LDAUU TEXT,
Pseudopotential1 TEXT, Pseudopotential2 TEXT, NELM TEXT, NSW TEXT,
LASPH TEXT, MAXMIX TEXT, EDIFF TEXT, ISMEAR TEXT, ISTART TEXT,
LDAU TEXT, LMAXMIX TEXT, EDIFFG TEXT, ISPIN TEXT, LDAUTYPE TEXT,
LORBIT TEXT, AEXX TEXT, NKRED TEXT, PRECFOCK TEXT, NBANDS TEXT,
LMAXFOCK TEXT, ALGO TEXT, LHFCALC TEXT, TIME TEXT, HFSCREEN TEXT)''')

with open('ParametersInputVASP.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into ParametersInputVASP(Energy, NELMDL,
SYMPREC, ISYM, IBRION, SIGMA, NELMIN, KPointSampling, KPoints, LDAUL, LDAUJ,
ENCUT, ISIF, ICHARG, GGA, LDAUPRINT, LDAUU, Pseudopotential1, Pseudopotential2,
NELM, NSW, LASPH, MAXMIX, EDIFF, ISMEAR, ISTART, LDAU, LMAXMIX, EDIFFG, ISPIN,
LDAUTYPE, LORBIT, AEXX, NKRED, PRECFOCK, NBANDS, LMAXFOCK, ALGO, LHFCALC, TIME, HFSCREEN)
                  VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.12 ParametersInputQE Table

Data corresponding to this relation is stored in ParametersInputQE.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table ParametersInputQE(Energy FLOAT,
calculation TEXT, verbosity TEXT, restartmode TEXT, pseudodir TEXT,
outdir TEXT, prefix TEXT, nstep TEXT, wfcollect TEXT,
forcconvthr TEXT, etotconvthr TEXT, ibrav TEXT, nat TEXT,
ntyp TEXT, ecutwfc TEXT, ecutrho TEXT, nspin TEXT,
startingmagnetization1 TEXT, occupations TEXT, smearing TEXT,
degauss TEXT, ldaplusu TEXT, ldaplusukind TEXT, celldm1 TEXT,
```

```
celldm2 TEXT, celldm3 TEXT, HubbardU1 TEXT, HubbardU2 TEXT,
electronmaxstep TEXT, convthr TEXT, diagonalization TEXT,
diagothrinit TEXT, diagofullacc TEXT, startingwfc TEXT,
mixingmode TEXT, mixingbeta TEXT, iondynamics TEXT,
upscale TEXT, Kpoints TEXT, Pseudopotential1 TEXT,
Pseudopotential2 TEXT)''')

with open('ParametersInputQE.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into ParametersInputQE(Energy, calculation,
verbosity, restartmode, pseudodir, outdir, prefix, nstep, wfcollect,
forcconvthr, etotconvthr, ibrav, nat, ntyp, ecutwfc, ecutrho, nspin,
startingmagnetization1, occupations, smearing, degauss, ldaplusu,
ldaplusukind, celldm1, celldm2, celldm3, HubbardU1,
HubbardU2, electronmaxstep, convthr, diagonalization,
diagothrinit, diagofullacc, startingwfc, mixingmode, mixingbeta,
iondynamics, upscale, KPoints, Pseudopotential1, Pseudopotential2)
                  VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.13 ParametersPosFinalVASP Table

Data corresponding to this relation is stored in ParametersPosFinalVASP.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table ParametersPosFinalVASP(Energy FLOAT, Scale FLOAT,
 PrimVec1 TEXT, PrimVec2 TEXT, PrimVec3 TEXT, Coord1 TEXT, Coord2 TEXT,
 Coord3 TEXT, Coord4 TEXT, Coord5 TEXT, Coord6 TEXT, Coord7 TEXT,
 Coord8 TEXT, Coord9 TEXT, Coord10 TEXT, Coord11 TEXT, Coord12 TEXT,
 Coord13 TEXT, Coord14 TEXT, Coord15 TEXT, Coord16 TEXT, Coord17 TEXT,
 Coord18 TEXT, Coord19 TEXT, Coord20 TEXT, Coord21 TEXT, Coord22 TEXT,
```

```
 Coord23 TEXT, Coord24 TEXT, Coord25 TEXT, Coord26 TEXT, Coord27 TEXT,
 Coord28 TEXT, Coord29 TEXT, Coord30 TEXT, Coord31 TEXT, Coord32 TEXT,
 Coord33 TEXT, Coord34 TEXT, Coord35 TEXT, Coord36 TEXT, Coord37 TEXT,
 Coord38 TEXT, Coord39 TEXT, Coord40 TEXT, Coord41 TEXT, Coord42 TEXT,
 Coord43 TEXT, Coord44 TEXT, Coord45 TEXT, Coord46 TEXT, Coord47 TEXT,
 Coord48 TEXT, Coord49 TEXT, Coord50 TEXT, Coord51 TEXT, Coord52 TEXT,
 Coord53 TEXT, Coord54 TEXT, Coord55 TEXT, Coord56 TEXT, Coord57 TEXT,
 Coord58 TEXT, Coord59 TEXT, Coord60 TEXT, Coord61 TEXT, Coord62 TEXT,
 Coord63 TEXT, Coord64 TEXT, Coord65 TEXT, Coord66 TEXT, Coord67 TEXT,
 Coord68 TEXT, Coord69 TEXT, Coord70 TEXT, Coord71 TEXT, Coord72 TEXT,
 Coord73 TEXT, Coord74 TEXT, Coord75 TEXT, Coord76 TEXT, Coord77 TEXT,
 Coord78 TEXT, Coord79 TEXT, Coord80 TEXT, Coord81 TEXT, Coord82 TEXT,
 Coord83 TEXT, Coord84 TEXT, Coord85 TEXT, Coord86 TEXT, Coord87 TEXT,
 Coord88 TEXT, Coord89 TEXT, Coord90 TEXT, Coord91 TEXT, Coord92 TEXT,
 Coord93 TEXT, Coord94 TEXT, Coord95 TEXT, Coord96 TEXT)''')

with open('ParametersPosFinalVASP.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into ParametersPosFinalVASP(Energy, Scale,
 PrimVec1, PrimVec2, PrimVec3, Coord1, Coord2,
 Coord3, Coord4, Coord5, Coord6, Coord7,
 Coord8, Coord9, Coord10, Coord11, Coord12,
 Coord13, Coord14, Coord15, Coord16, Coord17,
 Coord18, Coord19, Coord20, Coord21, Coord22,
 Coord23, Coord24, Coord25, Coord26, Coord27,
 Coord28, Coord29, Coord30, Coord31, Coord32,
 Coord33, Coord34, Coord35, Coord36, Coord37,
 Coord38, Coord39, Coord40, Coord41, Coord42,
 Coord43, Coord44, Coord45, Coord46, Coord47,
 Coord48, Coord49, Coord50, Coord51, Coord52,
 Coord53, Coord54, Coord55, Coord56, Coord57,
 Coord58, Coord59, Coord60, Coord61, Coord62,
 Coord63, Coord64, Coord65, Coord66, Coord67,
 Coord68, Coord69, Coord70, Coord71, Coord72,
 Coord73, Coord74, Coord75, Coord76, Coord77,
 Coord78, Coord79, Coord80, Coord81, Coord82,
 Coord83, Coord84, Coord85, Coord86, Coord87,
 Coord88, Coord89, Coord90, Coord91, Coord92,
```

```
 Coord93, Coord94, Coord95, Coord96)
                 VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.14 ParametersPosFinalQE Table

Data corresponding to this relation is stored in ParametersPosFinalQE.csv:

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table ParametersPosFinalQE(Energy FLOAT, Attyps TEXT,
 Coord1 TEXT, Coord2 TEXT, Coord3 TEXT, Coord4 TEXT, Coord5 TEXT,
 Coord6 TEXT, Coord7 TEXT, Coord8 TEXT, Coord9 TEXT, Coord10 TEXT,
 Coord11 TEXT, Coord12 TEXT)''')

with open('ParametersPosFinalQE.csv') as f:
    lines = f.readlines() #[0].split('\r')

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into ParametersPosFinalQE(Energy, Attyps,
 Coord1, Coord2, Coord3, Coord4, Coord5, Coord6, Coord7, Coord8,
 Coord9, Coord10, Coord11, Coord12)
                 VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', fields)

db.commit()
db.close()
```

3.15 FormEFigure Table

Data corresponding to this relation is stored in FormEFigure.csv:

```python
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

db.execute('''create table FormEFigure(CID TEXT, Figure TEXT)''')

with open('FormEFigure.csv') as f:
    lines = f.readlines()

for line in lines[1:]:
    fields = [x.strip() for x in line.split(',')]
    db.execute('''insert into FormEFigure(CID, Figure)
                  VALUES(?,?)''', fields)

db.commit()
db.close()
```

3.16 Reproducing the Calculation Input Files

In the Supporting Information document of past research,[S9] example codes pursuant to the reproduction of all input files needed to complete energetic calculations were created, querying data from the ".sqlite" database file. The input needed to reproduce any calculation, which includes converged atomic position data, as well as input parameters derived from ".in" files in QE and INCAR, KPOINTS, and POTCAR tags in VASP, was reproduced in these examples for VASP calculations. Improvements made to the storage conventions for data and implementation of a different relational schema necessitate that these examples be reproduced in this paper as well. Thus, corresponding example codes are transcribed below for first retrieving the INCAR, KPOINTS, and POTCAR associated with an example calculation or cID, then retrieving its CONTCAR associated information. The example below depicts the calculations required to calculate a Birch-Murnaghan EOS fitted volume and energy, namely calculations that employ a VASP calculator, the PBE functional, *p*-valence inclusive Ti and standard O PAW pseudopotentials, and a Hubbard U value of 3.00 eV on the Ti cation of a Rutile TiO_2 system:

3.16.1 Example: INCAR, KPOINTS, POTCAR Retrieval

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

NULLchar = '-'
datapts_dict = {}

for row in db.execute('''
select distinct input.ISTART, input.ICHARG, input.ENCUT, input.ISMEAR,
input.SIGMA, input.ISYM, input.IBRION, input.EDIFF, input.EDIFFG, input.MAXMIX,
input.NELMIN, input.NELM, input.NSW, input.ISPIN, input.ISIF, input.GGA,
input.LDAU, input.LDAUU, input.LDAUJ, input.LDAUL, input.LDAUPRINT, input.LASPH,
input.LMAXMIX, input.LORBIT, input.SYMPREC, input.NELMDL, input.LHFCALC,
input.ALGO, input.TIME, input.PRECFOCK, input.LMAXFOCK, input.AEXX, input.NKRED,
input.NBANDS, input.KPoints, input.KPointSampling, input.Pseudopotential1,
input.Pseudopotential2, input.Energy from Structure as s
inner join Symmetry as sy on sy.Struct=s.Struct
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feu.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where s.Morph='Rutile'
and mt.PseudoB='PAW-Ti-pv'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and feu.UValue='3.00'
;'''):
    datapts_list = []

    ( a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,
      u,v,w,x,y,aa,ab,ac,ad,ae,af,ag,ah,ai,aj,ak,al,am,an ) = row

    series =  (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,
               aa,ab,ac,ad,ae,af,ag,ah,ai,aj,ak,al,am,an)

    for z in series:
```

```
        datapts_list.append(z)

    datapts_dict[an] = datapts_list
    del datapts_dict[an][-1]

Input_check = {}
Input_match = {}

att_list = [ 'ISTART', 'ICHARG', 'ENCUT', 'ISMEAR', 'SIGMA', 'ISYM', 'IBRION',
'EDIFF', 'EDIFFG', 'MAXMIX', 'NELMIN', 'NELM', 'NSW', 'ISPIN', 'ISIF', 'GGA',
'LDAU', 'LDAUU', 'LDAUJ', 'LDAUL', 'LDAUPRINT', 'LASPH', 'LMAXMIX', 'LORBIT',
'SYMPREC', 'NELMDL', 'LHFCALC', 'ALGO', 'TIME', 'PRECFOCK', 'LMAXFOCK', 'AEXX',
'NKRED', 'NBANDS', 'KPoints', 'KPointSampling', 'Pseudopotential1', 'Pseudopotential2' ]

for i,j in enumerate( datapts_dict.keys() ):
    if datapts_dict[j] not in Input_check.values():
        list_match = []
        list_check = []

        for k,l in enumerate( att_list ):
            list_check.append( datapts_dict[j][k] )

            if datapts_dict[j][k] != NULLchar:
                list_match.append( (att_list[k], datapts_dict[j][k]) )

        Input_check[i] = list_check
        Input_match[i] = list_match

SortCAR = { "ISTART": 0, "ICHARG": 1, "ENCUT": 2, "ISMEAR": 3,
            "SIGMA": 4, "ISYM": 5, "IBRION": 6, "EDIFF": 7,
            "EDIFFG": 8, "MAXMIX": 9, "NELMIN": 10, "NELM": 11,
            "NSW": 12, "ISPIN": 13, "ISIF": 14,  "GGA": 10,
            "LDAU": 11, "LDAUU": 12, "LDAUJ": 13, "LDAUL": 14,
            "LDAUPRINT": 15, "LASPH": 16, "LMAXMIX": 17,
            "LORBIT": 17, "SYMPREC": 18, "NELMDL": 19,
            "KPointSampling": 20, "KPoints": 21,
            "Pseudopotential1": 22, "Pseudopotential2": 23 }

Input_INCKPTPOT = sorted(Input_match[0], key=lambda tag: SortCAR[tag[0]])

num_KPT = 2
```

```
num_POT = 2
num_INCAR  = len(Input_INCKPTPOT) - num_KPT - num_POT

for i in range( num_INCAR ):
    print Input_INCKPTPOT[i][0],'=',Input_INCKPTPOT[i][1]
print NULLchar

print "0"
print Input_INCKPTPOT[num_INCAR+1][0]
print str(Input_INCKPTPOT[num_INCAR+1][1]).replace(';',' ')
print "0 0 0"
print NULLchar

for i in range( num_POT ):
    print Input_INCKPTPOT[num_INCAR+num_KPT+i][0],Input_INCKPTPOT[num_INCAR+num_KPT+i][1]
```

```
ISTART = 0

ICHARG = 2

ENCUT = 600

ISMEAR = 0

SIGMA = 0.05

ISYM = 1

IBRION = 1

EDIFF = 5.00E-06

EDIFFG = -0.01

MAXMIX = -100

NELMIN = 5

NELM = 200

LDAU = .TRUE.

NSW = 100

LDAUU = 3.00;0.00

ISPIN = 2
```

```
LDAUJ = 0.00;0.00
ISIF = 4
LDAUL = 2;-1
LDAUPRINT = 1
LASPH = .TRUE.
LMAXMIX = 4
SYMPREC = 1.00E-06
NELMDL = -10
-
0
KPoints
8 8 8
0 0 0
-
Pseudopotential1 PAW_PBE Ti_pv 07Sep2000
Pseudopotential2 PAW_PBE O 08Apr2002
```

The script below generates the ordered CONTCAR files used to generate the fitted energy of the system described above, which is derived from the Birch-Murnaghan equation of state fitting procedure and is used to generate plotted data associated with cID values. When available in the database, these CONTCAR input coordinates form structurally relaxed systems that can be outputted to POSCAR files.

3.16.2 Example: CONTCAR Retrieval

```
import sqlite3
import re

Title = 'Ti O'
Attyps = '2 4'
```

```
db = sqlite3.connect('ITEOO_data.sqlite')

NULLchar = '-'
datapts_dict = {}

for row in db.execute('''
select pos.Scale, pos.PrimVec1, pos.PrimVec2, pos.PrimVec3,
pos.Coord1, pos.Coord2, pos.Coord3, pos.Coord4, pos.Coord5,
pos.Coord6, pos.Coord7, pos.Coord8, pos.Coord9, pos.Coord10,
pos.Coord11, pos.Coord12, pos.Coord13, pos.Coord14, pos.Coord15,
pos.Coord16, pos.Coord17, pos.Coord18, pos.Coord19, pos.Coord20,
pos.Coord21, pos.Coord22, pos.Coord23, pos.Coord24, pos.Coord25,
pos.Coord26, pos.Coord27, pos.Coord28, pos.Coord29, pos.Coord30,
pos.Coord31, pos.Coord32, pos.Coord33, pos.Coord34, pos.Coord35,
pos.Coord36, pos.Coord37, pos.Coord38, pos.Coord39, pos.Coord40,
pos.Coord41, pos.Coord42, pos.Coord43, pos.Coord44, pos.Coord45,
pos.Coord46, pos.Coord47, pos.Coord48, pos.Coord49, pos.Coord50,
pos.Coord51, pos.Coord52, pos.Coord53, pos.Coord54, pos.Coord55,
pos.Coord56, pos.Coord57, pos.Coord58, pos.Coord59, pos.Coord60,
pos.Coord61, pos.Coord62, pos.Coord63, pos.Coord64, pos.Coord65,
pos.Coord66, pos.Coord67, pos.Coord68, pos.Coord69, pos.Coord70,
pos.Coord71, pos.Coord72, pos.Coord73, pos.Coord74, pos.Coord75,
pos.Coord76, pos.Coord77, pos.Coord78, pos.Coord79, pos.Coord80,
pos.Coord81, pos.Coord82, pos.Coord83, pos.Coord84, pos.Coord85,
pos.Coord86, pos.Coord87, pos.Coord88, pos.Coord89, pos.Coord90,
pos.Coord91, pos.Coord92, pos.Coord93, pos.Coord94, pos.Coord95,
pos.Coord96, pos.Energy from Structure as s
inner join Symmetry as sy on sy.Struct=s.Struct
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feu.EOpt
inner join ParametersPosFinalVASP as pos on pos.Energy=cre.Energy
where s.Morph='Rutile'
and mt.PseudoB='PAW-Ti-pv'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and feu.UValue='3.00'
```

```
;'''):
    datapts_list = []
    (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,
     aa,ab,ac,ad,ae,af,ag,ah,ai,aj,ak,al,am,an,ao,ap,aq,
     ar,at,au,av,aw,ax,ay,
     ba,bb,bc,bd,be,bf,bg,bh,bi,bj,bk,bl,bm,bn,bo,bp,bq,
     br,bs,bt,bu,bv,bw,bx,by,
     ca,cb,cc,cd,ce,cf,cg,ch,ci,cj,ck,cl,cm,cn,co,cp,cq,
     cr,ct,cu,cv,cw,cx,cy,
     da,db,dc) = row

    series = ( a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,
               aa,ab,ac,ad,ae,af,ag,ah,ai,aj,ak,al,am,an,ao,ap,aq,
               ar,at,au,av,aw,ax,ay,
               ba,bb,bc,bd,be,bf,bg,bh,bi,bj,bk,bl,bm,bn,bo,bp,bq,
               br,bs,bt,bu,bv,bw,bx,by,
               ca,cb,cc,cd,ce,cf,cg,ch,ci,cj,ck,cl,cm,cn,co,cp,cq,
               cr,ct,cu,cv,cw,cx,cy,
               da,db,dc)

    for z in series:
        datapts_list.append(z)

    datapts_dict[a] = datapts_list
    del datapts_dict[a][-1]

Input_check = {}
Input_match = {}

att_list = [ 'Scale', 'PrimVec1', 'PrimVec2', 'PrimVec3',
             'Coord1', 'Coord2', 'Coord3', 'Coord4', 'Coord5',
             'Coord6', 'Coord7', 'Coord8', 'Coord9', 'Coord10',
             'Coord11', 'Coord12', 'Coord13', 'Coord14', 'Coord15',
             'Coord16', 'Coord17', 'Coord18', 'Coord19', 'Coord20',
             'Coord21', 'Coord22', 'Coord23', 'Coord24', 'Coord25',
             'Coord26', 'Coord27', 'Coord28', 'Coord29', 'Coord30',
             'Coord31', 'Coord32', 'Coord33', 'Coord34', 'Coord35',
             'Coord36', 'Coord37', 'Coord38', 'Coord39', 'Coord40',
             'Coord41', 'Coord42', 'Coord43', 'Coord44', 'Coord45',
             'Coord46', 'Coord47', 'Coord48', 'Coord49', 'Coord50',
             'Coord51', 'Coord52', 'Coord53', 'Coord54', 'Coord55',
```

```
88              'Coord56', 'Coord57', 'Coord58', 'Coord59', 'Coord60',
89              'Coord61', 'Coord62', 'Coord63', 'Coord64', 'Coord65',
90              'Coord66', 'Coord67', 'Coord68', 'Coord69', 'Coord70',
91              'Coord71', 'Coord72', 'Coord73', 'Coord74', 'Coord75',
92              'Coord76', 'Coord77', 'Coord78', 'Coord79', 'Coord80',
93              'Coord81', 'Coord82', 'Coord83', 'Coord84', 'Coord85',
94              'Coord86', 'Coord87', 'Coord88', 'Coord89', 'Coord90',
95              'Coord91', 'Coord92', 'Coord93', 'Coord94', 'Coord95',
96              'Coord96' ]
97
98
99  SortCAR = { "Scale": 0, "PrimVec1": 1, "PrimVec2": 2, "PrimVec3": 3,
100 "Coord1": 4, "Coord2": 5, "Coord3": 6, "Coord4": 7,
101 "Coord5": 8, "Coord6": 9 }
102
103 for i,j in enumerate( datapts_dict.keys() ):
104     list_match = []
105
106     for k,l in enumerate( att_list ):
107         if datapts_dict[j][k] != NULLchar:
108             list_match.append( (att_list[k], str(datapts_dict[j][k]) ) )
109
110     Input_match[datapts_dict[j][0]] = list_match
111     Input_match[datapts_dict[j][0]].sort(key=lambda val: SortCAR[val[0]])
112
113 Input_POSCAR = sorted(Input_match.iteritems(), key=lambda (k,v): k)
114
115 Scales_POSCAR = len( Input_match.keys() )
116 Lines_POSCAR  = len( SortCAR )
117
118 for i in range( Scales_POSCAR ):
119     print Title
120
121     for j in range( Lines_POSCAR ):
122         linePOSCAR = re.sub(';', ' ', Input_POSCAR[i][1][j][1].rstrip())
123
124         if Input_POSCAR[i][1][j][0] == 'Coord1':
125             print Attyps
126             print linePOSCAR
127         else:
128             print linePOSCAR
```

```
    print NULLchar
```

```
Ti O
4.66
4.632842111 -7.392e-06 0.0
-7.392e-06 4.632842111 0.0
0.0 0.0 3.002109699
2 4
0.0 0.0 0.0
0.500000569953 0.500000569953 0.499998384636
0.304392469305 0.304392469305 0.0
0.695608670601 0.695608670601 0.0
0.804396384902 0.195604755005 0.499998384636
0.195604755005 0.804396384902 0.499998384636
-
Ti O
4.67
4.644718061 -5.136e-06 0.0
-5.136e-06 4.644718061 0.0
0.0 0.0 3.006046776
2 4
0.0 0.0 0.0
0.500000761619 0.500000761619 0.499998872938
0.304589330459 0.304589330459 0.0
0.695412192778 0.695412192778 0.0
0.804591571444 0.195407798808 0.499998872938
0.195407798808 0.804591571444 0.499998872938
```

```
-
Ti O
4.68
4.656770629 2.126e-06 0.0
2.126e-06 4.656770629 0.0
0.0 0.0 3.009758652
2 4
0.0 0.0 0.0
0.500000777899 0.500000777899 0.500000223938
0.304773300023 0.304773300023 0.0
0.695226108365 0.695226108365 0.0
0.804776503618 0.195222904771 0.500000223938
0.195222904771 0.804776503618 0.500000223938
-
Ti O
4.69
4.668738853 -1.736e-06 0.0
-1.736e-06 4.668738853 0.0
0.0 0.0 3.013583089
2 4
0.0 0.0 0.0
0.500000308756 0.500000308756 0.499999487487
0.30498825792 0.30498825792 0.0
0.695010217685 0.695010217685 0.0
0.804992623676 0.195007993836 0.499999487487
0.195007993836 0.804992623676 0.499999487487
-
```

```
Ti O
4.7
4.680837066 5.052e-06 0.0
5.052e-06 4.680837066 0.0
0.0 0.0 3.017243303
2 4
0.0 0.0 0.0
0.499999773759 0.499999773759 0.499999452646
0.305222856055 0.305222856055 0.0
0.69477882783 0.69477882783 0.0
0.805225425037 0.19477412248 0.499999452646
0.19477412248 0.805225425037 0.499999452646
-
Ti O
4.71
4.693134044 3.564e-06 0.0
3.564e-06 4.693134044 0.0
0.0 0.0 3.020651418
2 4
0.0 0.0 0.0
0.50000025484 0.50000025484 0.500001420555
0.305484330474 0.305484330474 0.0
0.694516179207 0.694516179207 0.0
0.805487180062 0.194513329618 0.500001420555
0.194513329618 0.805487180062 0.500001420555
-
Ti O
```

```
4.72
4.705531569 7.982e-06 0.0
7.982e-06 4.705531569 0.0
0.0 0.0 3.023934776
2 4
0.0 0.0 0.0
0.50000004771 0.50000004771 0.500000863775
0.305739646731 0.305739646731 0.0
0.694260448689 0.694260448689 0.0
0.805742856857 0.194257238562 0.500000863775
0.194257238562 0.805742856857 0.500000863775
-
Ti O
4.73
4.718078713 1.0249e-05 0.0
1.0249e-05 4.718078713 0.0
0.0 0.0 3.027030972
2 4
0.0 0.0 0.0
0.499999050251 0.499999050251 0.50000149123
0.305996773615 0.305996773615 0.0
0.694003446389 0.694003446389 0.0
0.805999272799 0.194000947206 0.50000149123
0.194000947206 0.805999272799 0.50000149123
-
```

4 Results and Discussion

Supporting information pursuant to explaining the results achieved in this article can largely be divided into two sections, namely sections that primarily explain information pertinent to Hubbard U calculations and primarily explain information pertinent to hybrid functional calculations, with the latter section also containing information concerning the comparison and reconciliation of results achieved with both methods.

4.1 Hubbard U

In calculations featuring the incrementation of Hubbard U values, structural relaxation is completed in VASP using a multiple step procedure. Prior to accounting for electron-electron interaction error on the $3d$ Ti orbitals, starting polymorph structure[S19] DFT total energies (E_0) and equilibrium cell volumes (V_0) were resolved by first performing several fixed cell volume, variable cell shape and atomic coordinate structural relaxation calculations encompassing values of V_0, then calculating values of E_0 and V_0 for each system using the Birch-Murnaghan equation of state.[S10] The structural information of the completed relaxation calculation with cell volume closest to V_0 is subsequently applied to a variable cell volume, shape, and atomic coordinate relaxation, the resulting structure of which is integrated into calculations that account for electron-electron interaction error.[S20,S21] Subsequently, electron-electron interaction error is accounted for, namely by using the variable cell relaxed structure derived at $U = 0$ eV as a starting structure for singular fixed cell volume, variable cell shape and atomic coordinate structural relaxation calculations at U greater than 0 eV. This same structure at $U = 0$ eV is applied to a corresponding sets of fixed volume, fixed cell shape, variable atomic coordinate structural relaxations in QE at each value of U tested.

4.1.1 Sample Input Files: Hubbard U

In VASP, only input file information corresponding to (final) fixed cell volume, variable cell shape, and atomic coordinate structural relaxation calculations (e.g.: ISIF = 4) are stored in the database, as this is the information that is necessary to reproduce results demonstrated in this article. However, the steps implemented to perform structural relaxations in this study are detailed below, with sequentially ordered input files listed to illustrate the process through which structural relaxations were performed and pertinent energetics were resolved. For an example calculation of Rutile completed using standard PAW pseudopotentials and the PBE functional, calculations employing ISIF = 4 were completed over ranges of volumes encompassing an equilibrium volume determined from experiment[S19] at $U = 0$ eV, using the input files INCAR, KPOINTS, and POSCAR. Rutile features a primitive tetragonal unit cell and thus two distinct degrees of structural freedom, namely its a axis and c axis (or c/a ratio). Therefore, there are two distinct sets of volumes over which structural relaxations are performed, which correspond directly to variations over these degrees of structural freedom.

Energetic convergence is evaluated iteratively between distinct sets of freedom in each structure, while convergence is determined using a volumetric criterion. Thus, for Rutile, ISIF = 4 relaxations are first performed over the a axis for the first time (iteration #1, it1), then the c axis (iteration #2, it2) for the first time, and then iteratively over the a (it3, it5, etc.) and c (it2, it4, etc.) axes until a volumetric criterion is satisfied. The volumetric criterion for calculations is generally set by the (multiplicative or additive) increment over which equilibrium volumes are tested. For successive iterations of the same structural variable (e.g.: it1 and it3 or it2 and it4 for Rutile), different Birch-Murnaghan resolved volumetric ground states are resolved. Holding the increment over which volumes are tested (over intervals containing the ground state) constant, if the successive Birch-Murnaghan volumetric ground states of two similar successive iterations are closest to the same volumetrically incremented structure (in comparison to all other structures tested over an interval), then energetic convergence is considered achieved. Using a multiplicative scale to elucidate

a simplified example, iteration #1 of a Rutile structural relaxation can be performed on the a axis using ISIF = 4 calculations over the interval of 95%-105% of the experimentally predicted equilibrium cell volume, using increments of 1% (thus 11 calculations would be performed). If a Birch-Murnaghan resolved equilibrium volume 98.8% that of the experimental volume is resolved, iteration #2 is performed on the c axis using the structure resolved at 99% of the experimental volume, as this incremented structure is closest to the 98.8% Birch-Murnaghan resolved ground state volume. Similarly, the structure closest to the Birch-Murnaghan resolved ground state of iteration #2 is applied to calculation #3. Applying comparable volumetric interval and incrementation (i.e.: 1% intervals centered around the original experimental volume) to the third iteration, if a Birch-Murnaghan resolved ground state volume of 98.5% $<V_0<$99.5% is calculated during the third iteration, then the structural relaxation is considered complete and the DFT energy of the structure relaxed is considered converged. An expression representing this approach is written below in the form of Pythonic pseudocode[S2] and conditional statements:

Listing 1: Pythonic pseudocode representation of first step of structural optimization

```
import math
from decimal import *

V_it = []
dof_num = 2
for i in (1 1 N) :
    Energies = []
    Volumes  = []

    for v in (95 S 105) :
        Energies.append( get.Energy(v) )
        Volumes.append( get.Volume(v) )

    E_opt, V_opt = BMurn( Energies, Volumes )

    V_it.append( V_opt )

    if i =< dof_num:
        pass
    else:
        S_float = math.log10( float(S) ) / 100.
        if Decimal( V_it[i] ).quantize( Decimal(S_float) ) == Decimal( V_it[i - dof_num] ).quantize(
        Decimal(S_float) ) :
            E_final = E_opt
            V_final = V_opt
            break

print E_final, V_final
```

The iterative cycle above, which illustrates the aggregation of data that has already been calculated within a fixed range of volumes, requires that several variables be defined or explained. Firstly, "N" represents the total number of iterations accomplished by the structural relaxation, while "i" represents the current iteration being performed by the loop. However, if the pseudocode above was adapted to execute calculations as well, the "for" loop above would be substituted with a "while" loop, "i" would become a counter for the loop itself, and "N" would not be defined. The volume of a current system, or at least the measure of a structural parameter monotonically related to volume over which ISIF = 4 calculations are performed (e.g.: a lattice constant), is represented by "v", whereas "S" represents the increment over which this structural parameter is varied. "Energies" and "Volumes" lists store the DFT total energies and volumes (respectively) of the current structural iteration, while the inferred subroutine or method "BMurn" operates on these lists using the Birch-Murnaghan equation of state to produced the fitted energy and volume values, "E_opt" and "V_opt" (respectively). The variable "dof_num" represents the number of degrees of structural freedom that are modeled in the structural relaxation approach accomplished above, or the number of iterations to be performed before the same structural parameter is repeated in the iterative relaxation procedure. In this procedure, only the a axis and c axis are considered as degrees of freedom,[S19] thus this iterative structural relaxation occurs while alternatively varying those two structural parameters. For each iteration, a value of fitted volume "V_it" is retained. When the value of any two subsequent values of "V_it" varied over the same structural parameter does not change within a tolerance defined by "S", the finalized fitted energetic and volumetric values "E_final" and "V_final" are achieved, respectively. This iterative cycle can also be generalized for orthorhombic systems with three degrees of structural freedom by setting "dof_num" equal to 3.

Using an additive scale as opposed to a multiplicative scale (shown above), as well as a c/a ratio in accompaniment with the "Scale" of a VASP POSCAR file, the following INCAR, KPOINTS, and POSCAR (different files for each iteration) files can be used to perform this

first step of structural optimization. Note that variation in the a axis is performed first, then variation in c/a:

Listing 2: INCAR Rutile PBE PAW_standard VASP it1,it2,it3

```
ISTART = 0 ; ICHARG = 2
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 1

IBRION = 1
EDIFF = 5E-06; EDIFFG = -0.01

MAXMIX = -100
NELMIN = 5
NELMDL = -10
NELM = 200
NSW = 100

ISPIN = 2
ISIF = 4
```

Listing 3: KPOINTS Rutile PBE PAW_standard VASP it1,it2,it3

```
8x8x8
0
Monkhorst
8 8 8
0 0 0
```

The first structural iteration (it1) takes its c/a ratio and atomic coordinates from experimental results, varying the scale (second line) of the POSCAR file to achieve a ground state fitted volume (and lattice constant) closest to that displayed in the POSCAR below (i.e.: lattice constant of 4.66 Angstroms).

Listing 4: POSCAR Rutile PBE PAW_standard VASP it1

```
4.66
4.66
1   0   0
0   1   0
0   0   0.643993896
2 4
direct
  0.000000000   0.000000000   0.000000000
  0.500000000   0.500000000   0.500000000
  0.304880731   0.304880731   0.000000000
 -0.304880731  -0.304880731   0.000000000
  0.804875587   0.195124413   0.500000000
 -0.804875587  -0.195124413   0.500000000
```

The second iteration (it2) varies the c/a ratio, taking atomic coordinates from the CONTCAR of the first relaxation and the a axis length or Scale from the fitting procedure of the first relaxation.

Listing 5: POSCAR Rutile PBE PAW_standard VASP it2

```
4.65922056943
4.65922056943
1 0 0
0 1 0
0 0 0.64
2 4
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.5000000000000000
  0.3048134906850396  0.3048134906850396  0.0000000000000000
  0.6951865093149605  0.6951865093149605  0.0000000000000000
  0.8048075306488891  0.1951924693511108  0.5000000000000000
  0.1951924693511108  0.8048075306488891  0.5000000000000000
```

The final iteration takes the finalized atomic coordinates and fitted c/a ratio from the previous calculation and applies them to a ISIF = 4 structural relaxation varied once again over a axis or Scale. This process of performing iterative calculations in series is repeated until the first iteration of a series has a resolved lattice constant equal to that of the third iteration within the precision imposed by the incrementation of the equation of state (in this case, 0.01 Angstroms).

Listing 6: POSCAR Rutile PBE PAW_standard VASP it3

```
4.66
4.66
1   0   0
0   1   0
0   0   0.638740876486
2 4
direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.5000000000000000
  0.3048082437656310  0.3048082437656310  0.0000000000000000
  0.6951917562343689  0.6951917562343689  0.0000000000000000
  0.8048024643031844  0.1951975356968156  0.5000000000000000
  0.1951975356968156  0.8048024643031844  0.5000000000000000
```

A calculation employing ISIF = 3 that uses the atomic coordinates of the system tested in the previous step closest to the energetic minimum is completed. This calculation uses

the following INCAR file and input file information that is otherwise comparable to the first step:

Listing 7: INCAR Rutile PBE PAW_standard VASP ISIF = 3

```
ISTART = 0 ; ICHARG = 2
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 1

IBRION = 1
EDIFF = 5E-06; EDIFFG = -0.01

MAXMIX = -100
NELMIN = 5
NELMDL = -10
NELM = 200
NSW = 100

ISPIN = 2
ISIF = 3
```

Listing 8: POSCAR Rutile PBE PAW_standard VASP ISIF = 3

```
4.66222984355
  4.66222984355
1 0 0
0 1 0
0 0 0.6375189084585891
2 4
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.5000000000000000
  0.3047756242591685  0.3047756242591685  0.0000000000000000
  0.6952243757408315  0.6952243757408315  0.0000000000000000
  0.8047805609392189  0.1952194390607812  0.5000000000000000
  0.1952194390607812  0.8047805609392189  0.5000000000000000
```

The structures resolved from these ISIF = 3 calculations are applied to Hubbard U calculations that are incremented over the range U = 1.0-6.0 eV (in 1.0 eV increments) in VASP, and are also applied to atomic coordinate relaxations in Quantum Espresso over the range U = 0.0-6.0 eV (in 1.0 eV increments). Though the QE calculations are completed with fixed cell volume and shape, they are still performed over ranges of volumes surrounding the ground state volume to resolve equilibrium energetics (E_0) and volumetric data (V_0) using the Birch-Murnaghan equation of state.[S10]

Listing 9: INCAR Rutile PBE PAW_standard VASP $U = 1.0$ eV

```
ISTART = 0 ; ICHARG = 2
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 1; SYMPREC = 1E-06

IBRION = 1
EDIFF = 5E-06; EDIFFG = -0.01

MAXMIX = -100
NELMIN = 5
NELMDL = -10
NELM = 200
NSW = 100

ISPIN = 2
ISIF = 4

LDAU = .TRUE.
LDAUTYPE = 2
LDAUL = 2 -1
LDAUPRINT = 1

LASPH = .TRUE.
LMAXMIX = 4

LDAUJ = 0.00 0.00
LDAUU = 1.00  0.00
```

Listing 10: Input (Name.in) Rutile PBE PAW_standard QE $U = 1.0$ eV

```
&CONTROL
 calculation = "relax",
 verbosity = "low",
 restart_mode = "from_scratch",
 pseudo_dir = "../../",
 outdir = "./",
 prefix = "BO2_relax",
 nstep = 100,
 wf_collect = .false.
 forc_conv_thr = 1.0E-3
 etot_conv_thr = 1.0E-6
 /
&SYSTEM
 ibrav = 6
 nat = 6
 ntyp = 2
 ecutwfc = 50.0
 ecutrho = 600.0
 nspin = 2
 starting_magnetization(1) = 0.0
 occupations = "smearing",
 smearing = "gaussian",
 degauss = 0.01
 lda_plus_u = .true.
 lda_plus_u_kind = 0
 celldm(1) = 8.85
 celldm(3) = 0.637562623236
 Hubbard_U(1) = 1.00
 /
&ELECTRONS
 electron_maxstep = 100
 conv_thr = 1D-9
 diagonalization = "david",
 diago_thr_init = 1D-2
 diago_full_acc = .false.
 startingwfc = "atomic+random",
 mixing_mode = "plain",
 mixing_beta = 0.3
 /
&IONS
 ion_dynamics = "bfgs",
 upscale = 1000
 /
ATOMIC_SPECIES
Ti   1.0   Ti.pbe-sp-van_ak.UPF
O    1.0   O.pbe-rrkjus.UPF

ATOMIC_POSITIONS (crystal)
Ti       0.000000000   0.000000000   0.000000000
Ti       0.500000000   0.500000000   0.500000000
O        0.305356333   0.305356333   0.000000000
O        0.694643667   0.694643667   0.000000000
O        0.805354772   0.194645228   0.500000000
O        0.194645228   0.805354772   0.500000000

K_POINTS (automatic)
 8 8 8 0 0 0
```

4.1.2 Plot Generation: Hubbard U

In the article, there are four plots featuring Hubbard U incrementation of the Rutile, Anatase, Columbite, and Brookite polymorphs, which illustrate the effects of varying U over sets of polymorphs sharing similar features. In the first plot featuring U variation, energetics for the PBE and LDA functionals are featured with the use of standard pseudopotentials, in accompaniment with energetics featuring the PBE functional, a standard Ti pseudopotential, and a soft O pseudopotential. The relational algebraic expression and complementary MySQL query used to generate these plots are transcribed below:

$$\Pi_{PseudoO,Functional,Morph,UValue,EOpt,Stoich1}$$

$$\sigma[(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O') \vee (PseudoO =' PAW - O - s')]$$

$$\sigma \wedge [(Functional =' PBE') \vee (Functional =' LDA')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query PBE LDA Ti O O_s
import matplotlib.pyplot as plt
import sqlite3

```

```
db = sqlite3.connect('ITEOO_data.sqlite')

Pseudo_O, Functional, Polymorph, Uvalue, Eopt, atomnum = [], [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct mt.PseudoO, mt.Functional, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and (mt.PseudoO='PAW-O' or mt.PseudoO='PAW-O-s')
and (mt.Functional='PBE' or mt.Functional='LDA')
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d,e,f = row
    datapts_list.append( (a,b,c,d,e,f) )

    Pseudo_O += [a]
    Functional += [b]
    Polymorph += [c]

    datapts_dict[row] = datapts_list

Pseudo_list = list( set( Pseudo_O ) )
Functional_list = list( set( Functional ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Pseudo_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}
```

```
    for j in Functional_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = {}

        for k in Morph_list:
            SortElist[i][j][k] = {}
            EBsiteMatch[i][j][k] = []
            del_list = []

            for l in datapts_dict:
                if l[0] == i and l[1] == j and l[2] == k:
                    SortElist[ l[0] ][ l[1] ][ l[2] ][ l[3] ] = float( l[4] ) / float( l[5] )
                    del_list.append( datapts_dict[l] )
                else:
                    pass

            for l in del_list:
                del l

for i in Pseudo_list:
    for j in Functional_list:
        for k in Morph_list:
            U_list = SortElist[i][j][k].keys()
            U_list.sort()
            EmapU_list = []

            for l in U_list:
                EmapU_value = SortElist[i][j][k][l] - SortElist[i][j][WRT_Morph][l]
                EmapU_list.append( EmapU_value )

            EBsiteMatch[i][j][k].append( EmapU_list )
            EBsiteMatch[i][j][k].append( U_list )

ax = plt.gca()

print "PBE Functional, O pseudopotential:"
print "Rutile: " + str(EBsiteMatch['PAW-O']['PBE']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O']['PBE']['Rutile'][0][2:4])
print str(EBsiteMatch['PAW-O']['PBE']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PAW-O']['PBE']['Anatase'][0][0:2])
```

```
print str(EBsiteMatch['PAW-O']['PBE']['Anatase'][0][2:4])
print str(EBsiteMatch['PAW-O']['PBE']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['PAW-O']['PBE']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O']['PBE']['Columbite'][0][2:4])
print str(EBsiteMatch['PAW-O']['PBE']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['PAW-O']['PBE']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O']['PBE']['Brookite'][0][2:4])
print str(EBsiteMatch['PAW-O']['PBE']['Brookite'][0][4:]) + "\n"

print "LDA Functional, O pseudopotential:"
print "Rutile: " + str(EBsiteMatch['PAW-O']['LDA']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O']['LDA']['Rutile'][0][2:4])
print str(EBsiteMatch['PAW-O']['LDA']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PAW-O']['LDA']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O']['LDA']['Anatase'][0][2:4])
print str(EBsiteMatch['PAW-O']['LDA']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['PAW-O']['LDA']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O']['LDA']['Columbite'][0][2:4])
print str(EBsiteMatch['PAW-O']['LDA']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['PAW-O']['LDA']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O']['LDA']['Brookite'][0][2:4])
print str(EBsiteMatch['PAW-O']['LDA']['Brookite'][0][4:]) + "\n"

print "PBE Functional, O_s pseudopotential:"
print "Rutile: " + str(EBsiteMatch['PAW-O-s']['PBE']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['PBE']['Rutile'][0][2:4])
print str(EBsiteMatch['PAW-O-s']['PBE']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PAW-O-s']['PBE']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['PBE']['Anatase'][0][2:4])
print str(EBsiteMatch['PAW-O-s']['PBE']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['PAW-O-s']['PBE']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['PBE']['Columbite'][0][2:4])
print str(EBsiteMatch['PAW-O-s']['PBE']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['PAW-O-s']['PBE']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['PBE']['Brookite'][0][2:4])
print str(EBsiteMatch['PAW-O-s']['PBE']['Brookite'][0][4:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['PAW-O']['PBE']['Rutile'][1], EBsiteMatch['PAW-O']['PBE']['Rutile'][0], 'k-')

plt.plot(EBsiteMatch['PAW-O']['PBE']['Anatase'][1], EBsiteMatch['PAW-O']['PBE']['Anatase'][0],
```

```
'bo-', label = r'$\Delta$E$_{R-A}$,PBE')
plt.plot(EBsiteMatch['PAW-O']['PBE']['Columbite'][1], EBsiteMatch['PAW-O']['PBE']['Columbite'][0],
'ro-', label = r'$\Delta$E$_{R-C}$,PBE')
plt.plot(EBsiteMatch['PAW-O']['PBE']['Brookite'][1], EBsiteMatch['PAW-O']['PBE']['Brookite'][0],
'go-', label = r'$\Delta$E$_{R-B}$,PBE')

plt.plot(EBsiteMatch['PAW-O-s']['PBE']['Anatase'][1], EBsiteMatch['PAW-O-s']['PBE']['Anatase'][0],
'bv-', label = r'$\Delta$E$_{R-A}$,PBEs')
plt.plot(EBsiteMatch['PAW-O-s']['PBE']['Columbite'][1], EBsiteMatch['PAW-O-s']['PBE']['Columbite'][0],
'rv-', label = r'$\Delta$E$_{R-C}$,PBEs')
plt.plot(EBsiteMatch['PAW-O-s']['PBE']['Brookite'][1], EBsiteMatch['PAW-O-s']['PBE']['Brookite'][0],
'gv-', label = r'$\Delta$E$_{R-B}$,PBEs')

plt.plot(EBsiteMatch['PAW-O']['LDA']['Anatase'][1], EBsiteMatch['PAW-O']['LDA']['Anatase'][0],
'bs-', label = r'$\Delta$E$_{R-A}$,LDA')
plt.plot(EBsiteMatch['PAW-O']['LDA']['Columbite'][1], EBsiteMatch['PAW-O']['LDA']['Columbite'][0],
'rs-', label = r'$\Delta$E$_{R-C}$,LDA')
plt.plot(EBsiteMatch['PAW-O']['LDA']['Brookite'][1], EBsiteMatch['PAW-O']['LDA']['Brookite'][0],
'gs-', label = r'$\Delta$E$_{R-B}$,LDA')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.3) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper center', prop={'size':6.5}, ncol = 2)
plt.axvspan(2.79, 4.3, facecolor='m', alpha=0.5)

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-RACB-PBELDAPBEs' + '.' + ext, dpi=300)
plt.clf()
```

PBE Functional, O pseudopotential:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [-0.081062179999999984, -0.052122670000002813]

[-0.02370224500000262, 0.005027775000002066]

[0.033729065000002834, 0.062429919999999584, 0.090796440000001866]

Columbite: [-0.0041790349999999421, 0.012991809999999049]

[0.025531667499997468, 0.034880245000000087]

[0.042688772500000027, 0.049023697500000907, 0.054641072500000831]

Brookite: [-0.040741385000000463, -0.018720677500002836]

[4.6207499998729418e-05, 0.017773080000001329]

[0.035646842500000275, 0.052357034999999996, 0.069235472500000839]

LDA Functional, O pseudopotential:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [-0.012130820000002984, 0.017095379999997107]

[0.046785209999999466, 0.076938040000001706]

[0.10735272500000193, 0.13794045499999896, 0.16851013000000137]

Columbite: [-0.02016459000000026, -0.0054327399999998249]

[0.0065270649999966679, 0.016430554999999458]

[0.024749195000001833, 0.032084362500000907, 0.038648985000001801]

Brookite: [-0.017426940000000002, 0.0031198474999989401]

[0.022930064999997057, 0.042383980000000321]

[0.061421407499999248, 0.08016881249999841, 0.09873492750000068]

PBE Functional, O_s pseudopotential:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

```
Anatase: [-0.079714805000001832, -0.050936149999998293]
[-0.022045460000001071, 0.0069371999999994216]
[0.035951820000001078, 0.065040934999998967, 0.09360556000000031]
Columbite: [-0.005619084999999302, 0.011333950000000925]
[0.024398860000001577, 0.034633069999998156]
[0.042769752499999925, 0.049681027500000141, 0.055710704999999194]
Brookite: [-0.03926586000000043, -0.018175699999996908]
[0.0011124675000004913, 0.019419677500000176]
[0.037090197499999533, 0.054461702499999376, 0.071194444999999718]
```

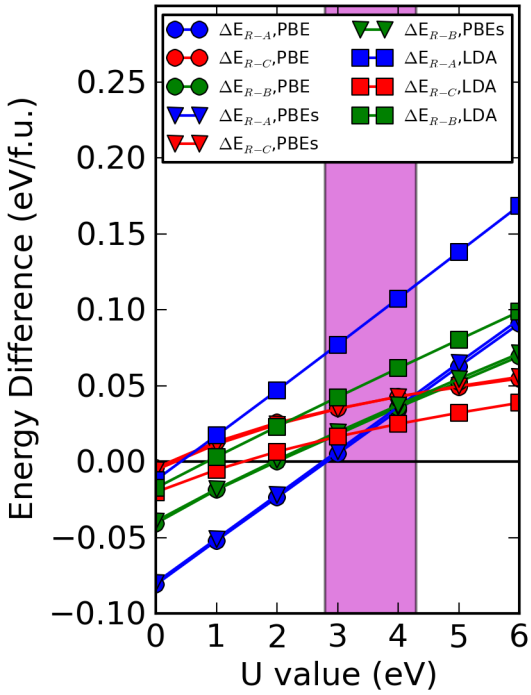

Figure S2

The second plot features solely the PBE functional, albeit pseudopotential sets in this plot feature p-valence inclusive Ti and standard O, as well as p-valence inclusive Ti and soft O. The relational algebraic expression and complementary MySQL query used to generate these plots are transcribed below:

$$\Pi_{PseudoO,Morph,UValue,EOpt,Stoich1}$$

$$\sigma[(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti - pv')]$$

$$\sigma \wedge [(PseudoO =' PAW - O') \vee (PseudoO =' PAW - O - s')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6') \vee (UValue =' 7')$$

$$\sigma \vee (UValue =' 8') \vee (UValue =' 9')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query PBE Ti_pv O O_s
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Pseudo_O, Polymorph, Uvalue, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct mt.PseudoO, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
```

```
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti-pv'
and (mt.PseudoO='PAW-O' or mt.PseudoO='PAW-O-s')
and mt.Functional='PBE'
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6' or feu.UValue='7'
or feu.UValue='8' or feu.UValue='9')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    Pseudo_O += [a]
    Polymorph += [b]

    datapts_dict[row] = datapts_list

Pseudo_list = list( set( Pseudo_O ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Pseudo_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Morph_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
                del_list.append( datapts_dict[k] )
            else:
                pass
```

```
        for l in del_list:
            del l

for i in Pseudo_list:
    for j in Morph_list:
        U_list = SortElist[i][j].keys()
        U_list.sort()
        EmapU_list = []

        for k in U_list:
            EmapU_value = SortElist[i][j][k] - SortElist[i][WRT_Morph][k]
            EmapU_list.append( EmapU_value )

        EBsiteMatch[i][j].append( EmapU_list )
        EBsiteMatch[i][j].append( U_list )

ax = plt.gca()

print "PBE Functional, Ti_pv O pseudopotentials:"
print "Rutile: " + str(EBsiteMatch['PAW-O']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O']['Rutile'][0][2:5])
print str(EBsiteMatch['PAW-O']['Rutile'][0][5:8])
print str(EBsiteMatch['PAW-O']['Rutile'][0][8:])
print "Anatase: " + str(EBsiteMatch['PAW-O']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O']['Anatase'][0][2:5])
print str(EBsiteMatch['PAW-O']['Anatase'][0][5:8])
print str(EBsiteMatch['PAW-O']['Anatase'][0][8:])
print "Columbite: " + str(EBsiteMatch['PAW-O']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O']['Columbite'][0][2:5])
print str(EBsiteMatch['PAW-O']['Columbite'][0][5:8])
print str(EBsiteMatch['PAW-O']['Columbite'][0][8:])
print "Brookite: " + str(EBsiteMatch['PAW-O']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O']['Brookite'][0][2:5])
print str(EBsiteMatch['PAW-O']['Brookite'][0][5:8])
print str(EBsiteMatch['PAW-O']['Brookite'][0][8:]) + "\n"

print "PBE Functional, Ti_pv O_s pseudopotentials:"
print "Rutile: " + str(EBsiteMatch['PAW-O-s']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][5:8])
```

```
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][8:])
print "Anatase: " + str(EBsiteMatch['PAW-O-s']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][8:])
print "Columbite: " + str(EBsiteMatch['PAW-O-s']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][8:])
print "Brookite: " + str(EBsiteMatch['PAW-O-s']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][8:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['PAW-O']['Rutile'][1], EBsiteMatch['PAW-O']['Rutile'][0], 'k-')

plt.plot(EBsiteMatch['PAW-O']['Anatase'][1], EBsiteMatch['PAW-O']['Anatase'][0],
'bo-', label = r'$\Delta$E$_{R-A}$,PBEpv')
plt.plot(EBsiteMatch['PAW-O']['Columbite'][1], EBsiteMatch['PAW-O']['Columbite'][0],
'ro-', label = r'$\Delta$E$_{R-C}$,PBEpv')
plt.plot(EBsiteMatch['PAW-O']['Brookite'][1], EBsiteMatch['PAW-O']['Brookite'][0],
'go-', label = r'$\Delta$E$_{R-B}$,PBEpv')

plt.plot(EBsiteMatch['PAW-O-s']['Anatase'][1], EBsiteMatch['PAW-O-s']['Anatase'][0],
'bs-', label = r'$\Delta$E$_{R-A}$,PBEpvs')
plt.plot(EBsiteMatch['PAW-O-s']['Columbite'][1], EBsiteMatch['PAW-O-s']['Columbite'][0],
'rs-', label = r'$\Delta$E$_{R-C}$,PBEpvs')
plt.plot(EBsiteMatch['PAW-O-s']['Brookite'][1], EBsiteMatch['PAW-O-s']['Brookite'][0],
'gs-', label = r'$\Delta$E$_{R-B}$,PBEpvs')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.15) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper center', prop={'size':6}, ncol = 2)
plt.axvspan(4.74, 7.0, facecolor='m', alpha=0.5)

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
```

```
    plt.savefig('./figures/TiO2-stability-RACB-pvpvs' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, Ti_pv O pseudopotentials:
Rutile: [0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0]
Anatase: [-0.092100235000000197, -0.072958069999998543]
[-0.053949949999999802, -0.035017605000000174, -0.016135710000000358]
[0.0023775249999999915, 0.020546449999997662, 0.038342809999999616]
[0.055574434999996924, 0.072017510000002005]
Columbite: [-0.016600305000000759, 0.00035245000000116988]
[0.013370840000000328, 0.023233480000001805, 0.03092596000000114]
[0.037183912500001526, 0.042455104999998383, 0.046829240000000993]
[0.050712797499997464, 0.054088982500001492]
Brookite: [-0.050446005000001293, -0.031892899999999003]
[-0.016287384999998267, -0.0023999524999993582, 0.01045637750000239]
[0.022726665000000423, 0.034499479999997362, 0.045864807499999216]
[0.056799632499998864, 0.067223224999999331]

PBE Functional, Ti_pv O_s pseudopotentials:
Rutile: [0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0]
Anatase: [-0.090680054999999982, -0.071306109999998313]
[-0.052094455000002426, -0.032983779999998575, -0.014090194999997863]
```

[0.0045151549999999929, 0.022981115000000329, 0.040872674999999248]

[0.058173119999999301, 0.074692024999997386]

Columbite: [-0.01677761999999916, -8.7690000000861801e-05]

[0.012759249999998445, 0.022705037499999747, 0.030520512500000763]

[0.036886420000001863, 0.042409889999998285, 0.047035590000000127]

[0.051085342499998632, 0.054613257499998014]

Brookite: [-0.048847532499998181, -0.030495177500000636]

[-0.014870725000001528, -0.00083195499999888511, 0.012186297500001331]

[0.02451357750000227, 0.036434485000000905, 0.047960935000002536]

[0.058967419999998327, 0.069486784999998719]

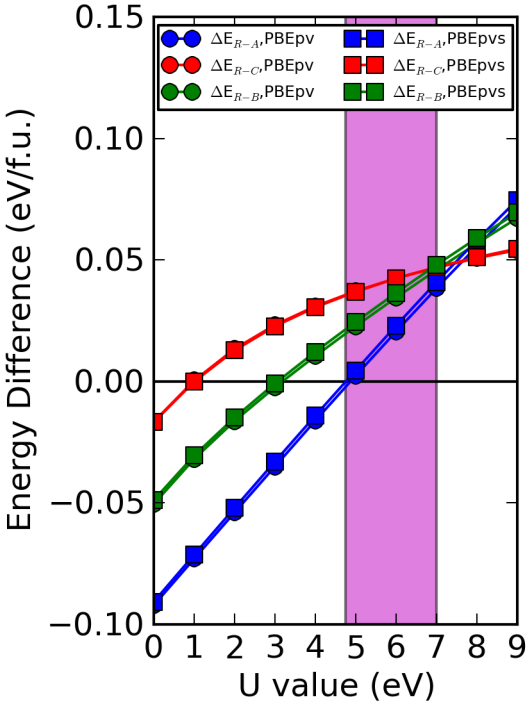

Figure S3

The third plot features solely the PBE functional, albeit pseudopotential sets in this plot feature *s*-valence inclusive Ti and standard O, as well as *s*-valence inclusive Ti and soft O. The relational algebraic expression and complementary MySQL query used to generate these

plots are transcribed below:

$$\Pi_{PseudoO,Morph,UValue,EOpt,Stoich1}$$

$$\sigma[(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti - sv')]$$

$$\sigma \wedge [(PseudoO =' PAW - O') \vee (PseudoO =' PAW - O - s')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6') \vee (UValue =' 7')$$

$$\sigma \vee (UValue =' 8') \vee (UValue =' 9')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query PBE Ti_sv O O_s
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Pseudo_O, Polymorph, Uvalue, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct mt.PseudoO, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
```

```
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti-sv'
and (mt.PseudoO='PAW-O' or mt.PseudoO='PAW-O-s')
and mt.Functional='PBE'
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6' or feu.UValue='7'
or feu.UValue='8' or feu.UValue='9')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    Pseudo_O += [a]
    Polymorph += [b]

    datapts_dict[row] = datapts_list

Pseudo_list = list( set( Pseudo_O ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Pseudo_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Morph_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
                del_list.append( datapts_dict[k] )
```

```
            else:
                pass

        for l in del_list:
            del l

for i in Pseudo_list:
    for j in Morph_list:
        U_list = SortElist[i][j].keys()
        U_list.sort()
        EmapU_list = []

        for k in U_list:
            EmapU_value = SortElist[i][j][k] - SortElist[i][WRT_Morph][k]
            EmapU_list.append( EmapU_value )

        EBsiteMatch[i][j].append( EmapU_list )
        EBsiteMatch[i][j].append( U_list )

ax = plt.gca()

print "PBE Functional, Ti_sv O pseudopotentials:"
print "Rutile: " + str(EBsiteMatch['PAW-O']['Rutile'][0][0:2])
print str(EBsiteMatch['PAW-O']['Rutile'][0][2:5])
print str(EBsiteMatch['PAW-O']['Rutile'][0][5:8])
print str(EBsiteMatch['PAW-O']['Rutile'][0][8:])
print "Anatase: " + str(EBsiteMatch['PAW-O']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O']['Anatase'][0][2:5])
print str(EBsiteMatch['PAW-O']['Anatase'][0][5:8])
print str(EBsiteMatch['PAW-O']['Anatase'][0][8:])
print "Columbite: " + str(EBsiteMatch['PAW-O']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O']['Columbite'][0][2:5])
print str(EBsiteMatch['PAW-O']['Columbite'][0][5:8])
print str(EBsiteMatch['PAW-O']['Columbite'][0][8:])
print "Brookite: " + str(EBsiteMatch['PAW-O']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O']['Brookite'][0][2:5])
print str(EBsiteMatch['PAW-O']['Brookite'][0][5:8])
print str(EBsiteMatch['PAW-O']['Brookite'][0][8:]) + "\n"

print "PBE Functional, Ti_sv O_s pseudopotentials:"
print "Rutile: " + str(EBsiteMatch['PAW-O-s']['Rutile'][0][0:2])
```

```
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Rutile'][0][8:])
print "Anatase: " + str(EBsiteMatch['PAW-O-s']['Anatase'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Anatase'][0][8:])
print "Columbite: " + str(EBsiteMatch['PAW-O-s']['Columbite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Columbite'][0][8:])
print "Brookite: " + str(EBsiteMatch['PAW-O-s']['Brookite'][0][0:2])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][2:5])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][5:8])
print str(EBsiteMatch['PAW-O-s']['Brookite'][0][8:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['PAW-O']['Rutile'][1], EBsiteMatch['PAW-O']['Rutile'][0], 'k-')

plt.plot(EBsiteMatch['PAW-O']['Anatase'][1], EBsiteMatch['PAW-O']['Anatase'][0],
'bo-', label = r'$\Delta$E$_{R-A}$,PBEsv')
plt.plot(EBsiteMatch['PAW-O']['Columbite'][1], EBsiteMatch['PAW-O']['Columbite'][0],
'ro-', label = r'$\Delta$E$_{R-C}$,PBEsv')
plt.plot(EBsiteMatch['PAW-O']['Brookite'][1], EBsiteMatch['PAW-O']['Brookite'][0],
'go-', label = r'$\Delta$E$_{R-B}$,PBEsv')

plt.plot(EBsiteMatch['PAW-O-s']['Anatase'][1], EBsiteMatch['PAW-O-s']['Anatase'][0],
'bs-', label = r'$\Delta$E$_{R-A}$,PBEsvs')
plt.plot(EBsiteMatch['PAW-O-s']['Columbite'][1], EBsiteMatch['PAW-O-s']['Columbite'][0],
'rs-', label = r'$\Delta$E$_{R-C}$,PBEsvs')
plt.plot(EBsiteMatch['PAW-O-s']['Brookite'][1], EBsiteMatch['PAW-O-s']['Brookite'][0],
'gs-', label = r'$\Delta$E$_{R-B}$,PBEsvs')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.1) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper center', prop={'size':6}, ncol = 2)
plt.axvspan(5.8, 8.2, facecolor='m', alpha=0.5)

plt.gcf().subplots_adjust(left=0.27)

```

```
for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-RACB-svsvs' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, Ti_sv O pseudopotentials:
Rutile: [0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0]
Anatase: [-0.094374219999998843, -0.078359495000000834]
[-0.062416365000000695, -0.046427270000002352, -0.0305251300000009]
[-0.01476947000000095, 0.00088234499999728655, 0.016071959999997887]
[0.030955935000001489, 0.045444939999999434]
Columbite: [-0.01433857500000002, 0.00074320999999955006]
[0.01246743999999822, 0.021624347499997754, 0.02885451999999944]
[0.034937379999998797, 0.039858472499997077, 0.04409091750000016]
[0.047848267500000929, 0.051188480000000425]
Brookite: [-0.050264074999997632, -0.034201514999999461]
[-0.020637135000001194, -0.0084279074999997761, 0.0030013449999977126]
[0.01373522749999978, 0.024097179999998275, 0.034103410000000167]
[0.043912065000000666, 0.053353257499999529]

PBE Functional, Ti_sv O_s pseudopotentials:
Rutile: [0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0]
Anatase: [-0.092738669999999246, -0.076523710000000023]
```

```
[-0.060386694999998269, -0.04426828999999799, -0.028419155000001695]
[-0.012491020000002351, 0.003334350000002928, 0.018666955000000485]
[0.033633014999999489, 0.048122259999999528]
Columbite: [-0.0141523449999994, 0.0005109550000028662]
[0.011996330000002331, 0.021098524999999313, 0.028491827499998124]
[0.034470839999997338, 0.039645769999999914, 0.044125305000001447]
[0.048050769999999687, 0.051508152499998516]
Brookite: [-0.048430632499997017, -0.032640882499997304]
[-0.019072982499999114, -0.0067917625000006865, 0.0046064474999987226]
[0.015516134999998599, 0.026016297500000007, 0.036198260000002591]
[0.046092070000000263, 0.055560822499998608]
```

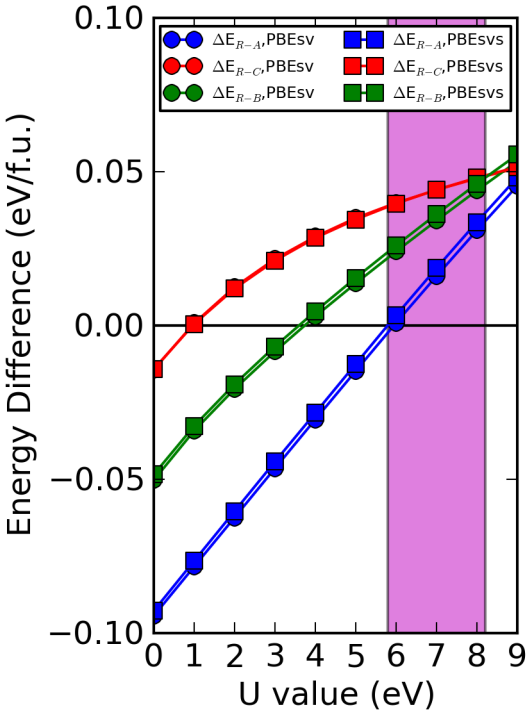

Figure S4

The fourth plot features solely standard pseudpotentials, albeit calculation in this plot use the PBEsol, PW91, and AM05 functionals. The relational algebraic expression and

complementary MySQL query used to generate these plots are transcribed below:

$$\Pi_{Functional,Morph,UValue,EOpt,Stoich1}$$

$$\sigma[(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Functional =' PS') \vee (Functional =' PW91') \vee (Functional =' AM05')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query PBEsol PW91 AM05 Ti O
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Functional, Polymorph, Uvalue, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select mt.Functional, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
```

```
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and (mt.Functional='PS' or mt.Functional='PW91' or mt.Functional='AM05')
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    Functional += [a]
    Polymorph += [b]

    datapts_dict[row] = datapts_list

Functional_list = list( set( Functional ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Functional_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Morph_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
                del_list.append( datapts_dict[k] )
            else:
                pass

```

```
            for l in del_list:
                del l

for i in Functional_list:
    for j in Morph_list:
        U_list = SortElist[i][j].keys()
        U_list.sort()
        EmapU_list = []

        for k in U_list:
            EmapU_value = SortElist[i][j][k] - SortElist[i][WRT_Morph][k]
            EmapU_list.append( EmapU_value )

        EBsiteMatch[i][j].append( EmapU_list )
        EBsiteMatch[i][j].append( U_list )

ax = plt.gca()

print "PBEsol Functional:"
print "Rutile: " + str(EBsiteMatch['PS']['Rutile'][0][0:2])
print str(EBsiteMatch['PS']['Rutile'][0][2:4])
print str(EBsiteMatch['PS']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PS']['Anatase'][0][0:2])
print str(EBsiteMatch['PS']['Anatase'][0][2:4])
print str(EBsiteMatch['PS']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['PS']['Columbite'][0][0:2])
print str(EBsiteMatch['PS']['Columbite'][0][2:4])
print str(EBsiteMatch['PS']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['PS']['Brookite'][0][0:2])
print str(EBsiteMatch['PS']['Brookite'][0][2:4])
print str(EBsiteMatch['PS']['Brookite'][0][4:]) + "\n"

print "PW91 Functional:"
print "Rutile: " + str(EBsiteMatch['PW91']['Rutile'][0][0:2])
print str(EBsiteMatch['PW91']['Rutile'][0][2:4])
print str(EBsiteMatch['PW91']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PW91']['Anatase'][0][0:2])
print str(EBsiteMatch['PW91']['Anatase'][0][2:4])
print str(EBsiteMatch['PW91']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['PW91']['Columbite'][0][0:2])
print str(EBsiteMatch['PW91']['Columbite'][0][2:4])
```

```
print str(EBsiteMatch['PW91']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['PW91']['Brookite'][0][0:2])
print str(EBsiteMatch['PW91']['Brookite'][0][2:4])
print str(EBsiteMatch['PW91']['Brookite'][0][4:]) + "\n"

print "AM05 Functional:"
print "Rutile: " + str(EBsiteMatch['AM05']['Rutile'][0][0:2])
print str(EBsiteMatch['AM05']['Rutile'][0][2:4])
print str(EBsiteMatch['AM05']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['AM05']['Anatase'][0][0:2])
print str(EBsiteMatch['AM05']['Anatase'][0][2:4])
print str(EBsiteMatch['AM05']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['AM05']['Columbite'][0][0:2])
print str(EBsiteMatch['AM05']['Columbite'][0][2:4])
print str(EBsiteMatch['AM05']['Columbite'][0][4:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['PS']['Rutile'][1], EBsiteMatch['PS']['Rutile'][0], 'k-')

plt.plot(EBsiteMatch['PS']['Anatase'][1], EBsiteMatch['PS']['Anatase'][0],
         'bo-', label = r'$\Delta$E$_{R-A}$,PBEsol')
plt.plot(EBsiteMatch['PS']['Columbite'][1], EBsiteMatch['PS']['Columbite'][0],
         'ro-', label = r'$\Delta$E$_{R-C}$,PBEsol')
plt.plot(EBsiteMatch['PS']['Brookite'][1], EBsiteMatch['PS']['Brookite'][0],
         'go-', label = r'$\Delta$E$_{R-B}$,PBEsol')

plt.plot(EBsiteMatch['PW91']['Anatase'][1], EBsiteMatch['PW91']['Anatase'][0],
         'bs-', label = r'$\Delta$E$_{R-A}$,PW91')
plt.plot(EBsiteMatch['PW91']['Columbite'][1], EBsiteMatch['PW91']['Columbite'][0],
         'rs-', label = r'$\Delta$E$_{R-C}$,PW91')
plt.plot(EBsiteMatch['PW91']['Brookite'][1], EBsiteMatch['PW91']['Brookite'][0],
         'gs-', label = r'$\Delta$E$_{R-B}$,PW91')

plt.plot(EBsiteMatch['AM05']['Anatase'][1], EBsiteMatch['AM05']['Anatase'][0],
         'bv-', label = r'$\Delta$E$_{R-A}$,AM05')
plt.plot(EBsiteMatch['AM05']['Columbite'][1], EBsiteMatch['AM05']['Columbite'][0],
         'rv-', label = r'$\Delta$E$_{R-C}$,AM05')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.2) )
plt.ylabel('Energy Difference (eV/f.u.)')
```

```
plt.legend(loc = 'upper center', prop={'size':6}, ncol = 2)
plt.axvspan(2.79, 4.0, facecolor='m', alpha=0.5)

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-RACB-PSPW91AM05' + '.' + ext, dpi=300)
plt.clf()
```

```
PBEsol Functional:
Rutile: [0.0, 0.0]
[0.0, 0.0]
[0.0, 0.0, 0.0]
Anatase: [-0.044709259999997641, -0.016211800000000665]
[0.012816614999998421, 0.042070845000001356]
[0.071510459999998943, 0.10124286000000282, 0.13038651999999828]
Columbite: [-0.011132689999996614, 0.0035706650000015827]
[0.015194449999999193, 0.02447457750000126]
[0.032272934999998171, 0.038946925000001187, 0.044734534999999909]
Brookite: [-0.028607902499999227, -0.0084837224999994021]
[0.010678325000000655, 0.029249395000000789]
[0.047449107500000309, 0.065380857500002776, 0.082839024999998401]

PW91 Functional:
Rutile: [0.0, 0.0]
[0.0, 0.0]
[0.0, 0.0, 0.0]
Anatase: [-0.076491340000000463, -0.04972126500000229]
[-0.02236320999999819, 0.0051534599999989439]
```

```
[0.032760174999999947, 0.060414525000002328, 0.087801800000001151]
Columbite: [-0.0044369400000014991, 0.010715344999997711]
[0.023467515000000105, 0.033255520000000871]
[0.041004615000002076, 0.047393387500001438, 0.053046765000001272]
Brookite: [-0.039122715000001307, -0.018795817500002698]
[-0.00028209749999774658, 0.017352007500001321]
[0.034144902499999574, 0.05064824000000101, 0.0671415599999996]

AM05 Functional:
Rutile: [0.0, 0.0]
[0.0, 0.0]
[0.0, 0.0, 0.0]
Anatase: [-0.078308700000000897, -0.049587325000000959]
[-0.020423039999997172, 0.0089234000000004698]
[0.03828767499999941, 0.067754265000001368, 0.096866719999997741]
Columbite: [-0.0055456800000008855, 0.0092836999999974523]
[0.020877130000002353, 0.030148307499999305]
[0.037715014999999852, 0.044340922500001767, 0.050151567499998606]
```

Upon modification of the above queries in the org-mode source file introduced in Section 1 of this Supporting Information document, the four example queries shown above can serve as templates through which any combination of BO_2 polymorphic trend data featuring U incrementation can be plotted. With some modification of the post-processing and plotting code above, hybrid functional and linear response U data can be superimposed on this data in the plots above as well. This is illustrated in Sections 4.1.4 and 4.2.2 of this document.

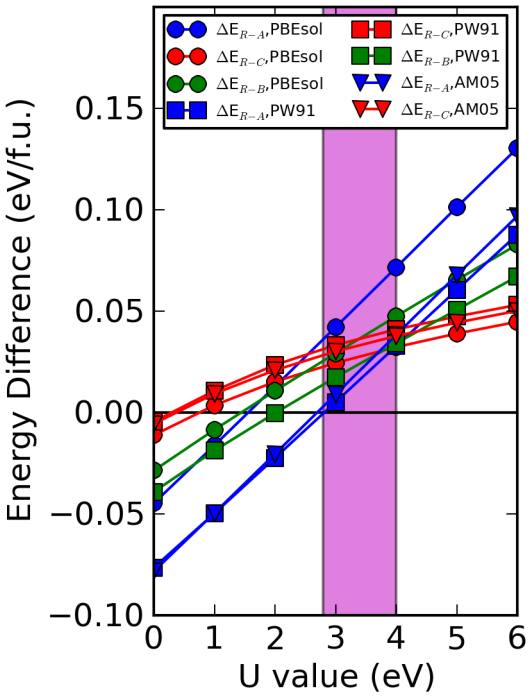

Figure S5

4.1.3 Validation of Columbite as an Upper Bound for the Metastable Range in TiO_2

The Columbite polymorph can alternatively be classified via the α-PbO_2 structure (*Pbcn* point symmetry, or *Pcan* point symmetry in an alternate environment)[S22] or as the scrutinyite polymorph in single metal cation systems.[S23] Several first-principles sources indicate that Columbite is the most stable unstable phase when subjected to incrementally increasing pressure, though that Columbite is also less stable than the stable bulk Rutile and Anatase phases.[S22,S24,S25] Experimental results consistently indicate that Columbite is the first phase to be accessed with the application of pressure in compression cycles,[S26–S31] as well as being the last phase accessed in decompression cycles prior to reacquiring Rutile[S25,S32] at standard temperature and pressure conditions. However, in the absence of the application of pressure to bulk TiO_2 phases, comprehensive validation that Columbite is the most stable unstable phase over the span of electron structure correction conditions tested has not been accom-

plished to the knowledge of the authors of this article. In the case of Hubbard U calculations, this can be verified by evaluating the relative energetic favorability (w.r.t. Rutile) of known TiO_2 polymorphs stabilized at higher pressures over the ranges of U values tested in this article (U = 0.0-6.0 eV). These polymorphs, which are detailed in past articles,[S22,S24,S25] include Columbite (C), Baddeleyite (Ba), Cotunnite (Co), Pyrite (P), and Fluorite (F). An evaluation of the relative stabilities of these polymorphs that validates that Columbite is the most stable unstable polymorph with no applied pressure, which is accomplished over a range of U values both consistent with that studied (U = 0.0-6.0 eV) and the structural limitations of Baddeleyite (see further below), is completed below. The relational algebraic expression and complementary MySQL query used to generate this plots is transcribed below, in addition to the plot itself, which illustrates the relative stability of Columbite with respect to the other expected high pressure polymorphs:

$$\Pi_{Morph,UValue,EOpt,NumAtoms}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Columbite')$$

$$\sigma \vee (Morph =' Baddeleyite') \vee (Morph =' Cotunnite')$$

$$\sigma \vee (Morph =' Fluorite') \vee (Morph =' Pyrite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query high pressure polymorph
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, Uvalue, Eopt, atomnum = [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
```

```
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Columbite' or s.Morph='Baddeleyite' or
s.Morph='Cotunnite' or s.Morph='Fluorite' or s.Morph='Pyrite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Functional='PBE'
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Polymorph += [a]

    datapts_dict[row] = datapts_list

Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Morph_list:
    SortElist[i] = {}
    EBsiteMatch[i] = []
    del_list = []

    for j in datapts_dict:
        if j[0] == i:
            SortElist[ j[0] ][ j[1] ] = float( j[2] ) / float( j[3] )
            del_list.append( datapts_dict[j] )
        else:
            pass

    for l in del_list:
        del l

for i in Morph_list:
    U_list = SortElist[i].keys()
```

```
    U_list.sort()
    EmapU_list = []

    for j in U_list:
        EmapU_value = SortElist[i][j] - SortElist[WRT_Morph][j]
        EmapU_list.append( EmapU_value )

    EBsiteMatch[i].append( EmapU_list )
    EBsiteMatch[i].append( U_list )

ax = plt.gca()

print "PBE Functional, Ti O pseudopotentials:"
print "Rutile: " + str(EBsiteMatch['Rutile'][0][0:2])
print str(EBsiteMatch['Rutile'][0][2:4])
print str(EBsiteMatch['Rutile'][0][4:])
print "Columbite: " + str(EBsiteMatch['Columbite'][0][0:2])
print str(EBsiteMatch['Columbite'][0][2:4])
print str(EBsiteMatch['Columbite'][0][4:])
print "Baddeleyite: " + str(EBsiteMatch['Baddeleyite'][0][0:2])
print str(EBsiteMatch['Baddeleyite'][0][2:4])
print str(EBsiteMatch['Baddeleyite'][0][4:])
print "Cotunnite: " + str(EBsiteMatch['Cotunnite'][0][0:2])
print str(EBsiteMatch['Cotunnite'][0][2:4])
print str(EBsiteMatch['Cotunnite'][0][4:])
print "Fluorite: " + str(EBsiteMatch['Fluorite'][0][0:2])
print str(EBsiteMatch['Fluorite'][0][2:4])
print str(EBsiteMatch['Fluorite'][0][4:])
print "Pyrite: " + str(EBsiteMatch['Pyrite'][0][0:2])
print str(EBsiteMatch['Pyrite'][0][2:4])
print str(EBsiteMatch['Pyrite'][0][4:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Rutile'][1], EBsiteMatch['Rutile'][0], 'k-')

plt.plot(EBsiteMatch['Columbite'][1], EBsiteMatch['Columbite'][0],
         'bo-', label = r'$\Delta$E$_{R-C}$')
plt.plot(EBsiteMatch['Baddeleyite'][1], EBsiteMatch['Baddeleyite'][0],
         'ro-', label = r'$\Delta$E$_{R-Ba}$')
plt.plot(EBsiteMatch['Cotunnite'][1], EBsiteMatch['Cotunnite'][0],
         'go-', label = r'$\Delta$E$_{R-Co}$')
```

```
plt.plot(EBsiteMatch['Fluorite'][1], EBsiteMatch['Fluorite'][0],
         'co-', label = r'$\Delta$E$_{R-F}$')
plt.plot(EBsiteMatch['Pyrite'][1], EBsiteMatch['Pyrite'][0],
         'mo-', label = r'$\Delta$E$_{R-P}$')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 1.2) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper center', prop={'size':9}, ncol = 2)

plt.gcf().subplots_adjust(left=0.20)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-RCBaCoFP-PBE' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, Ti O pseudopotentials:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Columbite: [-0.0041790349999999421, 0.012991809999999049]

[0.025531667499997468, 0.034880245000000087]

[0.042688772500000027, 0.049023697500000907, 0.054641072500000831]

Baddeleyite: [0.068526564999999096, 0.11540673499999698]

[0.15783202000000074, 0.1968255900000031]

[0.23244494500000101, 0.26398295999999988, 0.29190553500000149]

Cotunnite: [0.69656986499999718, 0.73965338749999887]

[0.77861947749999771, 0.81588487500000184]

[0.84990483500000025, 0.87976105250000103, 0.90678868000000179]

Fluorite: [0.76067140999999694, 0.73158858999999765]

[0.70339385499999807, 0.67846820000000108]

[0.65804550500000047, 0.64093578499999992, 0.62901731000000183]
```

```
Pyrite: [0.62867331499999679, 0.59953638499999684]
[0.56888037499999911, 0.53874001249999992]
[0.50997492750000006, 0.48115787500000096, 0.45410749500000236]
```

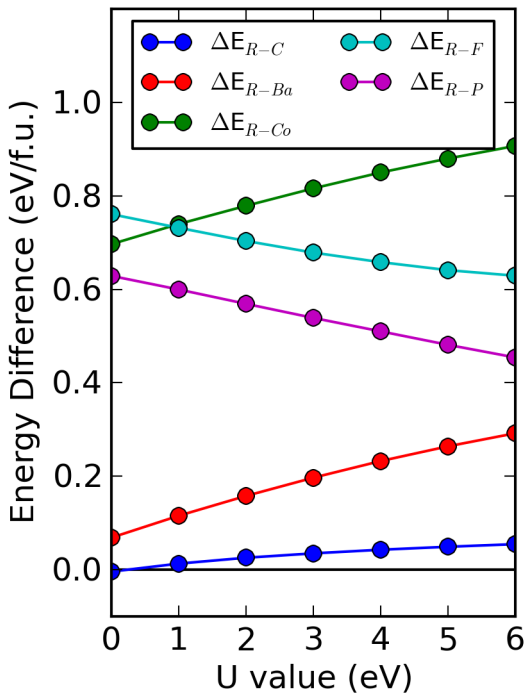

Figure S6

In the plot shown above, note that calculated results for the Baddeleyite polymorph are only physically representative of experimental structures up to a U value of 4.0 eV. Unlike the other polymorphs tested, when the ground state fitted energies (E_0) and volumes (V_0) of Baddeleyite are evaluated over a range of unit cell volumes encompassing its experimentally predicted volume of 112.24 $\AA^3$,[S22] at least two energetic minima can be resolved from visual inspection of its E-V curve at $U = 0.0$ eV. This effect persists as U is increased, as is illustrated in examples calculated at $U = 1.0$ eV and $U = 4.0$ eV that are shown below. For each Baddeleyite E-V curve at a particular U value, there exists a single, apparently local energetic minimum occurring at volumes nearer the experimentally predicted volume than other minima occurring at lower energies and higher cell volumes. Despite the apparent

presence of these lower energy, higher cell volume minima, the well-defined higher energy, lower cell volume minima are employed to evaluate Rutile-Baddeleyite relative energetics (all of these are listed under "EOpt"). The relative energetic phase stability of Baddeleyite is evaluated using these higher energies due to the unphysicality of structural results observed around apparent lower energetic minima.

In TiO_2, the Baddeleyite polymorph observes experimentally resolved lattice vectors of a = 4.989 Å, b = 5.049 Å, and c = 5.149 Å in an alternate environment used in this study.[S22] As cell volume increases towards or beyond lower volume energetic minima at all values of U, the b/a ratio lowers, ultimately becoming less than one at each U value when approaching lower energy, higher cell volume minima. Considering that the relative lengths of lattice constants have reversed at these higher volumes, the structures resolved at these volumes do not represent experimental Baddeleyite structures physically. Therefore, a criterion of b/a >1 has been imposed to resolve physically realistic energetic minima for the Baddeleyite polymorph. At values of $U \leq 4.0$, physically realistic relative lengths of lattice constants are achieved at the higher energy, lower volume minimum of Baddeleyite in all E-V curves. However, in the U = 5.0 and 6.0 eV E-V curves, non-physical structures are resolved at all observed energetic minima. Thus, Baddeleyite is only evaluated up to U = 4.0 eV. Despite this limitation imposed by the structural criterion used to resolve realistic Baddeleyite structures, Columbite has still been shown to be the most stable unstable polymorph of all TiO_2 polymorphs tested. The plots illustrating this conclusion, in addition to the relational algebraic expression and complementary MySQL query used to generate these plots, are transcribed below. Note that, for each value of U, lattice vectors are taken from the calculation with a value of "Energy" closest to a corresponding value of "EOpt":

$$\Pi_{Energy,LDAUU,Eopt,PrimVec1,PrimVec2}$$

$$\sigma \wedge [(Morph =' Baddeleyite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange$$

$$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP \bowtie ParametersPosFinalVASP]$$

```
#+caption: MySQL query high pressure Baddeleyite criterion
import matplotlib.pyplot as plt
import sqlite3
import numpy as np

db = sqlite3.connect('ITEOO_data.sqlite')

U_vals = []
EtoEOptU_dict = {}
NULL_CHAR = '-'
SColon_CHAR = ';'
WRT_U = 0.
Morph_list = ['Baddeleyite']

for row in db.execute('''
select distinct cre.Energy, cre.EOpt from Structure as s
```

```
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feu.EOpt
where (s.Morph='Baddeleyite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and (mt.Functional='PBE')
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    a,b = row
    try:
        EtoEOptU_dict[b].append(a)
    except KeyError:
        EtoEOptU_dict[b] = []
        EtoEOptU_dict[b].append(a)

Emin_list = []
for i in EtoEOptU_dict.keys():
    Ediff_list = []

    for j in EtoEOptU_dict[i]:
        Ediff_value = abs( float(i) - float(j) )
        Ediff_list.append( Ediff_value )

    Emindex = Ediff_list.index( min( Ediff_list ) )
    Emin_list.append( EtoEOptU_dict[i][ Emindex ] )

datapts_dict = {}
for row in db.execute('''
select distinct input.Energy, input.LDAUU, pos.PrimVec1, pos.PrimVec2 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feu.EOpt
```

```
inner join ParametersPosFinalVASP as pos on pos.Energy=cre.Energy
inner join ParametersInputVASP as input on input.Energy=pos.Energy
where (s.Morph='Baddeleyite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and (mt.Functional='PBE')
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    U_vals += [b]
    if a in Emin_list:
        datapts_dict[row] = datapts_list

U_vals_list = list( set( U_vals ) )
SortElist = {}
EBsiteMatch = {}
Ulabel_list = []
for i in Morph_list:
    SortElist[i] = {}
    EBsiteMatch[i] = []

    for j in U_vals_list:
        Utype_val = str(j).split(SColon_CHAR)[0]

        Ulabel_list.append( Utype_val )

        SortElist[i][ Utype_val ] = {}

        del_list = []

        for k in datapts_dict:
            if k[1] == j:
                a_string = str( k[2] ).split(SColon_CHAR)
                c_string = str( k[3] ).split(SColon_CHAR)
```

```
                a_vec = np.zeros( len(a_string) )
                c_vec = np.zeros( len(c_string) )

                for l in range( len(a_string) ):
                    a_vec[l] = float( a_string[l] )
                    c_vec[l] = float( c_string[l] )

                a_dot = np.dot( a_vec, a_vec )
                c_dot = np.dot( c_vec, c_vec )

                ca_ratio = ( c_dot / a_dot )**(0.5)
                SortElist[ i ][ Utype_val ] = ca_ratio
                del_list.append( datapts_dict[k] )
            else:
                pass

        for l in del_list:
            del l

for i in Morph_list:
    U_list = SortElist[i].keys()
    camapU_list = []
    U_list.sort()

    for k in U_list:
        camapU_value = SortElist[i][k]
        camapU_list.append( camapU_value )

    EBsiteMatch[i].append( camapU_list )
    EBsiteMatch[i].append( U_list )

print "U values: " + str(EBsiteMatch['Baddeleyite'][1])
print "b/a ratios: " + str(EBsiteMatch['Baddeleyite'][0][0:3])
print str(EBsiteMatch['Baddeleyite'][0][3:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Baddeleyite'][1], EBsiteMatch['Baddeleyite'][0], 'bo-', label = r'$\Delta$b/a(Baddeleyite)')

plt.xlabel( 'U value (eV)' )
plt.ylabel('b/a ratio')
plt.legend(loc = 'lower left', prop={'size':9})
```

```
plt.gcf().subplots_adjust(left=0.25)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-baratio-Ba-PBE' + '.' + ext, dpi=300)
plt.clf()
```

```
U values: ['0', '1.00', '2.00', '3.00', '4.00', '5.00', '6.00']
b/a ratios: [1.0112192506637021, 1.0115589103973111, 1.0090726957847269]
[1.005478860956093, 1.0038897458598666, 0.9988197707526224, 0.99447933299670865]
```

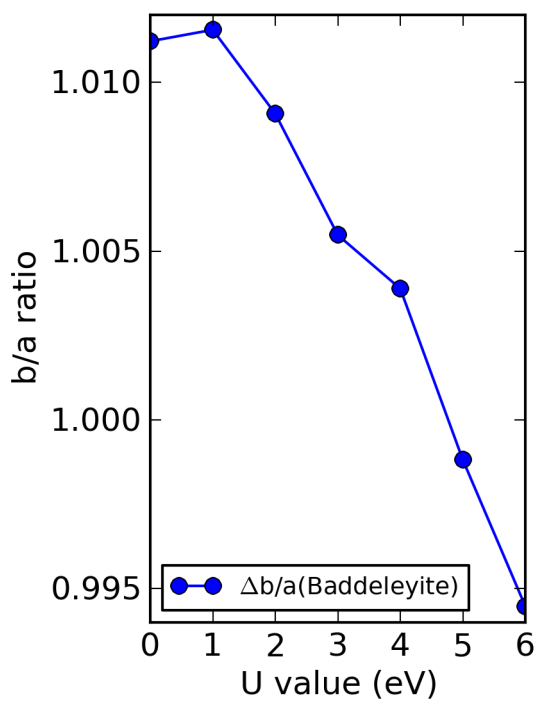

Figure S7

Additionally, the Quantum Espresso software package is used to calculate first-principles U values for the Rutile, Anatase, and Columbite polymorphs via linear response theory. A region of metastability is formed by the intersection of the Rutile-Columbite relative

energetic curve with the x axis, illustrating the U dependent energetic ranges within which metastable polymorphs (i.e.: Anatase and Brookite) are formed. In order to validate that the metastable range reviewed in this study via VASP can largely have first-principles linear response values of U from QE directly applied to it, the metastable regions formed by VASP and QE are comparatively reviewed in the plot below. As can be shown in the plot below, the Rutile-Columbite profiles generated from VASP PAW pseudopotentials and QE Ultrasoft pseudopotentials are within strong qualitative and quantitative agreeemnt, indicating that the metastability regions formed in each software package and pseudopotential pair are comparable. The plot illustrating this conclusion, in addition to the relational algebraic expression and complementary MySQL query used to generate this plot, are transcribed below:

$$\Pi_{Morph,Software,UValue,EOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Columbite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti') \vee (PseudoB =' US - Ti - pv - sv')]$$

$$\sigma \wedge [(PseudoO =' PAW - O') \vee (PseudoO =' US - O')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP') \vee (Software =' QE')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```python
#+caption: MySQL query VASP vs. QE Rutile Columbite
import matplotlib.pyplot as plt
import sqlite3

convRytoeV = 13.605698066

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, Calculator, Uvalue, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select s.Morph, mt.Software, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on ( mt.MID=c1.MID and mt.SID=c1.SID )
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Columbite')
and c1.At1='Ti'
and (mt.PseudoB='PAW-Ti' or mt.PseudoB='US-Ti-pv-sv')
and (mt.PseudoO='PAW-O' or mt.PseudoO='US-O')
and mt.Functional='PBE'
and (mt.Software='VASP' or mt.Software='QE')
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    Polymorph += [a]
    Calculator += [b]

    datapts_dict[row] = datapts_list

Morph_list = list( set( Polymorph ) )
Calculator_list = list( set( Calculator ) )
SortElist = {}
EBsiteMatch = {}
```

```
42
43  for i in Morph_list:
44      SortElist[i] = {}
45      EBsiteMatch[i] = {}
46
47      for j in Calculator_list:
48          SortElist[i][j] = {}
49          EBsiteMatch[i][j] = []
50          del_list = []
51
52          for k in datapts_dict:
53              if k[0] == i and k[1] == j:
54                  SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
55                  del_list.append( datapts_dict[k] )
56              else:
57                  pass
58
59          for l in del_list:
60              del l
61
62  for i in Morph_list:
63      for j in Calculator_list:
64          U_list = SortElist[i][j].keys()
65          U_list.sort()
66          EmapU_list = []
67
68          for k in U_list:
69              if j == 'QE':
70                  EmapU_value = ( SortElist[i][j][k] - SortElist[WRT_Morph][j][k] ) * convRytoeV
71                  EmapU_list.append( EmapU_value )
72              else:
73                  EmapU_value = SortElist[i][j][k] - SortElist[WRT_Morph][j][k]
74                  EmapU_list.append( EmapU_value )
75
76          EBsiteMatch[i][j].append( EmapU_list )
77          EBsiteMatch[i][j].append( U_list )
78
79  ax = plt.gca()
80
81  print "PBE Functional, VASP Calculator:"
82  print "Rutile: " + str(EBsiteMatch['Rutile']['VASP'][0])
```

```
print "Columbite: " + str(EBsiteMatch['Columbite']['VASP'][0][0:2])
print str(EBsiteMatch['Columbite']['VASP'][0][2:4])
print str(EBsiteMatch['Columbite']['VASP'][0][4:]) + "\n"

print "PBE Functional, QE Calculator:"
print "Rutile: " + str(EBsiteMatch['Rutile']['QE'][0])
print "Columbite: " + str(EBsiteMatch['Columbite']['QE'][0][0:2])
print str(EBsiteMatch['Columbite']['QE'][0][2:4])
print str(EBsiteMatch['Columbite']['QE'][0][4:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Rutile']['VASP'][1], EBsiteMatch['Rutile']['VASP'][0], 'k-')

plt.plot(EBsiteMatch['Columbite']['VASP'][1], EBsiteMatch['Columbite']['VASP'][0],
         'bo-', label = r'$\Delta$E$_{R-C,VASP}$')
plt.plot(EBsiteMatch['Columbite']['QE'][1], EBsiteMatch['Columbite']['QE'][0],
         'ro-', label = r'$\Delta$E$_{R-C,QE}$')

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.02, 0.08) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-VASPQE-PBE' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, VASP Calculator:

Rutile: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Columbite: [-0.0041790349999999421, 0.012991809999999049]

[0.025531667499997468, 0.034880245000000087]

[0.042688772500000027, 0.049023697500000907, 0.054641072500000831]


PBE Functional, QE Calculator:
```

```
Rutile: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Columbite: [-0.015910163176169306, -0.0024395016634164847]
[0.010531830730182714, 0.022543621268282393]
[0.033032253906866593, 0.041712349130765561, 0.048457373946970188]
```

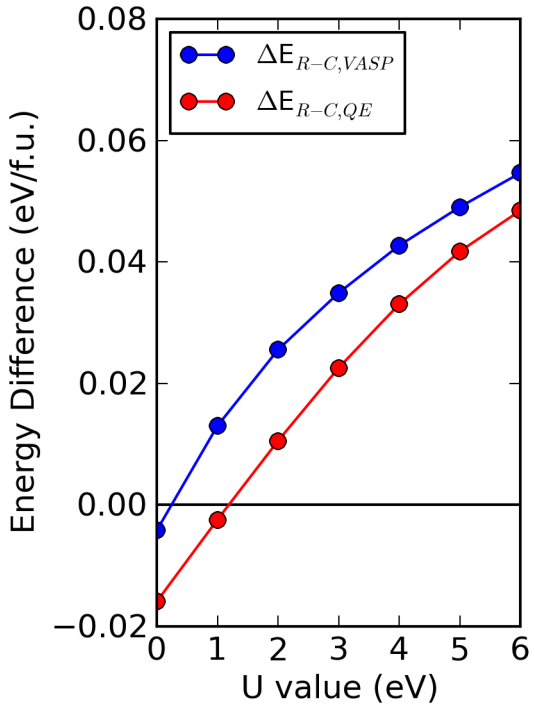

Figure S8

Given that the Rutile-Columbite formation energetics evaluated in VASP and QE using the PBE functional largely overlap, the first-principles Hubbard U calculation results derived using either software package (calculator) should be largely transferable in their application to corresponding energetic data. The transferability of the first-principles U values will be evaluated in the next subsection, Section 4.1.4.

4.1.4 Standard vs. Self-consistent Linear Response

The first-principles method of linear response theory was applied to several TiO_2 polymorphic systems, in order to assess the extent to which formation energy ordering relating

to these polymorphs could be predicted, given known experimental outcomes (e.g.: Rutile is the most stable bulk TiO_2 polymorph, Columbite is the least stable of the four studied, etc.). The types of systems evaluated in this study reflect the results of Hubbard U incrementation calculations, in which pseudopotential and functional were varied to reveal that, with respect to standard pseudopotentials and the PBE functional, the implementation of Ti_pv pseudopotentials, Ti_sv pseudopotentials, and the PBEsol functional significantly affected Rutile-Anatase formation energetic ordering without precluding it. Thus, the systems studied incorporate standard PAW pseudopotentials and the PBE functional (Rutile, Anatase, Columbite, and Brookite polymorphs), Ti_pv pseudopotentials and the PBE functional (Rutile and Anatase polymorphs), Ti_sv pseudopotentials and the PBE functional (Rutile and Anatase polymorphs), and the standard pseudopotentials with the PBEsol functional (Rutile and Anatase polymorphs). Consistent with experimental and first-principles expectations,[S3,S33] the TiO_2 polymorphs were studied as paramagnetic (PM), spin-polarized systems (ISPIN = 2, MAGMOM values are not ordered and have an average value of 0). Past research, which has performed non-magnetic (NM) Rutile TiO_2 calculations (completed with ISPIN = 1 in VASP),[S34] has also been vindicated in this work, as modeled PM and NM Rutile and Anatase TiO_2 systems have observed equivalent U values as results when using standard pseudopotentials and the PBE functional. Example input files depicting these systems will later be used to illustrate how first-principles U values can be calculated via linear response theory in VASP, while showing that the correctly executed consideration of magnetization achieves equivalent results in PM and NM TiO_2 systems.

Regardless of the magnetic state observed or induced in a system, linear response calculations performed on a particular periodic system are accomplished by perturbing the orbital occupations of d or f-orbital containing atoms on non-equivalent atomic sites.[S35,S36] For Rutile, Anatase, Columbite, and Brookite TiO_2 systems, this requires that Ti cation orbitals be perturbed on the sites at the $2a$, $4a$, $4c$, and $8c$ Wyckoff positions, respectively, as each of these TiO_2 polymorph structures has only one site symmetry occupied by a cation.[S19,S36]

Given perturbations on these sites, symmetry relationships can be used to reproduce an entire response matrix (chi), which serves as a constituent part of a calculated U value. In particular, the bare (or initial, chi_0) and converged (or final, chi_0) response matrices of a U calculation represent the initial and final responses to the perturbation of electronic charge density in pertinent orbitals, while the responses themselves are represented by changes in the summed electronic occupations of those orbitals. Standard linear response approaches calculate the difference of these responses to account for spurious electron-electron interactions in semiconducting and insulating materials, generally using the structural and electronic characteristics of an electronic ground state (DFT ground state) not accounting for the addition of U as calculation inputs. In other words, an input U value (U_{in}) of 0 eV is applied to achieve an output U value (U_{out}) greater than 0 eV, even though the inputted structural characteristics, electronic structure, and thus orbital occupational information are not necessarily consistent with those of the outputted electronic ground state (DFT + U ground state). This inconsistency in the electronic structures of inputted and outputted systems can be accounted for, namely with a self-consistent linear response approach, which will be explained in greater detail later.[S37–S39] The standard linear response calculation procedure used to calculate U_{out} without accounting for the possible inconsistency in electronic ground states is transcribed in Equation 1 below:[S37]

$$U_{out} = \chi_0^{-1} - \chi^{-1} \tag{1}$$

Using VASP, this calculation can be completed via a four step procedure that employs the "LDAUTYPE = 3" setting revealed on the official VASP forum.[S21,S40] Firstly, take the relevant TiO_2 structures achieved after variable cell volume relaxation using the "ISIF = 3" setting in Subsection 4.1.1, produce an appropriate supercell representation of that structure, and then complete an equivalent relaxation on that supercell representation. An example of this calculation step using the $2 \times 2 \times 2$ supercell representation of TiO_2 Rutile with standard pseudopotentials and the PBE functional is depicted below, illustrated via the

presentation of the INCAR, KPOINTS, and POSCAR files needed to complete this step:

Listing 11: INCAR Rutile PBE PAW_standard VASP Relaxation Step

```
ISTART = 0 ; ICHARG = 2
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 1

IBRION = 1
EDIFF = 5E-05; EDIFFG = -0.02

MAXMIX = -100
NELMIN = 5
NELM = 200
NSW = 100

ISPIN = 2
ISIF = 3

LDAU = .TRUE.
LDAUTYPE = 2
LDAUL = 2 -1
LDAUU = 0.00 0.00
LDAUJ = 0.00 0.00

LDAUPRINT= 2
LASPH = .TRUE.
LMAXMIX = 4
LORBIT = 11
```

Listing 12: KPOINTS Rutile PBE PAW_standard VASP Relaxation Step

```
4x4x4
0
Monkhorst-Pack
4 4 4
0 0 0
```

Listing 13: POSCAR Rutile PBE PAW_standard VASP Relaxation Step

```
Ti O
4.66
1   0   0
0   1   0
0   0   0.643993896
2 4
direct
  0.000000000   0.000000000   0.000000000
  0.500000000   0.500000000   0.500000000
  0.304880731   0.304880731   0.000000000
 -0.304880731  -0.304880731   0.000000000
  0.804875587   0.195124413   0.500000000
 -0.804875587  -0.195124413   0.500000000
```

The calculation above was completed for the PM case, as is indicated by the presence of "ISPIN = 2" in the INCAR file. Completion of the NM case (with "ISPIN" = 1 in the INCAR file) reveals equivalent results, as will be shown later for both Rutile and Anatase polymorphs. The results achieved using the input files shown above are not directly related to any final calculated values in the paper, thus the input and output files associated with the step above are not present in the database. Creation of the supercells needed to complete this first step can be accomplished via the ase.io.vasp module and associated supercell command found in the Atomic Simulation Environment (ASE) software package. The command, which is reproduced in the code below, is performed on a POSCAR file in the directory containing the code itself:[S41]

Listing 14: Python commands used to generate supercells

```
import ase.io.vasp
cell = ase.io.vasp.read_vasp("POSCAR")
ase.io.vasp.write_vasp("POSCAR_2",cell*(2,2,2), label='Ti O', direct=True, sort=True)
```

In the second step of this four step procedure, the CONTCAR of the structurally relaxed system from the first step is imported into a second calculation, namely one that performs solely electronic relaxation at a higher precision specified by the "EDIFF" tag. This step in the procedure is used to converge a CHGCAR file used to store orbital occupation information, which is subsequently imported into the third step of this calculation procedure:[S21,S40]

Listing 15: INCAR Rutile PBE PAW_standard VASP Convergence Step

```
ENCUT = 600
ISMEAR = 0
SIGMA = 0.05

ISYM = 1
EDIFF = 1E-06
ISPIN = 2

LDAU = .TRUE.
LDAUTYPE = 2
LDAUL = 2 -1 -1
LDAUU = 0.00 0.00 0.00
LDAUJ = 0.00 0.00 0.00

LDAUPRINT= 2
LASPH = .TRUE.
LMAXMIX = 4
LORBIT = 11
```

Listing 16: KPOINTS Rutile PBE PAW_standard VASP Convergence Step

```
4x4x4
0
Monkhorst-Pack
4 4 4
0 0 0
```

Listing 17: POSCAR Rutile PBE PAW_standard VASP Convergence Step

```
Ti Ti O
   1.00000000000000
     9.3217564470847751    0.0000000000000000    0.0000000000000000
     0.0000000000000000    9.3217564470847751    0.0000000000000000
     0.0000000000000000    0.0000000000000000    5.9372793380803959
    1   15   32
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.5000000000000000
  0.7500000000000000  0.7500000000000000  0.7500000000000000
  0.2500000000000000  0.2500000000000000  0.2500000000000000
  0.0000000000000000  0.0000000000000000  0.5000000000000000
  0.7500000000000000  0.7500000000000000  0.2500000000000000
  0.0000000000000000  0.5000000000000000  0.5000000000000000
  0.2500000000000000  0.2500000000000000  0.7500000000000000
  0.2500000000000000  0.7500000000000000  0.7500000000000000
  0.7500000000000000  0.2500000000000000  0.7500000000000000
  0.5000000000000000  0.0000000000000000  0.5000000000000000
  0.0000000000000000  0.5000000000000000  0.0000000000000000
  0.2500000000000000  0.7500000000000000  0.2500000000000000
  0.7500000000000000  0.2500000000000000  0.2500000000000000
  0.5000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.0000000000000000
  0.0976626291255371  0.9023373708744629  0.7500000000000000
  0.4023373708744629  0.5976626291255371  0.7500000000000000
  0.9023373708744629  0.5976626291255371  0.7500000000000000
  0.6523349025344345  0.1523349025344345  0.0000000000000000
  0.8476650974655655  0.3476650974655655  0.0000000000000000
  0.9023373708744629  0.0976626291255371  0.2500000000000000
  0.5976626291255371  0.4023373708744629  0.2500000000000000
  0.6523349025344345  0.1523349025344345  0.5000000000000000
  0.3476650974655655  0.8476650974655655  0.5000000000000000
  0.8476650974655655  0.3476650974655655  0.5000000000000000
  0.5976626291255371  0.4023373708744629  0.7500000000000000
  0.6523349025344345  0.6523349025344345  0.0000000000000000
  0.8476650974655655  0.8476650974655655  0.0000000000000000
  0.9023373708744629  0.5976626291255371  0.2500000000000000
  0.5976626291255371  0.9023373708744629  0.2500000000000000
  0.6523349025344345  0.6523349025344345  0.5000000000000000
  0.8476650974655655  0.8476650974655655  0.5000000000000000
  0.9023373708744629  0.0976626291255371  0.7500000000000000
  0.1523349025344345  0.6523349025344345  0.5000000000000000
  0.5976626291255371  0.9023373708744629  0.7500000000000000
  0.3476650974655655  0.3476650974655655  0.5000000000000000
  0.0976626291255371  0.9023373708744629  0.2500000000000000
  0.4023373708744629  0.5976626291255371  0.2500000000000000
  0.3476650974655655  0.8476650974655655  0.0000000000000000
  0.1523349025344345  0.1523349025344345  0.5000000000000000
  0.1523349025344345  0.1523349025344345  0.0000000000000000
  0.1523349025344345  0.6523349025344345  0.0000000000000000
  0.0976626291255371  0.4023373708744629  0.2500000000000000
  0.4023373708744629  0.0976626291255371  0.7500000000000000
  0.0976626291255371  0.4023373708744629  0.7500000000000000
  0.3476650974655655  0.3476650974655655  0.0000000000000000
  0.4023373708744629  0.0976626291255371  0.2500000000000000
```

Note that, in all steps of this procedure directly relating to the calculation of perturbations, the perturbed cation must be distinguished as a separate atomic species of the same composition as matching non-perturbed species in both atomic type (atom indices of the

POSCAR file) and POTCAR designations. Thus, for TiO_2 species, atomic type based information must be represented through three indices (e.g.: Ti Ti O) as oppposed to two indices (e.g.: Ti O). In the third step of this procedure, CHGCAR information created in the second step is imported into a non-selfconsistent calculation completed using "ICHARG = 11". The final orbital occupations of this non-selfconsistent calculation are then used to produce the initial response (chi_0) needed to calculate a first-principles U value of interest. At this point in the procedure, the LDAUTYPE tag must have an associated value of '3', while LDAUU and LDAUJ tags are now used to vary the quantity of spin-up and spin-down perturbation contributions applied to a particular response calculation.[S21,S40] In the calculations performed in this paper, perturbations are uniformly applied to spin-up and spin-down contributions over a charge potential range of α = -0.150 – 0.150 eV in 0.05 eV increments, in order to verify the linearity of the occupation perturbations over a sufficient range and sampling of perturbations. An example of the bare response INCAR file (chi_0), completed at a perturbation potential (α) of 0.10 eV performed in the third step of this procedure, is transcribed below (the KPOINTS and POSCAR files match that shown in the "Convergence" step above):[S21,S40]

Listing 18: INCAR Rutile PBE PAW_standard VASP Initial Perturbation Step

```
ICHARG = 11
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 0

ISPIN = 2

EDIFF = 1E-07

LDAU = .TRUE.
LDAUTYPE = 3
LDAUL = 2 -1 -1
LDAUU = 0.10 0.00 0.00
LDAUJ = 0.10 0.00 0.00

LDAUPRINT = 2
LASPH = .TRUE.
LMAXMIX = 4
LORBIT = 11
```

Lastly, a self-consistent calculation is completed using calculation parameters matching that of the third step, except without implementing the converged "CHGCAR" file from the second step. An example of this final response INCAR file (chi), completed at a perturbation potential (α) of 0.10 eV performed in the fourth and final step of this procedure, is transcribed below (the KPOINTS and POSCAR files match that shown in the "Convergence" step above). Note that the perturbation range and incrementation (α = -0.150 – 0.150 eV, 0.05 eV increments) seen in the third step is also employed in this step:[S21,S40]

Listing 19: INCAR Rutile PBE PAW_standard VASP Final Perturbation Step

```
ENCUT = 600
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 0

ISPIN = 2

EDIFF = 1E-07

LDAU = .TRUE.
LDAUTYPE = 3
LDAUL = 2 -1 -1
LDAUU = 0.10 0.00 0.00
LDAUJ = 0.10 0.00 0.00

LDAUPRINT = 2
LASPH = .TRUE.
LMAXMIX = 4
LORBIT = 11
```

The input files above, which correspond to the case of the PM TiO_2 Rutile polymorph, yielded results that can be compared to their NM analogues using a PBE functional and standard pseudopotentials. The effects of incorporating or neglecting spin polarization and magnetism during different steps of the procedure above were evaluated to assess the impact of considering PM and NM states in TiO_2 systems. In the case of Rutile, calculations were completed while neglecting spin polarization and magnetism (NM case) in both the second ("convergence") and third/fourth steps ("perturbation") above ("NS-NS"), incorporating spin polarization and magnetism (PM case) in the "convergence" step but ignoring it in "perturbation" steps ("S-NS"), performing a NM "convergence" step and PM "perturbation" step ("NS-S"), and completing PM calculations across both steps ("S-S"). The results of

these calculations, which are queried, displayed, and effectively compared below, reveal that spin polarization cannot be ignored in a convergence calculation and then incorporated into perturbation calculations due to the unphysical magnetic moments achieved when performing this combination ("NS-S"). However, in the case of TiO_2 polymorphs, these combinations of calculations achieved otherwise equivalent perturbed atom occupation and magnetism results, thus NM and PM modeled TiO_2 systems would possess the same value of U (for "NS-NS", "S-NS", and "S-S"), as is shown below:

$$\Pi_{CalcType,AtomIndex,DChargeRHigh,DMagP}$$

$$\sigma[(Morph =' Rutile')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(CalcType =' Chi0NS - NS') \vee (CalcType =' Chi0NS - S')$$

$$\sigma \vee (CalcType =' Chi0S - NS') \vee (CalcType =' Chi0S - S')$$

$$\sigma \vee (CalcType =' ChiNS - NS') \vee (CalcType =' ChiNS - S')$$

$$\sigma \vee (CalcType =' ChiS - NS') \vee (CalcType =' ChiS - S')]$$

$$\sigma \wedge [(AtomIndex =' 1')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie CalcResultLRCalc$$

$$\bowtie ParametersResponseVASP]$$

```
#+caption: MySQL query Linear Response NS-NS NS-S S-NS S-S
import matplotlib.pyplot as plt
```

```
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Calc_type, Atom_index, d_charge, d_mag = [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select prv.CalcType, prv.AtomIndex, prv.DChargeRHigh, prv.DMagP from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join CalcResultLRCalc as crlrc on crlrc.CID=mt.CID
inner join ParametersResponseVASP as prv on crlrc.Energy = prv.Energy
where s.Morph='Rutile'
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Functional='PBE'
and mt.Software='VASP'
and crlrc.U3dIn='0'
and (prv.CalcType='ChiONS-NS' or prv.CalcType='ChiONS-S' or prv.CalcType='ChiOS-NS'
or prv.CalcType='ChiOS-S' or prv.CalcType='ChiNS-NS'
or prv.CalcType='ChiNS-S' or prv.CalcType='ChiS-NS' or prv.CalcType='ChiS-S')
and prv.AtomIndex='1'
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Calc_type += [a]
    Atom_index += [b]

    datapts_dict[row] = datapts_list

CalcType_list = list( set( Calc_type ) )
AtomIndex_list = list( set( Atom_index ) )
SortElist = {}
EBsiteMatch = {}

```

```python
for i in CalcType_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in AtomIndex_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ] = [ float( k[2] ), float( k[3] ) ]
                del_list.append( datapts_dict[k] )
            else:
                pass

        for l in del_list:
            del l

print "Linear response orbital occupancy perturbations:"
print "No spin (convergence step), no spin (perturbation step) [NS-NS]: " + str(SortElist['ChiONS-NS'][1][0])
print "No spin (convergence step), spin (perturbation step) [NS-S]: " + str(SortElist['ChiONS-S'][1][0])
print "Spin (convergence step), no spin (perturbation step) [S-NS]: " + str(SortElist['ChiOS-NS'][1][0])
print "Spin (convergence step), spin (perturbation step) [S-S]: " + str(SortElist['ChiOS-S'][1][0]) + "\n"

print "Linear response magnetic moments (perturbed atom, index 1):"
print "No spin (convergence step), no spin (perturbation step) [NS-NS]: " + str(SortElist['ChiONS-NS'][1][1])
print "No spin (convergence step), spin (perturbation step) [NS-S]: " + str(SortElist['ChiONS-S'][1][1])
print "Spin (convergence step), no spin (perturbation step) [S-NS]: " + str(SortElist['ChiOS-NS'][1][1])
print "Spin (convergence step), spin (perturbation step) [S-S]: " + str(SortElist['ChiOS-S'][1][1]) + "\n"
```

```
Linear response orbital occupancy perturbations:
No spin (convergence step), no spin (perturbation step) [NS-NS]: 2.574
No spin (convergence step), spin (perturbation step) [NS-S]: 2.592
Spin (convergence step), no spin (perturbation step) [S-NS]: 2.597
Spin (convergence step), spin (perturbation step) [S-S]: 2.597


Linear response magnetic moments (perturbed atom, index 1):
```

```
No spin (convergence step), no spin (perturbation step) [NS-NS]: 0.0
No spin (convergence step), spin (perturbation step) [NS-S]: -0.159
Spin (convergence step), no spin (perturbation step) [S-NS]: 0.0
Spin (convergence step), spin (perturbation step) [S-S]: 0.0
```

Particular linear response calculations were completed using a previous response matrix calculation methodology adapted into a Python code shown below.[S37] Each matrix is an $N \times N$ symmetric array depicting the interactions of each pair of atomic sites within a system. Given that the initial and final response matrices are constructed from the second derivatives of the U corrected total energy (E_U) with respect to perturbed orbital occupations (n), each response matrix element represents a first-order derivative relationship of orbital occupation (n) with respect to orbital perturbation (α). Given that a single atomic site is perturbed, each response contains the interaction between that pertubed site and each other atomic site in a system. When the first atom in a system is perturbed, this statement equates to populating the first column of the $N \times N$ matrix mentioned above with the first-order, linear differences (i.e.: occupation-perturbation slopes) induced in the orbital occupations of a particular site by introducing a perturbation. This serves as a representation of the interaction between a perturbed site (i) and another site (j), and is represented by Equation 2 below, namely over the widest range of perturbations allowable ($\alpha = \pm\ 0.15$ in this case). Note that the calculated U value formed from considering on-site and off-site interactions is resolved by taking the difference of the inverted chi and chi$_0$ matrix entries corresponding to on-site interactions of the perturbed cation (in this case, the (1, 1) entries in the N $\times$ N inverted chi and chi$_0$ matrices):

$$U_{out} = (\chi_0^{-1})_{i,i} - (\chi^{-1})_{i,i} \leftarrow \frac{n_0(\alpha = 0.15)_{i,i} - n_0(\alpha = -0.15)_{i,i}}{0.15 - (-0.15)} - \frac{n(\alpha = 0.15)_{i,i} - n(\alpha = -0.15)_{i,i}}{0.15 - (-0.15)} \tag{2}$$

Upon populating the first column with terms of the above form, equivalencies between

any two pairs of atomic sites are assessed in terms of several measures. For periodic crystal structures, atomic site pairs are equated via interatomic distance, spin state (magnetism), atomic composition, and Wyckoff position[S36,S37] Given that two atomic pairs share equivalent information relative to all of these measures, the first-order difference created in one equated response matrix entry (i.e.: interaction) is repeated in the other equated entry. Inversion of the matrices corresponding to the initial (chi_0) and final (chi) responses, upon population of each entire matrix in accordance with the aforementioned atomic pair equivalence criteria mentioned above, reveals the quantities that are to be subtracted to calculate a first-principles U value. In the case that the U value on the perturbed atom site is to be calculated, the difference of the inverted matrix entries corresponding to that perturbed atom site is taken to resolve the linear response calculated U. Thus, uncertainty in the calculation of U is obtained from a sequence of sources. Firstly, though each perturbation potential value is manually entered into a calculation input file and thus possesses precision negligibly limited by the VASP calculator itself, orbital occupancies are outputted from the code to their third decimal place value. Thus, each occupancy measurement has an inherent, measurement uncertainty (ϵ) of $\pm$ 0.001. Taking a first-order difference of two occupancy measurements requires addition in quadrature, therefore each response matrix entry has an inherent uncertainty of $\sqrt{(0.001)^2 + (0.001)^2} = \sqrt{2} \times 10^{-3}$ prior matrix inversion. This inherent uncertainty is scaled by the range of the perturbation ($\Delta(\alpha)$ = 0.15 - (-0.15) = 0.30) to yield the inherent uncertainty of a linear response matrix entry prior to inversion (σ_{prec}).

In order to calculate the uncertainty imposed on a single matrix entry by inversion of an entire matrix exactly, the implementation of Cayley-Hamilton theorem or a similar principle would be required to allow each matrix term contributing to uncertainty to be subjectable to common error propagation techniques.[S42,S43] In the case of matrices wherein single diagonal elements contribute predominately to matrix inversion operations, the uncertainty for those operations can be approximated as the uncertainty of inverting that single entry, namely by applying multiplication (division or inversion) in quadrature. In the linear response calcula-

tions performed in this paper, the condition of predominately large single diagonal elements can always be considered satisfied, given that the diagonal matrix terms associated with Ti cation perturbation are always at least one order of magnitude larger than implemented and observed off-diagonal terms. Thus, the uncertainty associated with matrix inversion is approximated as that of the single perturbed cation matrix entry of interest in each case. After matrix inversion, the subtraction of the perturbed cation inverted matrix entries renders a final contribution to uncertainty, namely via application of addition (subtraction) in quadrature.[S42,S43] Thus, the overall expression for the uncertainty of a U value (σ), with ϵ = 0.001 and $\Delta(\alpha)$ = 0.15 - (-0.15) = 0.30, is written as the equation set below:

$$\sigma_{prec} = \frac{\sqrt{(\epsilon)^2 + (\epsilon)^2}}{\Delta(\alpha)} \tag{3}$$

$$f_{diag,chi_0^{-1}/chi^{-1}} = \frac{n(\alpha = 0.15) - n(\alpha = -0.15)}{\Delta(\alpha)} \tag{4}$$

$$\sigma_{chi_0^{-1}/chi^{-1}} = \sqrt{(f_{diag,chi_0^{-1}/chi^{-1}})^{-2} \times \frac{(\sigma_{prec})^2}{(f_{diag,chi_0^{-1}/chi^{-1}})^2}} = \frac{\sigma_{prec}}{(f_{diag,chi_0^{-1}/chi^{-1}})^2} \tag{5}$$

$$\sigma = \sqrt{(\sigma_{chi_0^{-1}})^2 + (\sigma_{chi^{-1}})^2} \tag{6}$$

In the first expression above, the uncertainty associated with the measurement precision of any response matrix entry (σ_{prec}) is calculated. In the second expression above, the denominators of the relative errors needed for the error propagation calculations performed in the third expression is completed. These denominator values, $f_{\mathrm{diag,chi_0^{-1}/chi^{-1}}}$, correspond to the initial ($\mathrm{chi_0^{-1}}$) and final ($\mathrm{chi^{-1}}$) response contributions to the linear response calculated value of U yielded when only considering on-site, diagonal orbital occupation contributions from the 3d orbitals of the Ti cation. The difference of these contributions yields the effective value of U calculated with only on-site contributions ($\mathrm{U_{diag}}$ or "U3dOut"). Note that the

subscripted "chi_0/chi" indicates that this term is calculated separately for both the chi_0 and chi response matrices using the same formulation, albeit with occupancies corresponding to the respective initial (n_0) and final (n) responses. The third expression above features the calculation of this relative error, which results from multiplication (division or inversion) in quadrature, for the perturbed cation, using the terms developed in the first and second expressions to accomplish this task. The final expression features error associated with taking the difference of inverted initial and final response terms relating to the perturbed cation. In all calculations performed above, the measurement error and terms are all treated as mutually independent of one another and uncorrelated, as the response corresponding to each independent atomic site is dependent on the composition of the atom on its site, the structural symmetry local to the site, and its magnetic spin state in periodic systems. As a set, these are unique for each independent atomic site.[S39,S44,S45]

Calculations of all response matrices featuring these uncertainty propagation techniques, the calculation of U values, the verification of the linearity and intersection (at $\alpha = 0$) of initial (chi_0) and final (chi) perturbed cation responses, and a comparison of the U values achieved by considering (U_{out}) and ignoring (U_{diag}) off-site occupation contributions is demonstrated by the query and output below. Even though the linear response calculator developed below is prototypical and will have the capability of differentiating between atomic sites with different spin states and Wyckoff positions in future work, the current capabilities of the calculator below are suitable for PM TiO_2 systems with a single Ti site symmetry. Additionally, a constant computed background term ("comp_back") equal to -0.001 was applied to on-site O-O interactions for all systems assessed when a pertinent response matrix entry would have been otherwise been equal to 0, as this prevents the inversion of singular matrices with diagonal elements equal to zero. The value of -0.001 was also selected as a result of observing the magnitude and sign of O-O interaction response entries that were non-zero and applying that information accordingly.

$$\Pi_{PseudoB,Functional,Morph,AtomIndex,U3dOut,CalcType,Chi0[N150-P150],Chi[N150-P150]}$$

$$\Pi_{DChargeRHigh,DChargeRLow,DChargeRCenter,PChargeRHigh,PChargeRLow,PChargeRCenter}$$

$$\Pi_{Coord[1-96],PturbMax,PturbMin}$$

$$\sigma[(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti') \vee (PseudoB =' PAW - Ti - pv') \vee (PseudoB =' PAW - Ti - sv')]$$

$$\sigma \wedge [(Functional =' PBE') \vee (Functional =' PS')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' LR')]$$

$$\sigma \wedge [(U3dIn =' 0')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie CalcResultLRCalc$$

$$\bowtie ParametersResponseVASP \bowtie ParametersPosFinalVASP]$$

```
#+caption: MySQL query Linear Response Calculation
import matplotlib.pyplot as plt
import sqlite3
import numpy as np
from scipy import stats

NULL_CHAR = '-'        # NULL value in database
SColon_CHAR = ';'
comp_back = -1E-3     # computed background placed on on-diagonal unperturbed
                       # atomic indices distinguished from the perturbed atom
pturb_index = 0        # index of perturbed atom
epsilon_error = 0.001 # measurement error

WyckoffSym = '1'     # to be improved in future work
MagVal = '0'         # to be improved in future work
```

```
16
17 db = sqlite3.connect('ITEOO_data.sqlite')
18
19 Pseudo_B, Functional, Polymorph, CalcType, AtomIndices = [], [], [], [], []
20
21 coord_dict = {}
22
23 for row in db.execute('''
24 select distinct ppfv.Energy,
25 ppfv.Coord1, ppfv.Coord2, ppfv.Coord3, ppfv.Coord4, ppfv.Coord5,
26 ppfv.Coord6, ppfv.Coord7, ppfv.Coord8, ppfv.Coord9, ppfv.Coord10,
27 ppfv.Coord11, ppfv.Coord12, ppfv.Coord13, ppfv.Coord14, ppfv.Coord15,
28 ppfv.Coord16, ppfv.Coord17, ppfv.Coord18, ppfv.Coord19, ppfv.Coord20,
29 ppfv.Coord21, ppfv.Coord22, ppfv.Coord23, ppfv.Coord24, ppfv.Coord25,
30 ppfv.Coord26, ppfv.Coord27, ppfv.Coord28, ppfv.Coord29, ppfv.Coord30,
31 ppfv.Coord31, ppfv.Coord32, ppfv.Coord33, ppfv.Coord34, ppfv.Coord35,
32 ppfv.Coord36, ppfv.Coord37, ppfv.Coord38, ppfv.Coord39, ppfv.Coord40,
33 ppfv.Coord41, ppfv.Coord42, ppfv.Coord43, ppfv.Coord44, ppfv.Coord45,
34 ppfv.Coord46, ppfv.Coord47, ppfv.Coord48, ppfv.Coord49, ppfv.Coord50,
35 ppfv.Coord51, ppfv.Coord52, ppfv.Coord53, ppfv.Coord54, ppfv.Coord55,
36 ppfv.Coord56, ppfv.Coord57, ppfv.Coord58, ppfv.Coord59, ppfv.Coord60,
37 ppfv.Coord61, ppfv.Coord62, ppfv.Coord63, ppfv.Coord64, ppfv.Coord65,
38 ppfv.Coord66, ppfv.Coord67, ppfv.Coord68, ppfv.Coord69, ppfv.Coord70,
39 ppfv.Coord71, ppfv.Coord72, ppfv.Coord73, ppfv.Coord74, ppfv.Coord75,
40 ppfv.Coord76, ppfv.Coord77, ppfv.Coord78, ppfv.Coord79, ppfv.Coord80,
41 ppfv.Coord81, ppfv.Coord82, ppfv.Coord83, ppfv.Coord84, ppfv.Coord85,
42 ppfv.Coord86, ppfv.Coord87, ppfv.Coord88, ppfv.Coord89, ppfv.Coord90,
43 ppfv.Coord91, ppfv.Coord92, ppfv.Coord93, ppfv.Coord94, ppfv.Coord95,
44 ppfv.Coord96 from Structure as s
45 inner join Composition1 as c1 on c1.SID=s.SID
46 inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
47 inner join CalcResultLRCalc as crlrc on crlrc.CID=mt.CID
48 inner join ParametersResponseVASP as prv on prv.Energy=crlrc.Energy
49 inner join ParametersPosFinalVASP as ppfv on ppfv.Energy=prv.Energy
50 where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
51 and c1.At1='Ti'
52 and (mt.PseudoB='PAW-Ti' or mt.PseudoB='PAW-Ti-pv' or mt.PseudoB='PAW-Ti-sv')
53 and (mt.Functional='PBE' or mt.Functional='PS')
54 and mt.Software='VASP'
55 and mt.Method='LR'
56 and crlrc.U3dIn='0'
```

```
and (prv.CalcType='Chi' or prv.CalcType='Chi0' or prv.CalcType='ChiS-S'
or prv.CalcType='Chi0S-S')
;'''):
    coord_list = []
    for i in range(1, len(row), 1):
        if row[i] != NULL_CHAR:
            coord_list.append( row[i] )

    coord_dict[ row[0] ] = coord_list

charge_dict = {}
CalcType, AtomIndices = [], []
for row in db.execute('''
select prv.Energy, prv.CalcType, prv.AtomIndex, prv.DChargeRHigh, prv.DChargeRLow, prv.DChargeRCenter,
prv.PChargeRHigh, prv.PChargeRLow, prv.PChargeRCenter from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join CalcResultLRCalc as crlrc on crlrc.CID=mt.CID
inner join ParametersResponseVASP as prv on prv.Energy=crlrc.Energy
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or
s.Morph='Brookite')
and c1.At1='Ti'
and (mt.PseudoB='PAW-Ti' or mt.PseudoB='PAW-Ti-pv' or mt.PseudoB='PAW-Ti-sv')
and (mt.Functional='PBE' or mt.Functional='PS')
and mt.Software='VASP'
and mt.Method='LR'
and crlrc.U3dIn='0'
and (prv.CalcType='Chi' or prv.CalcType='Chi0' or prv.CalcType='ChiS-S'
or prv.CalcType='Chi0S-S')
;'''):
    CalcType.append( row[1] )
    AtomIndices.append( row[2] )

    charge_list = []
    for i in range(1, len(row), 1):
        charge_list.append( row[i] )

    if row[0] not in charge_dict:
        charge_dict[ row[0] ] = {}

    try:
```

```
        charge_dict[ row[0] ][ row[1] ][ row[2] ] = charge_list
    except KeyError:
        charge_dict[ row[0] ][ row[1] ] = {}
        charge_dict[ row[0] ][ row[1] ][ row[2] ] = charge_list

Uturb_dict = {}
Pseudo, Functional, Polymorph = [], [], []
for row in db.execute('''
select crlrc.Energy, mt.PseudoB, mt.Functional, s.Morph, crlrc.U3dOut,
crlrc.Chi0N150, crlrc.Chi0N100, crlrc.Chi0N050, crlrc.Chi0P000, crlrc.Chi0P050,
crlrc.Chi0P100, crlrc.Chi0P150, crlrc.ChiN150, crlrc.ChiN100, crlrc.ChiN050, crlrc.ChiP000, crlrc.ChiP050, crlrc.ChiP100,
crlrc.ChiP150, crlrc.PturbMax, crlrc.PturbMin from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join CalcResultLRCalc as crlrc on crlrc.CID=mt.CID
inner join ParametersResponseVASP as prv on prv.Energy=crlrc.Energy
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite' or s.Morph='Brookite')
and c1.At1='Ti'
and (mt.PseudoB='PAW-Ti' or mt.PseudoB='PAW-Ti-pv' or mt.PseudoB='PAW-Ti-sv')
and (mt.Functional='PBE' or mt.Functional='PS')
and mt.Software='VASP'
and mt.Method='LR'
and crlrc.U3dIn='0'
and (prv.CalcType='Chi' or prv.CalcType='Chi0' or prv.CalcType='ChiS-S'
or prv.CalcType='Chi0S-S')
;'''):
    Pseudo.append( row[1] )
    Functional.append( row[2] )
    Polymorph.append( row[3] )

    Uturb_list = []
    for i in range(1, len(row), 1):
        Uturb_list.append( row[i] )

    Uturb_dict[ row[0] ] = Uturb_list

combine_dict = {}
for i in Uturb_dict.keys():
    combine_dict[i] = [ coord_dict[i], charge_dict[i], Uturb_dict[i] ]

Pseudo_list = list( set (Pseudo) )
```

```
Functional_list = list( set( Functional ) )
Morph_list = list( set( Polymorph ) )
CalcType_list = list( set( CalcType ) )
AtomIndices_list = list( set( AtomIndices ) )

SortElist = {}
EBsiteMatch = {}

for i in Pseudo_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Functional_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = {}

        for k in Morph_list:
            EBsiteMatch[i][j][k] = {}
            EBsiteMatch[i][j][k]['Chi'] = {}
            EBsiteMatch[i][j][k]['Chi0'] = {}
            del_list = []

            for l in combine_dict.keys():
                if combine_dict[l][2][0] == i and combine_dict[l][2][1] == j and combine_dict[l][2][2] == k:
                    SortElist[i][j][k] = combine_dict[l]
                    del_list.append( combine_dict[l] )

        for k in del_list:
            del k

for i in Pseudo_list:
    for j in Functional_list:
        for k in Morph_list:
            try:
                coord_list = []

                for l,m in enumerate( SortElist[i][j][k][0] ):
                    if SortElist[i][j][k][0][l] != NULL_CHAR:
                        coord_value = str(m).split(SColon_CHAR)
                        coord_list.append( [ float( coord_value[0] ), float(
                         coord_value[1] ), float( coord_value[2] ) ] )
```

```
                EBsiteMatch[i][j][k]['coord'] = coord_list
                EBsiteMatch[i][j][k]['U_diag'] = SortElist[i][j][k][2][3]

                EBsiteMatch[i][j][k]['Chi0_Pturb'] = [ SortElist[i][j][k][2][4],
                                                       SortElist[i][j][k][2][5],
                                                       SortElist[i][j][k][2][6],
                                                       SortElist[i][j][k][2][7],
                                                       SortElist[i][j][k][2][8],
                                                       SortElist[i][j][k][2][9],
                                                       SortElist[i][j][k][2][10]]


                EBsiteMatch[i][j][k]['Chi_Pturb'] = [ SortElist[i][j][k][2][11],
                                                      SortElist[i][j][k][2][12],
                                                      SortElist[i][j][k][2][13],
                                                      SortElist[i][j][k][2][14],
                                                      SortElist[i][j][k][2][15],
                                                      SortElist[i][j][k][2][16],
                                                      SortElist[i][j][k][2][17] ]

                EBsiteMatch[i][j][k]['PturbMax'] = SortElist[i][j][k][2][18]
                EBsiteMatch[i][j][k]['PturbMin'] = SortElist[i][j][k][2][19]

                Pturb_divisor = float( len( EBsiteMatch[i][j][k]['Chi0_Pturb'] )
                                - 1 )
                Pturbcept_index = int( Pturb_divisor / 2. )
                EBsiteMatch[i][j][k]['List_Pturb'] = np.arange(
                 EBsiteMatch[i][j][k]['PturbMin'],
                 EBsiteMatch[i][j][k]['PturbMax'] + 0.0001,
                 ( EBsiteMatch[i][j][k]['PturbMax'] -
                   EBsiteMatch[i][j][k]['PturbMin'] ) /  Pturb_divisor )

                EBsiteMatch[i][j][k]['Intercept'] = abs(
                EBsiteMatch[i][j][k]['Chi0_Pturb'][Pturbcept_index] -
                 EBsiteMatch[i][j][k]['Chi_Pturb'][Pturbcept_index] )

                slope, intercept, r_value, p_value, std_err = stats.linregress(
                 EBsiteMatch[i][j][k]['List_Pturb'],
                 EBsiteMatch[i][j][k]['Chi0_Pturb'] )
                EBsiteMatch[i][j][k]['Chi0_R2'] = r_value**2
```

```
                slope, intercept, r_value, p_value, std_err = stats.linregress(
                 EBsiteMatch[i][j][k]['List_Pturb'],
                 EBsiteMatch[i][j][k]['Chi_Pturb'] )
                EBsiteMatch[i][j][k]['Chi_R2'] = r_value**2

                for l in range( len(EBsiteMatch[i][j][k]['coord']) ):
                    try:
                        if SortElist[i][j][k][1]['ChiS-S'][l+1][2] == 0.0:
                            EBsiteMatch[i][j][k]['Chi'][l+1] = (
                             float(SortElist[i][j][k][1]['ChiS-S'][l+1][5]) -
                             float(SortElist[i][j][k][1]['ChiS-S'][l+1][6]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )

                            EBsiteMatch[i][j][k]['ChiO'][l+1] = (
                             float(SortElist[i][j][k][1]['ChiOS-S'][l+1][5]) -
                             float(SortElist[i][j][k][1]['ChiOS-S'][l+1][6]) ) / (
                              float(EBsiteMatch[i][j][k]['PturbMax'])
                              - float(EBsiteMatch[i][j][k]['PturbMin']) )

                        else:
                            EBsiteMatch[i][j][k]['Chi'][l+1] = (
                             float(SortElist[i][j][k][1]['ChiS-S'][l+1][2]) -
                             float(SortElist[i][j][k][1]['ChiS-S'][l+1][3]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )

                            EBsiteMatch[i][j][k]['ChiO'][l+1] = (
                             float(SortElist[i][j][k][1]['ChiOS-S'][l+1][2]) -
                             float(SortElist[i][j][k][1]['ChiOS-S'][l+1][3]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )

                    except KeyError:
                        if SortElist[i][j][k][1]['Chi'][l+1][2] == 0.0:
                            EBsiteMatch[i][j][k]['Chi'][l+1] = (
                             float(SortElist[i][j][k][1]['Chi'][l+1][5]) -
                             float(SortElist[i][j][k][1]['Chi'][l+1][6]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )
```

```
                            EBsiteMatch[i][j][k]['Chi0'][l+1] = (
                             float(SortElist[i][j][k][1]['Chi0'][l+1][5]) -
                             float(SortElist[i][j][k][1]['Chi0'][l+1][6]) ) / (
                              float(EBsiteMatch[i][j][k]['PturbMax'])
                              - float(EBsiteMatch[i][j][k]['PturbMin']) )

                        else:
                            EBsiteMatch[i][j][k]['Chi'][l+1] = (
                             float(SortElist[i][j][k][1]['Chi'][l+1][2]) -
                             float(SortElist[i][j][k][1]['Chi'][l+1][3]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )

                            EBsiteMatch[i][j][k]['Chi0'][l+1] = (
                             float(SortElist[i][j][k][1]['Chi0'][l+1][2]) -
                             float(SortElist[i][j][k][1]['Chi0'][l+1][3]) ) / (
                             float(EBsiteMatch[i][j][k]['PturbMax'])
                             - float(EBsiteMatch[i][j][k]['PturbMin']) )

            except KeyError:
                pass

for i in Pseudo_list:
    for j in Functional_list:
        for k in Morph_list:
            try:
                sigma_error_1 = ( ( epsilon_error**(2) +
                    epsilon_error**(2) )**(0.5)
                  / ( EBsiteMatch[i][j][k]['PturbMax'] -
                    EBsiteMatch[i][j][k]['PturbMin'] ) )

                sigma_error_2_chi  = ( EBsiteMatch[i][j][k]['Chi'][1] )**(-2.0)
                sigma_error_2_chi0 = ( EBsiteMatch[i][j][k]['Chi0'][1] )**(-2.0)

                EBsiteMatch[i][j][k]['Sigma'] = (
                 (sigma_error_1 * sigma_error_2_chi)**(2.0) +
                 (sigma_error_1 * sigma_error_2_chi0)**(2.0) )**(0.5)
            except KeyError:
                pass
```

```
303
304 for i in Pseudo_list:
305     for j in Functional_list:
306         for k in Morph_list:
307             try:
308                 respmat_dim = len( EBsiteMatch[i][j][k]['coord'] )
309                 respmat_chi0 = np.zeros( (respmat_dim, respmat_dim) )
310                 respmat_chi = np.zeros( (respmat_dim, respmat_dim) )
311
312                 for l in range( respmat_dim ):
313                     respmat_chi0[0][l] = EBsiteMatch[i][j][k]['Chi0'][l+1]
314                     respmat_chi[0][l] = EBsiteMatch[i][j][k]['Chi'][l+1]
315
316                 for l in range( respmat_dim ):
317                     respmat_chi0[l][0] = respmat_chi0[0][l]
318                     respmat_chi[l][0] = respmat_chi[0][l]
319
320                 respmat_intxn = {}
321                 for l in range( 0, respmat_dim, 1 ):
322                     respmat_intxn[l] = {}
323
324                 for l in range( 0, respmat_dim, 1 ):
325                     respmat_intxn[l][l] = [ abs( np.subtract(
326                      EBsiteMatch[i][j][k]['coord'][l],
327                      EBsiteMatch[i][j][k]['coord'][l]) ),
328                      [ WyckoffSym, WyckoffSym ], [ MagVal, MagVal ] ]
329
330                     for m in range( 1, respmat_dim, 1 ):
331                         respmat_intxn[l][m] = [ abs( np.subtract(
332                          EBsiteMatch[i][j][k]['coord'][l],
333                      EBsiteMatch[i][j][k]['coord'][m]) ),
334                      [ WyckoffSym, WyckoffSym ], [ MagVal, MagVal ] ]
335                         respmat_intxn[m][l] = respmat_intxn[l][m]
336
337                 for l in range( 1, respmat_dim, 1 ):
338                     for m in range( 2, respmat_dim, 1 ):
339                         for n in range( 0, respmat_dim, 1 ):
340
341                             counter_sym = 0
342                             for o, p in zip( range( len(respmat_intxn[l][m][1])
343                              ), range( len(respmat_intxn[0][n][1]) ) ):
```

```
                                if ( respmat_intxn[l][m][1][o] !=
                                 respmat_intxn[l][m][1][p] ):
                                    counter_sym += 1
                                if ( respmat_intxn[l][m][2][o] !=
                                 respmat_intxn[l][m][2][p] ):
                                    counter_sym += 1

                        if counter_sym == 0:
                            respmat_chi0[l][m] = respmat_chi0[0][n]
                            respmat_chi[l][m] = respmat_chi[0][n]

                for l in range( 0, respmat_dim, 1 ):
                    for m in range( 1, respmat_dim, 1 ):
                        respmat_chi0[m][l] = respmat_chi0[l][m]
                        respmat_chi[m][l] = respmat_chi[l][m]

                for l in range(1, respmat_dim, 1 ):
                    if l < int( len( EBsiteMatch[i][j][k]['coord'] ) / 3. ):
                        respmat_chi0[l][l] = respmat_chi0[0][0]
                        respmat_chi[l][l] = respmat_chi[0][0]
                    else:
                        respmat_chi0[l][l] = comp_back
                        respmat_chi[l][l] = comp_back

                np.savetxt('./figures/respmat_chi0_' + str(i) + '_' + str(j)
                           + '_' + str(k) + '.txt', respmat_chi0)
                np.savetxt('./figures/respmat_chi_' + str(i) + '_' + str(j)
                           + '_' + str(k) + '.txt', respmat_chi)
                try:
                    invmat_chi0 = np.linalg.inv(respmat_chi0)
                    invmat_chi  = np.linalg.inv(respmat_chi)
                except (np.linalg.linalg.LinAlgError):
                    print "Error for system: " + str(i) + "," + str(j) + "," + str(k)
                    print respmat_chi0
                    print respmat_chi
                    import sys; sys.exit()
                EBsiteMatch[i][j][k]['U_val'] = (invmat_chi[ pturb_index + 1 ][ pturb_index + 1 ]
                                                 - invmat_chi0[ pturb_index + 1 ][ pturb_index + 1 ])

            except KeyError:
                pass
```

```
print "PBEsol Functional, Ti pseudopotential, Rutile: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PS']['Rutile']['Sigma']) + "\n"

print "PBEsol Functional, Ti pseudopotential, Anatase: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PS']['Anatase']['Sigma']) + "\n"

print "PBE Functional, Ti-pv pseudopotential, Rutile: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Rutile']['Sigma']) + "\n"

print "PBE Functional, Ti-pv pseudopotential, Anatase: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti-pv']['PBE']['Anatase']['Sigma']) + "\n"

print "PBE Functional, Ti-sv pseudopotential, Rutile: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
```

```
print str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Rutile']['Sigma']) + "\n"

print "PBE Functional, Ti-sv pseudopotential, Anatase: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti-sv']['PBE']['Anatase']['Sigma']) + "\n"

print "PBE Functional, Ti pseudopotential, Rutile: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['Sigma']) + "\n"

print "PBE Functional, Ti pseudopotential, Anatase: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PBE']['Anatase']['Sigma']) + "\n"

print "PBE Functional, Ti pseudopotential, Columbite: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Columbite']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Columbite']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PBE']['Columbite']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Columbite']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Columbite']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PBE']['Rutile']['Sigma']) + "\n"
```

```
print "PBE Functional, Ti pseudopotential, Brookite: "
print r'Coefficient of correlation (R^2) of Chi = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['Chi_R2'])
print r'Coefficient of correlation (R^2) of Chi0 = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['Chi0_R2'])
print r'Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) = '
print str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['Intercept'])
print r'U(diag) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['U_diag'])
print r'U(3d) = ' + str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['U_val'])
print ' +/- ' + str(EBsiteMatch['PAW-Ti']['PBE']['Brookite']['Sigma']) + "\n"
```

```
PBEsol Functional, Ti pseudopotential, Rutile:
Coefficient of correlation (R^2) of Chi = 1.0
Coefficient of correlation (R^2) of Chi0 = 0.999894597045
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 2.709923664
U(3d) = 2.72727272727
 +/- 0.120340278018

PBEsol Functional, Ti pseudopotential, Anatase:
Coefficient of correlation (R^2) of Chi = 0.999853315079
Coefficient of correlation (R^2) of Chi0 = 0.999891805336
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.006
U(diag) = 2.610340479
U(3d) = 2.55770542748
 +/- 0.116749834126

PBE Functional, Ti-pv pseudopotential, Rutile:
Coefficient of correlation (R^2) of Chi = 0.999446290144
Coefficient of correlation (R^2) of Chi0 = 0.999847435389
```

```
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 4.772727273
U(3d) = 4.77272727273
 +/- 0.2674732215

PBE Functional, Ti-pv pseudopotential, Anatase:
Coefficient of correlation (R^2) of Chi = 1.0
Coefficient of correlation (R^2) of Chi0 = 0.999842554397
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 4.365079365
U(3d) = 4.2950791794
 +/- 0.2432474491

PBE Functional, Ti-sv pseudopotential, Rutile:
Coefficient of correlation (R^2) of Chi = 0.999342969777
Coefficient of correlation (R^2) of Chi0 = 0.99985287976
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 6.029684601
U(3d) = 6.02968460111
 +/- 0.392087064657

PBE Functional, Ti-sv pseudopotential, Anatase:
Coefficient of correlation (R^2) of Chi = 0.999519538757
Coefficient of correlation (R^2) of Chi0 = 0.999926909133
```

```
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 5.730745907
U(3d) = 5.32069676882
 +/- 0.369770048421

PBE Functional, Ti pseudopotential, Rutile:
Coefficient of correlation (R^2) of Chi = 0.999825215567
Coefficient of correlation (R^2) of Chi0 = 0.999795188219
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 3.101503759
U(3d) = 3.1015037594
 +/- 0.137397900257

PBE Functional, Ti pseudopotential, Anatase:
Coefficient of correlation (R^2) of Chi = 0.999777282851
Coefficient of correlation (R^2) of Chi0 = 0.999877351204
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 3.007518797
U(3d) = 2.92857722597
 +/- 0.132767369204

PBE Functional, Ti pseudopotential, Columbite:
Coefficient of correlation (R^2) of Chi = 0.999835418038
Coefficient of correlation (R^2) of Chi0 = 0.999806562194
```

```
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 2.982632771
U(3d) = 2.98263277121
 +/- 0.137397900257

PBE Functional, Ti pseudopotential, Brookite:
Coefficient of correlation (R^2) of Chi = 0.999792748978
Coefficient of correlation (R^2) of Chi0 = 0.999819430315
Intercept (|Chi0(alpha = 0 eV) - Chi(alpha = 0 eV)|) =
0.0
U(diag) = 2.933607823
U(3d) = 2.93360782295
 +/- 0.128313152932
```

As shown in the output above and mentioned in the article, the linear response resolved values of U for systems with p-valence and s-valence inclusive Ti pseudopotentials illustrate the effects of modifying valence electron character on U value magnitude, which has been demonstrated for charged oxide cation systems previously.[S44] The changes in first-principles resolved U values and corresponding energetic ranges are consistent in the cases of Ti_sv and, to a lesser extent, Ti_pv Rutile-based calculations. When compared with past work,[S46] this consistency is not present in the Ti_pv and Ti_sv Anatase calculations. A literature assessment of several densities of state (DOS) and projected densities of state (PDOS) plots comparing the DFT and DFT+U electronic structures of Rutile and Anatase calculations, which apply the PBE functional and all variations of Ti pseudopotential, reveals a possible explanation for this lack of consistency. In the case of Rutile, the incrementation of U starting from $U = 0$ eV and increasing to $U = 3$ eV (with standard pseudopotentials)[S47] or $U =$ 4-6 eV (with p-valence or s-valence inclusive pseudopotentials)[S46] uniformly reveals

a splitting of the Ti $3d$ band at higher values of U, regardless of the pseudopotentials selected for the Ti cation. In contrast, the behavior of Anatase with U incrementation appears to be dependent on the electron configuration of the Ti $3d$ cation, which is similarly affected by pseudopotential selection. In the case of a $3d^2$ configuration of the Ti cation in Anatase, increasing the value of U from 0 eV to 3 eV does not appear to produce characteristic $3d$ band splitting,[S48] while a $3d^1$ electron configuration of the same cation reveals splitting upon incrementation of its associated U value.[S49] This difference in $3d$ band splitting represents the larger relative difference of the DFT and DFT + U electronic ground states in p-valence and s-valence inclusive TiO_2 Anatase calculations, inferring that application of the self-consistent Hubbard U approach would more significantly change the value of U resolved via first-principles for these calculations than calculations featuring standard pseudopotentials. Generally, consideration of the self-consistent linear response approach over the corresponding standard approach improves the value of the resolved U parameter, thus applying self-consistent linear response theory could increase the calculated U values of pertinent p-valence and s-valence Ti pseudopotential inclusive calculations to the extent that these U values predict an experimentally consistent formation energy ordering, namely predicting that Rutile is energetically more stable than Anatase.[S38]

Considering that standard linear response calculations are initialized using electronic structure information provided from GGA rather than GGA+U ground states, changes in the electronic and crystal structure of the material induced by addition of the U parameter are handled approximately. In general, GGA-based functionals incorrectly represent energetic ground states of strongly correlated systems by significantly overestimating electron-electron interaction in pertinent orbitals (frequently d or f orbitals), leading to inaccuracies in linear response calculations that supply an initial guess for U ($U_{in} = 0$ eV) solely formed using the GGA functional. However, the addition of a non-zero U_{in} to linear response calculations, while largely improving the accuracy of the ground state representation of a strongly correlated system, can also underestimate or overestimate the amount of on-site electron-electron

interaction necessary to produce a physical electronic structure. Given that the initial and final response matrices are constructed from the second derivatives of the U corrected total energy (E_U) with respect to n, the relationship between them should be the product of a constant and the effective change in orbital occupancy that occurs during the perturbuation (Δn). When the electronic ground state is accurately, physically, and consistently represented over U_{in} and U_{out} (namely at a sufficiently high non-zero U_{in}), linear response calculations yield linear changes scaled by this orbital occupancy constant and thus a linear relationship between U_{in} and U_{out}.[S38,S39] Therefore, the correct GGA+U ground state of a system can be represented with the correct amount of on-site electron-electron interaction at $U_{in} = 0$ eV when this linear relationship between U_{in} and U_{out} is extrapolated to include $U_{in} = 0$ eV, yielding the self-consistent amount of on-site interaction U_{scf} through Equation 7:[S38]

$$U_{out} = U_{scf} - \frac{U_{in}}{\Delta n} \tag{7}$$

For a set of inputted values of U (U_{in} = 0, 1.0, 2.0, 2.5, 3.0, 3.5, and 4.0 eV), an attempt to calculate the relationship above for PM TiO_2 Rutile using the PBE functional and standard Ti and O pseudopotentials is performed. This is done by applying the values of U_{in} (independent variable) and U_{out} (dependent variable) to a linear fit and then extrapolating, as is shown via the query and plot below:

$$\Pi_{U3dIn,U3dOut}$$

$$\sigma[(Morph =' Rutile')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' LR')]$$

$$\sigma \wedge [(U3dIn =' 0') \vee (U3dIn =' 1') \vee (U3dIn =' 2') \vee (U3dIn =' 2.5')$$

$$\sigma \vee (U3dIn =' 3') \vee (U3dIn =' 3.5') \vee (U3dIn =' 4')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie CalcResultLRCalc]$$

```
#+caption: MySQL query Self-Consistent Linear Response Theory
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

U3dIn, U3dOut = [], []
NULL_CHAR = '-'

for row in db.execute('''
select distinct crlrc.U3dIn, crlrc.U3dOut from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join CalcResultLRCalc as crlrc on crlrc.CID=mt.CID
where s.Morph='Rutile'
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.Functional='PBE'
and mt.Software='VASP'
and mt.Method='LR'
```

```
and (crlrc.U3dIn='0' or crlrc.U3dIn='1' or crlrc.U3dIn='2' or crlrc.U3dIn='2.5'
or crlrc.U3dIn='3' or crlrc.U3dIn='3.5' or crlrc.U3dIn='4')
;'''):
    a,b = row

    if b != NULL_CHAR:
        U3dIn.append( a )
        U3dOut.append( b )

ax = plt.gca()

print "PBE Functional, Ti O pseudopotentials, Rutile: \n"
print "U (input): " + str(U3dIn) + "\n"
print "U (output): " + str(U3dOut[0:3])
print str(U3dOut[3:]) + "\n"

plt.figure(figsize=(3,4))

plt.plot(U3dIn, U3dOut, 'ro-', label = r'U$_{out}$, Rutile')
plt.xticks( [0, 1, 2, 3, 4] )
plt.xlabel( 'Input U (eV)' )
plt.ylim( (3.0, 4.0) )
plt.ylabel( 'Output U (eV)' )
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.20)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-LRSC-R' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, Ti O pseudopotentials, Rutile:

U (input): [0.0, 1.0, 2.0, 2.5, 3.0, 3.5, 4.0]

U (output): [3.1015037589999999, 3.198869476, 3.3163265310000001]
[3.6687631029999999, 3.6324786320000002, 3.7304075239999999, 3.9162112929999999]
```

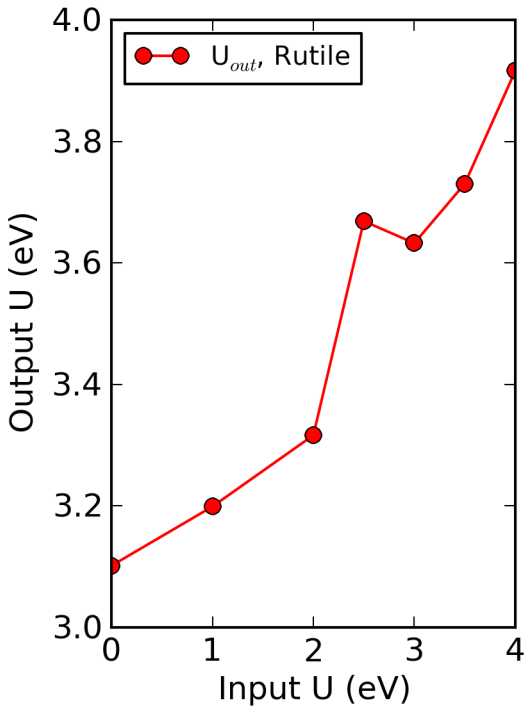

Figure S9

The query and plot above illustrates an attempt to achieve a self-consistent value of U for PBE TiO_2 Rutile with standard pseudopotentials. No direct method for importing electronic structures from the "convergence" step with U greater than 0 eV is currently implemented in this study, as LDAUU and LDAUJ no longer dictate the magnitude of the U parameter imparted to the calculation when LDAUTYPE = 3 in the "perturbation" steps. The four-step procedure detailed previously was replicated for each U value tested. Considering that U is greater than 0 in the "relaxation" and "convergence" steps of these calculations, the converged structures were distinct from the $U = 0$ eV case. The initial response (chi_0) perturbation steps received different CHGCAR for calculations featuring different values of U, which apparently and distinctly changed the slopes of calculated initial responses of perturbed atoms with respect to the $U = 0$ case. However, the final response (chi) "perturbation" steps received different WAVECAR at different values of U, in an attempt to influence

these response slopes and mimic the effect of placing a U value greater than 0 eV on these calculations. However, no effect was observed when this was attempted. As a result, the U_{out} values displayed above increase with increasing U_{in} rather than decrease monotonically, as is seen in past research.[S38] This likely results from the inability of chi perturbation contributions with no imparted GGA+U electronic structure to compensate for the increasing values of the chi_0 perturbation contributions to U, which has some GGA+U electronic structure character imparted to it via the information provided by pertinent CHGCAR files. A more extensive investigation of how to import electronic states, as shown in U-ramping method,[S50] or modification of the linear response calculation method in VASP[S21,S40] is needed to improve this result further.

4.2 Hybrid Functionals

In this study, hybrid functional calculations are completed primarily for the Rutile and Anatase TiO_2 polymorphs, applying the PBE functional for non-exact exchange, the PBE0[S11] and HSE06[S51,S52] methodologies for mixing PBE and exact HF exchange, and the standard Ti and O pseudopotentials to complete calculations. In a multi-step procedure similar to that completed in Hubbard U incrementation calculations, structural relaxation was accomplished for hybrid functional calculations. As stated previously for Hubbard U incrementation calculations, starting polymorph structure[S19] DFT total energies (E_0) and equilibrium cell volumes (V_0) were resolved by first performing several fixed cell volume, variable cell shape and atomic coordinate structural relaxation calculations encompassing values of V_0 on calculations featuring only PBE exchange. After a suitable range of PBE based calculations that can span both V_0 for the PBE case and the presumed corresponding minimum of a tested fraction of exact exchange (e.g.: $a = 0.25$) are completed, the CHGCAR and WAVECAR files produced in the output of PBE calculations at particular volumes can be applied as inputs to hybrid functional calculations of the same volume. These hybrid functional calculation results, which were achieved via input from previous PBE calculations, can then

be applied to achieve results that are detailed in this article using the Birch-Murnaghan EOS fitting approach[S10] detailed in Section 4.1. Modifications of this technique can be – and has been – performed to accommodate particular calculations. One of these modifications entails only importing the WAVECAR file from a PBE calculation into a hybrid functional calculation (ignoring the CHGCAR). Another modification entails incorporating an intermediate step into the procedure outlined above, in which CHGCAR and/or WAVECAR information is imported into a hybrid functional calculation at a lower ENCUT, after which the CHGCAR and WAVECAR information resulting from the lower ENCUT calculation is imported into a calculation at an ENCUT more suitable for description in the article. This type of modification was also completed for calculations completed with uniformly reduced k-point samplings afforded by the NKRED command. Yet another modification, which improves calculation speed for calculations involving higher fractions of exact exchange, similarly imports CHGCAR and WAVECAR information from a hybrid calculation with a relatively low exact exchange fraction (e.g.: $a = 0.50$) into a matching calculation of equal volume with a higher fraction (e.g.: $a = 0.75$). This approach resembles the U-ramping method detailed in previous work,[S50] though this approach is performed for hybrid functional calculations rather than Hubbard U incrementation calculations.

4.2.1 Sample Input Files: Hybrid Functionals

A single example set of input files characterizing the predominate implementation of the procedure detailed above is described as follows. The first step of the procedure developed in Section 4.1.1, namely that involving the iterative use of fixed cell shape, variable cell volume and atomic coordinate (ISIF = 4 in VASP) calculations, is performed for all PBE calculations of interest in hybrid functional calculations. Subsequently, the CONTCAR and WAVECAR produced from that calculation is applied to a hybrid functional structural relaxation calculation (also ISIF = 4) with a particular value of ENCUT and NKRED. In the event that electronic convergence was difficult to reach for a particular system, hybrid

functional calculations employing lower values of ENCUT (e.g.: 400 eV) and/or higher values of NKRED (e.g.: 3, rather than 2) initially received the WAVECAR and CONTCAR inputs from PBE calculations. Subsequently, the output of those low ENCUT and/or high NKRED calculations served as inputs for calculations used within this article, the input files of which are reproduced below. Note that, in cases where initial calculations with lower ENCUT or higher NKRED values were used as intermediates for more expensive calculations, system volume was always conserved over PBE, initial hybrid, and implemented hybrid calculations. Also, note that the INCAR and KPOINTS input file information used in these intermediate steps is, excluding differences in ENCUT and NKRED information, equivalent to that of analogous implemented hybrid functional calculations. The example below corresponds to an HSE06 calculation involving TiO_2 Rutile with 25% exact HF exchange, 75% PBE exchange at a fixed cell volume of 62.35 A^3.

Listing 20: INCAR Rutile HSE06 (PBE-HF) PAW_standard VASP Relaxation Step

```
ISTART = 1 ; ICHARG = 0
ENCUT = 550
ISMEAR = 0 ; SIGMA = 0.05

ISYM = 1; SYMPREC = 1E-06

IBRION = 1
EDIFF = 5E-05; EDIFFG = -0.02

MAXMIX = -75
NELMIN = 5
NELM = 250
NSW = 100

ISPIN = 2
ISIF = 4

LHFCALC = .TRUE.
HFSCREEN = 0.2
ALGO = Damped
TIME = 0.4

PRECFOCK = Normal
LMAXFOCK = 4
LASPH = .TRUE.

AEXX = 0.25

NKRED = 2

NBANDS = 25
```

Listing 21: KPOINTS Rutile HSE06 (PBE-HF) PAW_standard VASP Relaxation Step

```
6x6x6
0
Gamma
6 6 6
0 0 0
```

Listing 22: POSCAR Rutile HSE06 (PBE-HF) PAW_standard VASP Relaxation Step

```
Ti O
   4.61000000000000
     0.9980395576792923    0.0000037613141482    0.0000000000000000
     0.0000037613141482    0.9980395576792923    0.0000000000000000
     0.0000000000000000    0.0000000000000000    0.6388758158322786
   Ti   O
     2     4
Direct
  0.0000000000000000  0.0000000000000000  0.0000000000000000
  0.5000000000000000  0.5000000000000000  0.5000000000000000
  0.3047876991736018  0.3047876991736018  0.0000000000000000
  0.6952123008263982  0.6952123008263982  0.0000000000000000
  0.8047916748910851  0.1952083251089149  0.5000000000000000
  0.1952083251089149  0.8047916748910851  0.5000000000000000
```

Results from the calculations of the following form presented above, namely WAVECAR and CONTCAR files, can be integrated into calculations employing different fractions of exact exchange (AEXX). This can be performed to complete calculations at a higher fraction of exact exchange with reduced computational expense. All calculations completed using this technique used input parameters identical to those presented above, except for appropriately substituted values of AEXX.

4.2.2 Plot Generation: Hybrid Functionals

For all fractions of exact exchange evaluated for Rutile-Anatase hybrid functional formation energetics (a = 0.250, 0.500, 0.750, 0.825, 0.875, 0.950, 1.000) and corresponding Rutile-Columbite energetics (a = 0.250), a plot is developed below and is presented with its corresponding queries:

$\Pi_{Morph,Functional,FractionHF,EOpt,Stoich1}$

$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase') \vee (Morph =' Columbite')]$

$\sigma \wedge [(At1 =' Ti')]$

$\sigma \wedge [(PseudoB =' PAW - Ti')]$

$\sigma \wedge [(PseudoO =' PAW - O')]$

$\sigma \wedge [(Software =' VASP')]$

$\sigma \wedge [(Method =' E')]$

$\sigma \wedge [(NKRED =' 2')]$

$\sigma \wedge [(Functional =' HSE06') \vee (Functional =' PBE0')]$

$\sigma \wedge [(FractionHF =' 0.25') \vee (FractionHF =' 0.5') \vee (FractionHF =' 0.75')$

$\sigma \vee (FractionHF =' 0.825') \vee (FractionHF =' 0.875') \vee (FractionHF =' 0.95')$

$\sigma \vee (FractionHF =' 1')]$

$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEHFRange$

$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP]$

$$\Pi_{Morph,UValue,EOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(UValue =' 0')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query PBE0 vs. HSE06 Rutile Anatase
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITE00_data.sqlite')

Polymorph, Functional, FractionHF, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct s.Morph, mt.Functional, feh.FractionHF, feh.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where (s.Morph='Rutile' or s.Morph='Anatase' or s.Morph='Columbite')
and c1.At1='Ti'
```

```
21  and mt.PseudoB='PAW-Ti'
22  and mt.PseudoO='PAW-O'
23  and mt.Software='VASP'
24  and mt.Method='E'
25  and input.NKRED='2'
26  and (mt.Functional='HSE06' or mt.Functional='PBE0')
27  and (feh.FractionHF='0.25' or feh.FractionHF='0.5' or feh.FractionHF='0.75'
28  or feh.FractionHF='0.825' or feh.FractionHF='0.875'
29  or feh.FractionHF='0.95' or feh.FractionHF='1')
30  ;'''):
31      datapts_list = []
32      a,b,c,d,e = row
33      datapts_list.append( (a,b,c,d,e) )
34
35      Polymorph += [a]
36      Functional += [b]
37
38      datapts_dict[row] = datapts_list
39
40  Morph_list = list( set( Polymorph ) )
41  Functional_list = list( set( Functional ) )
42  SortElist = {}
43  EBsiteMatch = {}
44
45  for i in Morph_list:
46      SortElist[i] = {}
47      EBsiteMatch[i] = {}
48
49      for j in Functional_list:
50          SortElist[i][j] = {}
51          EBsiteMatch[i][j] = []
52          del_list = []
53
54          for k in datapts_dict:
55              if k[0] == i and k[1] == j:
56                  SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
57                  del_list.append( datapts_dict[k] )
58              else:
59                  pass
60
61          for l in del_list:
```

```
            del l
        if not SortElist[i][j]:
            del SortElist[i][j]

Polymorph, Uvalue, Eopt, atomnum = [], [], [], []
datapts_dict = {}

for row in db.execute('''
select distinct s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on mt.CID=feu.CID
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and mt.Functional='PBE'
and feu.UValue='0'
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Polymorph += [a]

    datapts_dict[row] = datapts_list

Morph_list_0 = list( set( Polymorph ) )
for i in Morph_list_0:
    SortElist[i]['PBE0'][float(0.)] = []
    SortElist[i]['HSE06'][float(0.)] = []
    del_list = []

    for j in datapts_dict:
        if j[0] == i:
            SortElist[ j[0] ]['PBE0'][float(0.)] = float( j[2] ) / float( j[3] )
            SortElist[ j[0] ]['HSE06'][float(0.)] = float( j[2] ) / float( j[3] )
            del_list.append( datapts_dict[j] )
        else:
```

```
            pass

for i in Morph_list:
    for j in Functional_list:
        try:
            HF_list = SortElist[i][j].keys()
            HF_list.sort()
            EmapHF_list = []

            for k in HF_list:
                EmapHF_value = SortElist[i][j][k] - SortElist[WRT_Morph][j][k]
                EmapHF_list.append( EmapHF_value )

            EBsiteMatch[i][j].append( EmapHF_list )
            EBsiteMatch[i][j].append( HF_list )
        except KeyError:
            pass

ax = plt.gca()

print "PBE0 Functional:"
print "Rutile: " + str(EBsiteMatch['Rutile']['PBE0'][0][0:2])
print str(EBsiteMatch['Rutile']['PBE0'][0][2:5])
print str(EBsiteMatch['Rutile']['PBE0'][0][5:])
print "Anatase: " + str(EBsiteMatch['Anatase']['PBE0'][0][0:2])
print str(EBsiteMatch['Anatase']['PBE0'][0][2:5])
print str(EBsiteMatch['Anatase']['PBE0'][0][5:]) + "\n"

print "HSE06 Functional:"
print "Rutile: " + str(EBsiteMatch['Rutile']['HSE06'][0][0:2])
print str(EBsiteMatch['Rutile']['HSE06'][0][2:5])
print str(EBsiteMatch['Rutile']['HSE06'][0][5:])
print "Anatase: " + str(EBsiteMatch['Anatase']['HSE06'][0][0:2])
print str(EBsiteMatch['Anatase']['HSE06'][0][2:5])
print str(EBsiteMatch['Anatase']['HSE06'][0][5:])
print "Columbite (a = " + str(EBsiteMatch['Columbite']['HSE06'][1]) + "): "
print str(EBsiteMatch['Columbite']['HSE06'][0]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Rutile']['PBE0'][1], EBsiteMatch['Rutile']['HSE06'][0], 'k-')

```

```
plt.plot(EBsiteMatch['Anatase']['PBE0'][1], EBsiteMatch['Anatase']['PBE0'][0],
'ro-', label = r'$\Delta$E$_{R-A,PBE0}$')
plt.plot(EBsiteMatch['Anatase']['HSE06'][1], EBsiteMatch['Anatase']['HSE06'][0],
'bo-', label = r'$\Delta$E$_{R-A,HSE06}$')
plt.plot(EBsiteMatch['Columbite']['HSE06'][1],
EBsiteMatch['Columbite']['HSE06'][0], 'go-', label = r'$\Delta$E$_{R-C,HSE06}$')

plt.xlabel( 'Exact Exchange Fraction (%)' )
plt.ylim( (-0.1, 0.05) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'lower right', prop={'size':9.5})

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-stability-RAC-HSE06PBE0' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE0 Functional:

Rutile: [0.0, 0.0]

[0.0, 0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [-0.081062179999999984, -0.063504779999995264]

[-0.033888025000003097, 0.004948500000000046853, 0.018051349999993249]

[0.026492750000002729, 0.040669000000001176, -0.039267699999996353]


HSE06 Functional:

Rutile: [0.0, 0.0]

[0.0, 0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [-0.081062179999999984, -0.065773704999998017]

[-0.039652625000002217, -0.005535270000000028676, 0.006335014999999976679]

[0.01398415999999969, 0.026451299999997957, -0.050894339999999261]
```

```
Columbite (a = [0.25]):
[0.015586054999999988]
```

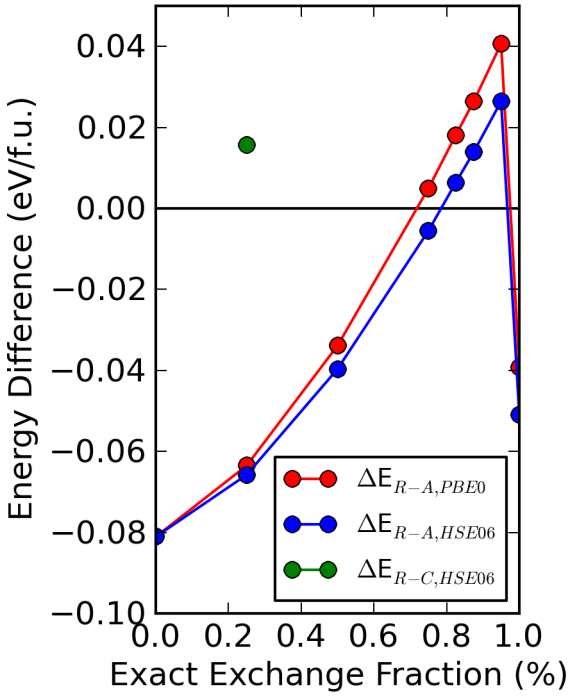

Figure S10

As was shown in Section 4.1.4, corresponding PBE functional results for the Rutile, Anatase, Columbite, and Brookite TiO_2 polymorphs were all equal to 3.0 eV within uncertainty and generally closely encompassed this value. Consistent with these results, those shown in the PBE, standard PAW pseudopotential subsection of Section 4.1.2, and those resolved in previous work,[S33,S53] a Rutile-Anatase formation energy of approximately +0.005 eV/f.u. TiO_2 can be predictively estimated. This result is in very strong agreement with that derived by Rao et al.[S46,S54] When this Rutile-Anatase formation energy is transferred from predictive Hubbard U involved results to non-predictive hybrid functional calculations, the PBE0 and HSE06 hybrid functionals appear to achieve this relative energetic value at approximately a (AEXX) = 0.75 and 0.82, respectively. Modification of the fraction of exact exchange fraction (a) in a hybrid functional calculations can thus be implemented to tune

material properties to experimentally or theoretically predicted results in this study, as was accomplished in past studies that featured the comparison of band structures achieved in TiO_2 Rutile and Anatase defect states using B3LYP and H&HLYP (Half & Half Lee-Young-Parr) hybrid functional calculations.[S33]

Validation of this result proceeds from performing several variations on the calculations presented above, such as uniformly changing their k-point grid reductions (NKRED).[S33,S46] The data and query associated with this validation (for NKRED = 1 and 2) for Rutile and Anatase at 25% HF exact exchange and 75% PBE exchange (HSE06 functional) is completed below. The similarity of these results, which are equivalent within a tolerance of 0.01 eV, illustrates that experimentally consistent energetic ordering can likely be achieved with either NKRED setting, though the fractions of exact exchange (a) at which this ordering will occur might be slightly different in each case:

$$\Pi_{Morph,NKRED,EOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(NKRED =' 1')or(NKRED =' 2')]$$

$$\sigma \wedge [(Functional =' HSE06')]$$

$$\sigma \wedge [(FractionHF =' 0.250')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEHFRange$$

$$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP]$$

```
#+caption: MySQL query HSE06 NKRED Rutile Anatase
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, NKRED, Eopt, atomnum = [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'
EtoNKRED_dict = {}

for row in db.execute('''
select distinct s.Morph, input.NKRED, cre.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and (input.NKRED='1' or input.NKRED='2')
and mt.Functional='HSE06'
and feh.FractionHF='0.25'
;'''):
    a,b,c,d = row
    try:
        EtoNKRED_dict[ (a,b) ].append( [c,d] )
    except KeyError:
        EtoNKRED_dict[ (a,b) ] = []
        EtoNKRED_dict[ (a,b) ].append( [c,d] )

E_NKRED2 = ( EtoNKRED_dict[ ('Anatase','2') ][0][0] / EtoNKRED_dict[ ('Anatase','2') ][0][1]
 ) - ( EtoNKRED_dict[ ('Rutile','2') ][0][0] / EtoNKRED_dict[ ('Anatase','2') ][0][1] )
E_NKRED1 = ( EtoNKRED_dict[ ('Anatase','1') ][0][0] / EtoNKRED_dict[
('Anatase','1') ][0][1]
```

```
 ) - ( EtoNKRED_dict[ ('Rutile','1') ][0][0] / EtoNKRED_dict[ ('Anatase','1') ][0][1] )

print "HSE06 Functional (a = 0.25):"
print "E(R-A)(NKRED = 2) = " + str(E_NKRED2)
print "E(R-A)(NKRED = 1) = " + str(E_NKRED1) + "\n"
```

```
HSE06 Functional (a = 0.25):
E(R-A)(NKRED = 2) = -0.065773705
E(R-A)(NKRED = 1) = -0.0751067731
```

Further validation of this result proceeds from investigating the discontinuity of the Rutile-Anatase formation energy in the limit of exact HF exchange ($a \rightarrow 1$). As can be shown in the plot above, Rutile-Anatase formation energy monotonically increases until reaching the discrete condition of $a = 1$. However, this plot only depicts results for a values up to 0.95, illustrating the question of whether Rutile-Anatase formation energy decreases at a pronounced, continuous rate between a values of 0.95 and 1.00 or there exists a discontinuity in the limit of exact HF exchange. The analysis presented below indicates that the latter case appears to be true. The magnitude of the DFT energy of Rutile continues to increase monotonically up to $a = 0.99$ for the HSE06 functional and then sharply decreases afterwards, inferring a discontinuity in the Rutile-Anatase formation energy. Further investigation is required to determine whether this is a numerical artifact of the calculations performed in this study, or whether a calculation performed entirely with exact HF exchange observes physical differences relative to its hybridized HF-PBE exchange analogues. Nevertheless, a clear conclusion, which is further assessed in Section 4.2.4, can be drawn, namely that the use of solely HF exchange in resolving TiO_2 (or, more generally, BO_2 or metal oxide) energetic and electronic properties should not be attempted without prior knowledge (experimental or otherwise) of the property assessed. The data and query associated with this validation for Rutile is completed below:

$$\Pi_{Morph,FractionHF,EOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(NKRED =' 2')]$$

$$\sigma \wedge [(Functional =' HSE06')]$$

$$\sigma \wedge [(FractionHF =' 0.250') \vee (FractionHF =' 0.500') \vee (FractionHF =' 0.750')$$

$$\sigma \vee (FractionHF =' 0.825') \vee (FractionHF =' 0.875') \vee (FractionHF =' 0.950')$$

$$\sigma \vee (FractionHF =' 0.975') \vee (FractionHF =' 0.990') \vee (FractionHF =' 1.000')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEHFRange$$

$$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP]$$

```
#+caption: MySQL query HSE06 AEXX limit Rutile
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, FractionHF, Eopt, atomnum = [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct s.Morph, feh.FractionHF, feh.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
```

```
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where s.Morph='Rutile'
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and input.NKRED='2'
and mt.Functional='HSE06'
and (feh.FractionHF='0.250' or feh.FractionHF='0.500' or feh.FractionHF='0.750'
or feh.FractionHF='0.825' or feh.FractionHF='0.875' or feh.FractionHF='0.950'
or feh.FractionHF='0.975' or feh.FractionHF='0.99' or feh.FractionHF='1.000')
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Polymorph += [a]
    FractionHF += [b]
    datapts_dict[row] = datapts_list

Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Morph_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}
    del_list = []

    for k in datapts_dict:
        if k[0] == i:
            SortElist[ k[0] ][ k[1] ] = float( k[2] ) / float( k[3] )
            del_list.append( datapts_dict[k] )
        else:
            pass

    for l in del_list:
        del l

```

```
for i in Morph_list:
    HF_list = SortElist[i].keys()
    HF_list.sort()
    print "HSE06 Functional: \n"

    for k in HF_list:
        print "E(R, a = " + str(k) + ") = " + str(SortElist[i][k])
```

```
HSE06 Functional:


E(R, a = 0.25) = -32.514307105

E(R, a = 0.5) = -39.02279279

E(R, a = 0.75) = -45.891205975

E(R, a = 0.825) = -48.01567473

E(R, a = 0.875) = -49.44671926

E(R, a = 0.95) = -51.61572035

E(R, a = 0.975) = -52.3201595

E(R, a = 0.99) = -52.7569735

E(R, a = 1.0) = -49.43858013
```

4.2.3 Structural Characterizations Relating to Energetic Calculations

In the previous section, Section 4.2.1, the value of the fraction of exact exchange (a) needed to match the Rutile-Anatase formation energy resolved via linear response calculation, previous work,[S33,S46,S54] and manual selection was demonstrated to be possible, namely via either calculated or visual linear interpolation of the formation energy as a function of a. Similarly, in this article and this supporting document, calculated or visual interpolation between data points on pertinent energetic trends is necessary for determining the U or a ranges over which predicted energetic ordering is consistent with experiment. In order to implement this technique, the relationship between pertinent energetic values (e.g.: formation energies) and either U or a must be continuous over the U or a intervals containing an energetic value

of interest. In addition to reviewing the energetic trends formed over U or a themselves, this evaluation can be performed over a representative structural coordinate (i.e.: in this case, equilibrium system volume or V_0), as is detailed in past work.[S38,S45] Furthermore, this evaluation is useful for the resolution of error magnitudes in calculations requiring the calculation of both equilibrium energies (E_0) and volumes (V_0), as is seen in the calculation of phase transition pressures (P) using the thermodynamic equilibrium relationship $\Delta H = \Delta E_0 + P\Delta V_0 = 0$ for bulk materials.[S46] This calculation of error magnitudes in phase transition pressure and energetic calculations can ultimately be used in determining the feasibility of candidate materials for epitaxial stabilization[S3,S55] For all TiO_2 polymorphs, including Rutile, Anatase, Columbite, Brookite, Cotunnite, Pyrite, Fluorite, and Baddeleyite, this continuous volumetric relationship is strongly observed over incrementation of U, as is shown in the Equilibrium Volume ratio ($V_0[U]/V_0[U = 0]$) vs. U value (eV) plot and associated query below. Note that the only exception to the monotonic volume expansion trends derived below occurs in the Baddeleyite polymorph between $U = 5$ and 6 eV, which, as mentioned in Subsection 4.1.3, is beyond the U range at which the Baddeleyite structure is modeled physically.

$$\Pi_{Morph,UValue,VOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Columbite')$$

$$\sigma \wedge (Morph =' Anatase') \vee (Morph =' Brookite')$$

$$\sigma \vee (Morph =' Baddeleyite') \vee (Morph =' Cotunnite')$$

$$\sigma \vee (Morph =' Fluorite') \vee (Morph =' Pyrite')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query Volumetric Expansion Hubbard U
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, Uvalue, Vopt, atomnum = [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select s.Morph, feu.UValue, feu.VOpt, c1.Stoich1 from Structure as s
```

```
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Columbite' or s.Morph='Anatase'
or s.Morph='Brookite' or s.Morph='Baddeleyite' or
s.Morph='Cotunnite' or s.Morph='Fluorite' or s.Morph='Pyrite')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Functional='PBE'
and mt.Software='VASP'
and mt.Method='E'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Polymorph += [a]

    datapts_dict[row] = datapts_list

Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in Morph_list:
    SortElist[i] = {}
    EBsiteMatch[i] = []
    del_list = []

    for j in datapts_dict:
        if j[0] == i:
            SortElist[ j[0] ][ j[1] ] = float( j[2] )
            del_list.append( datapts_dict[j] )
        else:
            pass
    for l in del_list:
        del l

```

```
55  for i in Morph_list:
56      U_list = SortElist[i].keys()
57      U_list.sort()
58      VmapU_list = []
59
60      for j in U_list:
61          VmapU_value = SortElist[i][j] / SortElist[i][0]
62          VmapU_list.append( VmapU_value )
63
64      EBsiteMatch[i].append( VmapU_list )
65      EBsiteMatch[i].append( U_list )
66
67  ax = plt.gca()
68
69  print "PBE Functional, Ti O pseudopotentials:"
70  print "Rutile: " + str(EBsiteMatch['Rutile'][0][0:2])
71  print str(EBsiteMatch['Rutile'][0][2:4])
72  print str(EBsiteMatch['Rutile'][0][4:])
73  print "Anatase: " + str(EBsiteMatch['Anatase'][0][0:2])
74  print str(EBsiteMatch['Anatase'][0][2:4])
75  print str(EBsiteMatch['Anatase'][0][4:])
76  print "Columbite: " + str(EBsiteMatch['Columbite'][0][0:2])
77  print str(EBsiteMatch['Columbite'][0][2:4])
78  print str(EBsiteMatch['Columbite'][0][4:])
79  print "Brookite: " + str(EBsiteMatch['Brookite'][0][0:2])
80  print str(EBsiteMatch['Brookite'][0][2:4])
81  print str(EBsiteMatch['Brookite'][0][4:])
82  print "Baddeleyite: " + str(EBsiteMatch['Baddeleyite'][0][0:2])
83  print str(EBsiteMatch['Baddeleyite'][0][2:4])
84  print str(EBsiteMatch['Baddeleyite'][0][4:])
85  print "Cotunnite: " + str(EBsiteMatch['Cotunnite'][0][0:2])
86  print str(EBsiteMatch['Cotunnite'][0][2:4])
87  print str(EBsiteMatch['Cotunnite'][0][4:])
88  print "Fluorite: " + str(EBsiteMatch['Fluorite'][0][0:2])
89  print str(EBsiteMatch['Fluorite'][0][2:4])
90  print str(EBsiteMatch['Fluorite'][0][4:])
91  print "Pyrite: " + str(EBsiteMatch['Pyrite'][0][0:2])
92  print str(EBsiteMatch['Pyrite'][0][2:4])
93  print str(EBsiteMatch['Pyrite'][0][4:]) + "\n"
94
95  plt.figure(figsize=(3,4))
```

```
plt.plot(EBsiteMatch['Rutile'][1], EBsiteMatch['Rutile'][0], 'ko-')

plt.plot(EBsiteMatch['Anatase'][1], EBsiteMatch['Anatase'][0],
         'bo-', label = r'$\Delta$V$_{R-A}$')
plt.plot(EBsiteMatch['Columbite'][1], EBsiteMatch['Columbite'][0],
         'ro-', label = r'$\Delta$V$_{R-C}$')
plt.plot(EBsiteMatch['Brookite'][1], EBsiteMatch['Brookite'][0],
         'go-', label = r'$\Delta$V$_{R-B}$')

plt.plot(EBsiteMatch['Baddeleyite'][1], EBsiteMatch['Baddeleyite'][0],
         'mo-', label = r'$\Delta$V$_{R-Ba}$')
plt.plot(EBsiteMatch['Cotunnite'][1], EBsiteMatch['Cotunnite'][0],
         'co--', label = r'$\Delta$V$_{R-Co}$')
plt.plot(EBsiteMatch['Fluorite'][1], EBsiteMatch['Fluorite'][0],
         'yo-', label = r'$\Delta$V$_{R-F}$')
plt.plot(EBsiteMatch['Pyrite'][1], EBsiteMatch['Pyrite'][0],
         'co-', label = r'$\Delta$V$_{R-P}$')

plt.xlabel( 'U value (eV)' )
plt.ylim( (1, 1.16) )
plt.ylabel('Volume Expansion Ratio (V$_{O}$[$U$]/V$_{O}$[$U$ = 0])')
plt.legend(loc = 'upper center', prop={'size':8.75}, ncol = 2)

plt.gcf().subplots_adjust(left=0.25)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-volume-RACBBaCoFP-PBE' + '.' + ext, dpi=300)
plt.clf()
```

PBE Functional, Ti O pseudopotentials:

Rutile: [1.0, 1.0092116973552991]

[1.0181651703403998, 1.0287131221094783]

[1.037946909721746, 1.0498999339323443, 1.0600143877763253]

Anatase: [1.0, 1.0100620977329202]

[1.020958013587518, 1.032476362955842]

[1.0445160894630539, 1.0572492151102395, 1.0697087109652161]

```
Columbite: [1.0, 1.0083482157650201]
[1.0184799592004223, 1.0286487338768038]
[1.038630657830466, 1.0506723615511206, 1.0612943523900722]
Brookite: [1.0, 1.0097644502638121]
[1.0206246682114886, 1.0326820337636602]
[1.0439282953823716, 1.0566582761274284, 1.0686844037053145]
Baddeleyite: [1.0, 1.0074677689097928]
[1.0185007770024468, 1.0308657625665223]
[1.0440522276300837, 1.0583815849651379, 1.0415286514664124]
Cotunnite: [1.0, 1.0206163997726225]
[1.0358758713309024, 1.0545842809284809]
[1.0716383198433475, 1.0929062329717745, 1.1085026377077054]
Fluorite: [1.0, 1.011590830068857]
[1.0230170111959742, 1.0347049763850813]
[1.0467830582162361, 1.0591564114368754, 1.0709175944474627]
Pyrite: [1.0, 1.0098949766779464]
[1.0201970652011072, 1.0301228077083382]
[1.0403626730062503, 1.0507848520409468, 1.0607610687581164]
```

In the hybrid functional calculations performed in this article, a noteworthy discontinuity in the trend depicting the relationship between Rutile-Anatase formation energy vs. a occurs in the limit of exact HF exchange ($a \rightarrow 1$). An assessment of the cell volume changes with a in hybrid functional calculations, as can be completed by calculating the ratio of equilibrium volume at a particular a versus that at $a = 0$ ($V_0[a]/V_0[a = 0]$, or $V_0[\%]/V_0[\% = 0]$), is performed below and indicates that changes in Rutile (R) and Anatase (A) cell volumes with a observe a strong, largely linear inverse relationship with corresponding changes in Rutile-Anatase formation energy. Furthermore, the discontinuity shown in the Rutile-Anatase formation energy is inversely replicated in the volumetric data, inferring a physical basis for

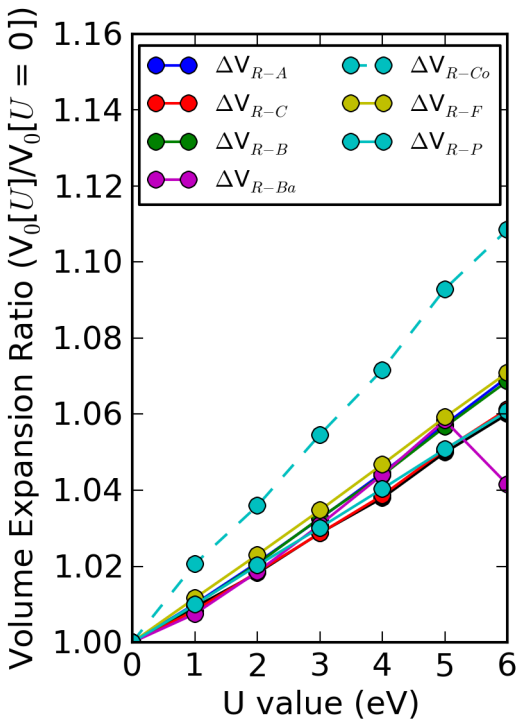

Figure S11

the discontinuity and the loss of monotonicity of the Formation Energy vs. a trend.

$\Pi_{Morph,Functional,FractionHF,VOpt,Stoich1}$

$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')]$

$\sigma \wedge [(At1 =' Ti')]$

$\sigma \wedge [(PseudoB =' PAW - Ti')]$

$\sigma \wedge [(PseudoO =' PAW - O')]$

$\sigma \wedge [(Software =' VASP')]$

$\sigma \wedge [(Method =' E')]$

$\sigma \wedge [(NKRED =' 2')]$

$\sigma \wedge [(Functional =' HSE06') \vee (Functional =' PBE0')]$

$\sigma \wedge [(FractionHF =' 0.250') \vee (FractionHF =' 0.500') \vee (FractionHF =' 0.750')$

$\sigma \vee (FractionHF =' 0.825') \vee (FractionHF =' 0.875') \vee (FractionHF =' 0.950')$

$\sigma \vee (FractionHF =' 1.000')]$

$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEHFRange$

$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP]$

$$\Pi_{Morph,UValue,VOpt,Stoich1}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(UValue =' 0')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query Rutile vs. Anatase Hybrid Functional Volumetric Expansion
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

Polymorph, Functional, FractionHF, Vopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select distinct s.Morph, mt.Functional, feh.FractionHF, feh.VOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
```

```
21  and mt.PseudoB='PAW-Ti'
22  and mt.PseudoO='PAW-O'
23  and mt.Software='VASP'
24  and mt.Method='E'
25  and input.NKRED='2'
26  and (mt.Functional='HSE06' or mt.Functional='PBE0')
27  and (feh.FractionHF='0.250' or feh.FractionHF='0.500' or feh.FractionHF='0.750'
28  or feh.FractionHF='0.825' or feh.FractionHF='0.875'
29  or feh.FractionHF='0.950' or feh.FractionHF='1.000')
30  ;'''):
31      datapts_list = []
32      a,b,c,d,e = row
33      datapts_list.append( (a,b,c,d,e) )
34
35      Polymorph += [a]
36      Functional += [b]
37
38      datapts_dict[row] = datapts_list
39
40  Morph_list = list( set( Polymorph ) )
41  Functional_list = list( set( Functional ) )
42  SortElist = {}
43
44  for i in Morph_list:
45      SortElist[i] = {}
46
47      for j in Functional_list:
48          SortElist[i][j] = {}
49          del_list = []
50
51          for k in datapts_dict:
52              if k[0] == i and k[1] == j:
53                  SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] )
54                  del_list.append( datapts_dict[k] )
55              else:
56                  pass
57
58          for l in del_list:
59              del l
60          if not SortElist[i][j]:
61              del SortElist[i][j]
```

```
Polymorph, Uvalue, Eopt, atomnum = [], [], [], []
datapts_dict = {}

for row in db.execute('''
select s.Morph, feu.UValue, feu.VOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and mt.Functional='PBE'
and feu.UValue='0'
;'''):
    datapts_list = []
    a,b,c,d = row
    datapts_list.append( (a,b,c,d) )

    Polymorph += [a]

    datapts_dict[row] = datapts_list

Morph_list_0 = list( set( Polymorph ) )
for i in Morph_list_0:
    SortElist[i]['PBE0'][float(0.)] = []
    SortElist[i]['HSE06'][float(0.)] = []
    del_list = []

    for j in datapts_dict:
        if j[0] == i:
            SortElist[ j[0] ]['PBE0'][float(0.)] = float( j[2] )
            SortElist[ j[0] ]['HSE06'][float(0.)] = float( j[2] )
            del_list.append( datapts_dict[j] )
        else:
            pass
    for l in del_list:
        del l
```

```
EBsiteMatch = {}
for i in Morph_list:
    EBsiteMatch[i] = {}

    for j in Functional_list:
        EBsiteMatch[i][j] = []
        HF_list = SortElist[i][j].keys()
        HF_list.sort()
        VmapHF_list = []

        for k in HF_list:
            VmapHF_value = SortElist[i][j][k] / SortElist[i][j][float(0.0)]
            VmapHF_list.append( VmapHF_value )

        EBsiteMatch[i][j].append( VmapHF_list )
        EBsiteMatch[i][j].append( HF_list )

ax = plt.gca()

print "PBE0 Functional:"
print "Rutile: " + str(EBsiteMatch['Rutile']['PBE0'][0][0:2])
print str(EBsiteMatch['Rutile']['PBE0'][0][2:5])
print str(EBsiteMatch['Rutile']['PBE0'][0][5:])
print "Anatase: " + str(EBsiteMatch['Anatase']['PBE0'][0][0:2])
print str(EBsiteMatch['Anatase']['PBE0'][0][2:5])
print str(EBsiteMatch['Anatase']['PBE0'][0][5:]) + "\n"

print "HSE06 Functional:"
print "Rutile: " + str(EBsiteMatch['Rutile']['HSE06'][0][0:2])
print str(EBsiteMatch['Rutile']['HSE06'][0][2:5])
print str(EBsiteMatch['Rutile']['HSE06'][0][5:])
print "Anatase: " + str(EBsiteMatch['Anatase']['HSE06'][0][0:2])
print str(EBsiteMatch['Anatase']['HSE06'][0][2:5])
print str(EBsiteMatch['Anatase']['HSE06'][0][5:]) + "\n"

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Rutile']['PBE0'][1], EBsiteMatch['Rutile']['PBE0'][0],
'ko-', label = r'$\Delta$V$_{R,PBE0}$')
plt.plot(EBsiteMatch['Rutile']['HSE06'][1], EBsiteMatch['Rutile']['HSE06'][0],
'bo-', label = r'$\Delta$V$_{R,HSE06}$')
```

```
plt.plot(EBsiteMatch['Anatase']['PBE0'][1], EBsiteMatch['Anatase']['PBE0'][0],
'ro-', label = r'$\Delta$V$_{A,PBE0}$')
plt.plot(EBsiteMatch['Anatase']['HSE06'][1], EBsiteMatch['Anatase']['HSE06'][0],
'go-', label = r'$\Delta$V$_{A,HSE06}$')

plt.xlabel( 'Exact Exchange Fraction (%)' )
plt.ylim( (0.9, 1.0) )
plt.ylabel('Volume Expansion Ratio (V$_{0}$[%]/V$_{0}$[% = 0])')
plt.legend(loc = 'lower left', prop={'size':9})

plt.gcf().subplots_adjust(left=0.25)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-volume-RA-PBE0HSE06' + '.' + ext, dpi=300)
plt.clf()
```

PBE0 Functional:

Rutile: [1.0, 0.96559541659233017]

[0.93939686487513052, 0.91805253415606602, 0.91353617288593036]

[0.9100241485596523, 0.90428664242169765, 0.96030836837451239]

Anatase: [1.0, 0.96702679715448925]

[0.94176296441954122, 0.92245482769244824, 0.91755354860712879]

[0.91478602932862785, 0.90925052592389111, 0.96399321803820104]

HSE06 Functional:

Rutile: [1.0, 0.96681851600776947]

[0.94136523175064557, 0.92140512233432703, 0.91593112697565338]

[0.9123427117168208, 0.90695563940846846, 0.9634757798018444]

Anatase: [1.0, 0.96868074485162858]

[0.94412405775022257, 0.92527235144683928, 0.92088458670742213]

[0.91795013154163063, 0.9134055543595857, 0.9687025767631664]

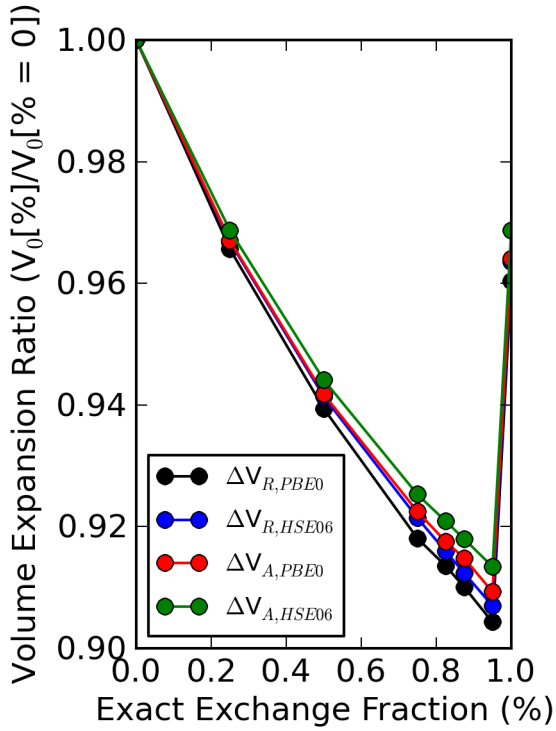

Figure S12

In previous work,[S46] the relationship between several structural factors and incrementation of the U parameter on Ti $3d$ cations was demonstrated. The c/a ratio assessment performed in this previous work observed particular similarity to the equilibrium volume results shown above, in that both the equilibrium volume ratio vs. a (or %, shown above) plot and the c/a ratio vs. a (fraction of exact HF exchange)[S46] observed the same linearly inverse trend that largely maintains monotonicity until a discontinuous point. At the discontinuous point, which occurs at a U value between 8 and 10 eV in the c/a vs. U plot shown in this previous work[S46] and at $a \rightarrow 1$ in this work, both trends lose monotonicity and proceed in an inverted direction. Given that this trend inversion occurs as a function of both U and a and is inversely proportional to comparable trends in formation energetics, the argument that trend inversion is an artifact of solely hybrid functional calculations is likely incorrect and a physical justification for this inversion appears feasible. The c/a ratio vs. a (fraction of exact HF exchange) pertaining to results achieved in this article is queried and plotted

below, yielding an effect matching that illustrated in past work for U-based calculations.[S46] Considering that the c/a ratio changes in Rutile shown below are directly proportional to those of the Rutile-Anatase formation energy shown as a function of a, comparable variation in the c/a ratio Anatase is comparable to the change in either cell volume with changes in a, and both of these conclusions were demonstrated in past work with variation in U,[S46] the discontinuous change in Rutile-Anatase formation energy ordering appears to be most consistently linked to structural changes in Rutile, though Anatase observes a distinct structural relationship with this energetic change as well.

$$\Pi_{Energy,EOpt,HFSCREEN,AEXX,PrimVec1,PrimVec3}$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')]$$

$$\sigma \wedge [(At1 =' Ti')]$$

$$\sigma \wedge [(PseudoB =' PAW - Ti')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(NKRED =' 2')]$$

$$\sigma \wedge [(Functional =' HSE06') \vee (Functional =' PBE0')]$$

$$\sigma \wedge [(FractionHF =' 0.250') \vee (FractionHF =' 0.500') \vee (FractionHF =' 0.750')$$

$$\sigma \vee (FractionHF =' 0.825') \vee (FractionHF =' 0.875') \vee (FractionHF =' 0.950')$$

$$\sigma \vee (FractionHF =' 1.000')]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEHFRange$$

$$\bowtie CalcResultEnergetics \bowtie ParametersInputVASP \bowtie ParametersPosFinal]$$

```python
#+caption: MySQL query Rutile vs. Anatase Hybrid Functional c/a Expansion
import matplotlib.pyplot as plt
import sqlite3
import numpy as np

db = sqlite3.connect('ITEOO_data.sqlite')

HFtype, FractionHF = [], []

EtoEOptHF_dict = {}
NULL_CHAR = '-'
SColon_CHAR = ';'
Morph_list = [ 'Rutile', 'Anatase' ]
WRT_Frac = 0.25

for row in db.execute('''
select distinct cre.Energy, cre.EOpt from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersInputVASP as input on input.Energy=cre.Energy
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and input.NKRED='2'
and (mt.Functional='HSE06' or mt.Functional='PBE0')
and (feh.FractionHF='0.250' or feh.FractionHF='0.500' or feh.FractionHF='0.750'
or feh.FractionHF='0.825' or feh.FractionHF='0.875'
or feh.FractionHF='0.950' or feh.FractionHF='1.000')
;'''):
    a,b = row
    try:
        EtoEOptHF_dict[b].append(a)
    except KeyError:
        EtoEOptHF_dict[b] = []
        EtoEOptHF_dict[b].append(a)
```

```
Emin_list = []
for i in EtoEOptHF_dict.keys():
    Ediff_list = []

    for j in EtoEOptHF_dict[i]:
        Ediff_value = abs( float(i) - float(j) )
        Ediff_list.append( Ediff_value )

    Emindex = Ediff_list.index( min( Ediff_list ) )
    Emin_list.append( EtoEOptHF_dict[i][ Emindex ] )

datapts_dict = {}
for row in db.execute('''
select distinct input.Energy, input.HFSCREEN, input.AEXX, pos.PrimVec1, pos.PrimVec3 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Composition2 as c2 on c1.MID=c2.MID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEHFRange as feh on feh.CID=mt.CID
inner join CalcResultEnergetics as cre on cre.EOpt=feh.EOpt
inner join ParametersPosFinalVASP as pos on pos.Energy=cre.Energy
inner join ParametersInputVASP as input on input.Energy=pos.Energy
where (s.Morph='Rutile' or s.Morph='Anatase')
and c1.At1='Ti'
and mt.PseudoB='PAW-Ti'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and mt.Method='E'
and input.NKRED='2'
and (mt.Functional='HSE06' or mt.Functional='PBE0')
and (feh.FractionHF='0.250' or feh.FractionHF='0.500' or feh.FractionHF='0.750'
or feh.FractionHF='0.825' or feh.FractionHF='0.875'
or feh.FractionHF='0.950' or feh.FractionHF='1.000')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    HFtype += [b]
    FractionHF += [c]

```

```
    if a in Emin_list:
        datapts_dict[row] = datapts_list

HFtype_list = list( set( HFtype ) )
FractionHF_list = list( set( FractionHF ) )
SortElist = {}
EBsiteMatch = {}
HFlabel_list = []
for i in HFtype_list:
    if i == NULL_CHAR:
        HFtype_val = 'PBE0'
    else:
        HFtype_val = 'HSE06'

    HFlabel_list.append( HFtype_val )

    SortElist[ HFtype_val ] = {}
    EBsiteMatch[ HFtype_val ] = {}

    for j in Morph_list:
        SortElist[ HFtype_val ][j] = {}
        EBsiteMatch[ HFtype_val ][j] = []

    del_list = []

    for k in datapts_dict:
        if k[1] == i:
            a_string = str( k[3] ).split(SColon_CHAR)
            c_string = str( k[4] ).split(SColon_CHAR)
            a_vec = np.zeros( len(a_string) )
            c_vec = np.zeros( len(c_string) )

            for l in range( len(a_string) ):
                a_vec[l] = float( a_string[l] )
                c_vec[l] = float( c_string[l] )

            a_dot = np.dot( a_vec, a_vec )
            c_dot = np.dot( c_vec, c_vec )

            ca_ratio = ( c_dot / a_dot )**(0.5)
            if ca_ratio < 1.:
```

```
                SortElist[ HFtype_val ]['Rutile'][ float(k[2]) ] = ca_ratio
            else:
                SortElist[ HFtype_val ]['Anatase'][ float(k[2]) ] = (2. * c_vec[2] ) / (
                 a_dot )**(0.5)
            del_list.append( datapts_dict[k] )
        else:
            pass

        for l in del_list:
            del l

for i in HFlabel_list:
    for j in Morph_list:
        HF_list = SortElist[i][j].keys()
        camapHF_list = []
        HF_list.sort()

        for k in HF_list:
            camapHF_value = SortElist[i][j][k] / SortElist[i][j][WRT_Frac]
            camapHF_list.append( camapHF_value )

        EBsiteMatch[i][j].append( camapHF_list )
        EBsiteMatch[i][j].append( HF_list )

ax = plt.gca()

print "PBE0 Functional:"
print "Rutile: " + str(EBsiteMatch['PBE0']['Rutile'][0][0:2])
print str(EBsiteMatch['PBE0']['Rutile'][0][2:4])
print str(EBsiteMatch['PBE0']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['PBE0']['Anatase'][0][0:2])
print str(EBsiteMatch['PBE0']['Anatase'][0][2:4])
print str(EBsiteMatch['PBE0']['Anatase'][0][4:])+ "\n"

print "HSE06 Functional:"
print "Rutile: " + str(EBsiteMatch['HSE06']['Rutile'][0][0:2])
print str(EBsiteMatch['HSE06']['Rutile'][0][2:4])
print str(EBsiteMatch['HSE06']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['HSE06']['Anatase'][0][0:2])
print str(EBsiteMatch['HSE06']['Anatase'][0][2:4])
print str(EBsiteMatch['HSE06']['Anatase'][0][4:]) + "\n"
```

```python

plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['PBE0']['Rutile'][1], EBsiteMatch['PBE0']['Rutile'][0],
         'ko-', label = r'$\Delta$c/a$_{R,PBE0}$')
plt.plot(EBsiteMatch['HSE06']['Rutile'][1], EBsiteMatch['HSE06']['Rutile'][0],
         'bo-', label = r'$\Delta$c/a$_{R,HSE06}$')
plt.plot(EBsiteMatch['PBE0']['Anatase'][1], EBsiteMatch['PBE0']['Anatase'][0],
         'ro-', label = r'$\Delta$c/a$_{A,PBE0}$')
plt.plot(EBsiteMatch['HSE06']['Anatase'][1], EBsiteMatch['HSE06']['Anatase'][0],
         'go-', label = r'$\Delta$c/a$_{A,HSE06}$')

plt.xlabel( 'Exact Exchange Fraction (%)' )
labels = [ WRT_Frac, 0.50, 0.75, 1.00 ]
plt.xticks(labels)
plt.ylim( (0.985, 1.020) )
plt.ylabel('c/a Ratio (c/a[%]/c/a[% = ' + str(WRT_Frac) + '])')
plt.legend(loc = 'upper left', prop={'size':8.5})

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-caratio-RA-PBE0HSE06' + '.' + ext, dpi=300)
plt.clf()
```

PBE0 Functional:

Rutile: [1.0, 1.0063091478087522]

[1.0113084048147694, 1.0128416876368977]

[1.0149720206132735, 1.0157098165835381, 1.0107930988102263]

Anatase: [1.0, 0.99612175090170263]

[0.98984577466746515, 0.99116042133601479]

[0.99358787868204745, 0.99365957535510041, 1.0014326837470342]

HSE06 Functional:

Rutile: [1.0, 1.0044737182863499]

[1.0083325068212536, 1.0098032610532839]

```
[1.0117841545742048, 1.0119760388900785, 1.0077913137599737]
Anatase: [1.0, 1.001498966435564]
[0.996251131410964, 0.99652933956113687]
[0.99625113130970622, 0.99763376373291413, 1.0055444099267918]
```

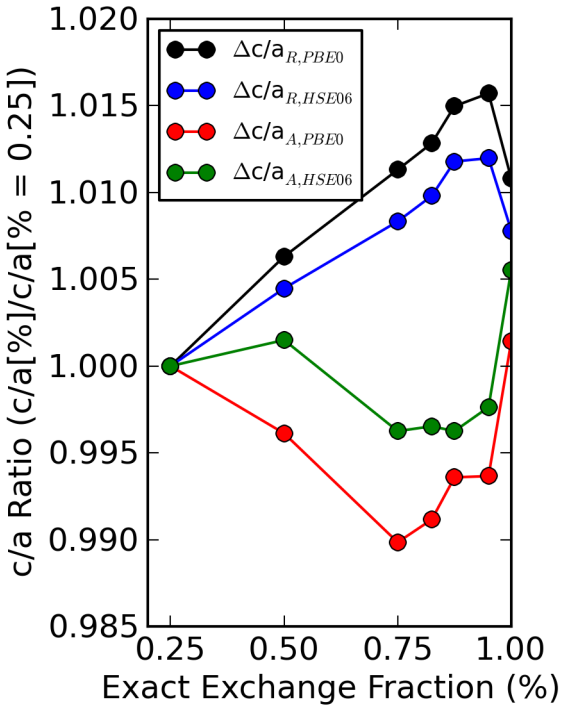

Figure S13

4.2.4 Energetic Comparisons: Hubbard U + Linear Response vs. Hybrid Functionals

As shown in the article and Section 4.2.2, the PBE0 and HSE06 functionals approach respective Rutile-Anatase formation energy maxima of approximately +0.045 and +0.030 eV/f.u. TiO_2 as a, the fraction of exact HF exchange in a hybrid functional calculation, approaches unity. At the limiting value of $a = 1$ itself, the Rutile-Anatase formation energies of both functions discontinuously and non-monotonically decline to -0.051 and -0.039 eV/f.u. TiO_2 for the PBE0 and HSE06 functionals, respectively. Both PBE0 and HSE06 Rutile-Anatase

formation energy trends have minima in the limit of solely PBE exchange ($a \rightarrow 0$), or approximately -0.081 eV/f.u. TiO_2. For the general case of any set of BO_2 materials, no currently available prior knowledge can be used to establish a fraction of exact exchange suitable for the precise predictive calculation of pertinent material properties, such as relative formation energetics. Therefore, in order to evaluate a maximum expectation of the energetic error associated with arbitrarily imposing a value of a in BO_2 formation energy calculations, two particular cases are considered. Given that calculations performed without considering or only considering HF exchange represent the extrema of the values of a that can be selected, the minimum value of the Rutile-Anatase formation energy is represented by the case of completely ignoring HF exchange, and the value approaching $a \rightarrow 1$ represents the maximum value of this formation energy, the quantities ($E_{\text{R-A},a \rightarrow 1}$ - $E_{\text{R-A},a = 1}$) and ($E_{\text{R-A},a \rightarrow 1}$ - $E_{\text{R-A},a = 0}$) represent maximum error values that can result from setting a to an arbitrary, single value in TiO_2 Rutile and Anatase formation energies. For the PBE0 and HSE06 functionals, the ($E_{\text{R-A},a \rightarrow 1}$ - $E_{\text{R-A},a = 1}$) and ($E_{\text{R-A},a \rightarrow 1}$ - $E_{\text{R-A},a = 0}$) values are approximately (+0.096, +0.126) eV/f.u. TiO_2 (PBE0) and (+0.069, +0.111) eV/f.u. TiO_2 (HSE06), respectively. Given that epitaxial stabilization occurs within an energetic window of 0.1-0.2 eV between two phases[S3] and these maximum errors observe the same order of magnitude of this stabilization window, the selection of an arbitrary value of a can significantly impact the prediction of epitaxial stabilization candidates relative to the criterion of feasible energetic stabilization.

In extending this error analysis to other BO_2 (B = Ti, V, Ru, Ir) systems, pertinent formation energetic results[S3] for the TiO_2, VO_2, RuO_2, and IrO_2 systems are queried, plotted, and analyzed below. Though these plots are assembled as a function of U rather than a due to issues associated with computational expense, the analysis below is accomplished to simulate the comparative evaluation of epitaxial stabilization candidacy for systems that exist either with or without prior knowledge of a first-principles value of U capable of correcting spurious electron-electron interactions. In the case of systems for which prior knowledge of a predicted

value of U does not exist, a U value that is resolved by fitting to satisfy other properties qualitatively, another non-predictive method, or comparison with a similar system is initially selected. This selected value of U, which was not predicted for the system associated with it, has a unspecified, likely higher level of error on it than corresponding U values calculated in this paper due to the crude qualitative fitting or other estimation method used to resolve it. In performing a hybrid functional calculation with an a parameter of arbitrary magnitude on such a BO_2 system based on a fitted U value, errors of the form ($E_{R\text{-}x,a_2}$ - $E_{R\text{-}x,a_1}$) are introduced, where x represents a non-Rutile (R) polymorph for which the Rutile-x formation energy is calculated and a values (a_1 and a_2) represent the HF exact exchange fractions of two calculations performed on the same formation energy trend. The corresponding maximum quantities for TiO_2 error calculated above approximately represent the magnitude of possible corresponding errors in BO_2 systems, given that either an arbitrary value of a is selected to calculate a pertinent BO_2 formation energy or an a value is selected via matching the energetics achieved via a qualitatively selected or fitted U. However, as shown below, the total variation of formation energies across trends of incremented U is generally larger in other BO_2 systems than in TiO_2, thus actual errors resulting from arbitrary or U fitted a selection will likely be larger than those resulting from adapting corresponding TiO_2 system errors.

$$\Pi_{At1,Morph,UValue,EOpt,Stoich1}$$

$$\sigma \wedge [(Functional =' PBE')]$$

$$\sigma \wedge [(PseudoO =' PAW - O')]$$

$$\sigma \wedge [(Software =' VASP')]$$

$$\sigma \wedge [(Method =' E')]$$

$$\sigma \wedge [(UValue =' 0') \vee (UValue =' 1') \vee (UValue =' 2') \vee (UValue =' 3')$$

$$\sigma \vee (UValue =' 4') \vee (UValue =' 5') \vee (UValue =' 6')]$$

$$\sigma \wedge [[(PseudoB =' PAW - Ti') \vee (PseudoB =' PAW - V')]$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Anatase')$$

$$\sigma \vee (Morph =' Columbite') \vee (Morph =' Brookite')]]$$

$$\sigma \wedge [[(PseudoB =' PAW - Ru') \vee (PseudoB =' PAW - Ir')]$$

$$\sigma \wedge [(Morph =' Rutile') \vee (Morph =' Pyrite') \vee (Morph =' Columbite')]]$$

$$[Structure \bowtie Composition1 \bowtie Metadata \bowtie FormEURange]$$

```
#+caption: MySQL query Epitaxial Comparison Ti V Ru Ir polymorph
import matplotlib.pyplot as plt
import sqlite3

db = sqlite3.connect('ITEOO_data.sqlite')

atom1, Polymorph, Uvalue, Eopt, atomnum = [], [], [], [], []

datapts_dict = {}
WRT_Morph = 'Rutile'

for row in db.execute('''
select c1.At1, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
```

```
inner join FormEURange as feu on feu.CID=mt.CID
where mt.Functional='PBE'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
and (mt.PseudoB='PAW-Ti' or mt.PseudoB='PAW-V')
and (s.Morph='Rutile' or s.Morph='Anatase'
or   s.Morph='Columbite' or s.Morph='Brookite')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    atom1 += [a]
    Polymorph += [b]

    datapts_dict[row] = datapts_list

At1_list = list( set( atom1 ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
EBsiteMatch = {}

for i in At1_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Morph_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
                del_list.append( datapts_dict[k] )
            else:
                pass

        for l in del_list:
```

```
            del l

for i in At1_list:
    for j in Morph_list:
        U_list = SortElist[i][j].keys()
        U_list.sort()
        EmapU_list = []

        for k in U_list:
            EmapU_value = SortElist[i][j][k] - SortElist[i][WRT_Morph][k]
            EmapU_list.append( EmapU_value )

        EBsiteMatch[i][j].append( EmapU_list )
        EBsiteMatch[i][j].append( U_list )

for row in db.execute('''
select c1.At1, s.Morph, feu.UValue, feu.EOpt, c1.Stoich1 from Structure as s
inner join Composition1 as c1 on c1.SID=s.SID
inner join Metadata as mt on (mt.MID=c1.MID and mt.SID=c1.SID)
inner join FormEURange as feu on feu.CID=mt.CID
where mt.Functional='PBE'
and mt.PseudoO='PAW-O'
and mt.Software='VASP'
and (feu.UValue='0' or feu.UValue='1' or feu.UValue='2' or feu.UValue='3'
or feu.UValue='4' or feu.UValue='5' or feu.UValue='6')
and mt.PseudoB='PAW-Ru' or mt.PseudoB='PAW-Ir'
and (s.Morph='Rutile' or s.Morph='Pyrite'
or   s.Morph='Columbite')
;'''):
    datapts_list = []
    a,b,c,d,e = row
    datapts_list.append( (a,b,c,d,e) )

    atom1 += [a]
    Polymorph += [b]

    datapts_dict[row] = datapts_list

At1_list = list( set( atom1 ) )
Morph_list = list( set( Polymorph ) )
SortElist = {}
```

```
EBsiteMatch = {}

for i in At1_list:
    SortElist[i] = {}
    EBsiteMatch[i] = {}

    for j in Morph_list:
        SortElist[i][j] = {}
        EBsiteMatch[i][j] = []
        del_list = []

        for k in datapts_dict:
            if k[0] == i and k[1] == j:
                SortElist[ k[0] ][ k[1] ][ k[2] ] = float( k[3] ) / float( k[4] )
                del_list.append( datapts_dict[k] )
            else:
                pass

        for l in del_list:
            del l

for i in At1_list:
    for j in Morph_list:
        U_list = SortElist[i][j].keys()
        U_list.sort()
        EmapU_list = []

        for k in U_list:
            EmapU_value = SortElist[i][j][k] - SortElist[i][WRT_Morph][k]
            EmapU_list.append( EmapU_value )

        EBsiteMatch[i][j].append( EmapU_list )
        EBsiteMatch[i][j].append( U_list )


ax = plt.gca()

print r'PBE Functional, TiO2:'
print "Rutile: " + str(EBsiteMatch['Ti']['Rutile'][0][0:2])
print str(EBsiteMatch['Ti']['Rutile'][0][2:4])
print str(EBsiteMatch['Ti']['Rutile'][0][4:])
```

```
print "Anatase: " + str(EBsiteMatch['Ti']['Anatase'][0][0:2])
print str(EBsiteMatch['Ti']['Anatase'][0][2:4])
print str(EBsiteMatch['Ti']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['Ti']['Columbite'][0][0:2])
print str(EBsiteMatch['Ti']['Columbite'][0][2:4])
print str(EBsiteMatch['Ti']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['Ti']['Brookite'][0][0:2])
print str(EBsiteMatch['Ti']['Brookite'][0][2:4])
print str(EBsiteMatch['Ti']['Brookite'][0][4:]) + "\n"

print r'PBE Functional, VO2:'
print "Rutile: " + str(EBsiteMatch['V']['Rutile'][0][0:2])
print str(EBsiteMatch['V']['Rutile'][0][2:4])
print str(EBsiteMatch['V']['Rutile'][0][4:])
print "Anatase: " + str(EBsiteMatch['V']['Anatase'][0][0:2])
print str(EBsiteMatch['V']['Anatase'][0][2:4])
print str(EBsiteMatch['V']['Anatase'][0][4:])
print "Columbite: " + str(EBsiteMatch['V']['Columbite'][0][0:2])
print str(EBsiteMatch['V']['Columbite'][0][2:4])
print str(EBsiteMatch['V']['Columbite'][0][4:])
print "Brookite: " + str(EBsiteMatch['V']['Brookite'][0][0:2])
print str(EBsiteMatch['V']['Brookite'][0][2:4])
print str(EBsiteMatch['V']['Brookite'][0][4:])+ "\n"

print r'PBE Functional, RuO2:'
print "Rutile: " + str(EBsiteMatch['Ru']['Rutile'][0][0:2])
print str(EBsiteMatch['Ru']['Rutile'][0][2:4])
print str(EBsiteMatch['Ru']['Rutile'][0][4:])
print "Pyrite: " + str(EBsiteMatch['Ru']['Pyrite'][0][0:2])
print str(EBsiteMatch['Ru']['Pyrite'][0][2:4])
print str(EBsiteMatch['Ru']['Pyrite'][0][4:])
print "Columbite: " + str(EBsiteMatch['Ru']['Columbite'][0][0:2])
print str(EBsiteMatch['Ru']['Columbite'][0][2:4])
print str(EBsiteMatch['Ru']['Columbite'][0][4:]) + "\n"

print r'PBE Functional, IrO2:'
print "Rutile: " + str(EBsiteMatch['Ir']['Rutile'][0][0:2])
print str(EBsiteMatch['Ir']['Rutile'][0][2:4])
print str(EBsiteMatch['Ir']['Rutile'][0][4:])
print "Pyrite: " + str(EBsiteMatch['Ir']['Pyrite'][0][0:2])
print str(EBsiteMatch['Ir']['Pyrite'][0][2:4])
```

```
print str(EBsiteMatch['Ir']['Pyrite'][0][4:])
print "Columbite: " + str(EBsiteMatch['Ir']['Columbite'][0][0:2])
print str(EBsiteMatch['Ir']['Columbite'][0][2:4])
print str(EBsiteMatch['Ir']['Columbite'][0][4:])+ "\n"


plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Ti']['Rutile'][1], EBsiteMatch['Ti']['Rutile'][0], 'k-')
plt.plot(EBsiteMatch['Ti']['Anatase'][1], EBsiteMatch['Ti']['Anatase'][0],
         'bo-', label = r'$\Delta$E$_{R-A}$')
plt.plot(EBsiteMatch['Ti']['Columbite'][1], EBsiteMatch['Ti']['Columbite'][0],
         'ro-', label = r'$\Delta$E$_{R-C}$')
plt.plot(EBsiteMatch['Ti']['Brookite'][1], EBsiteMatch['Ti']['Brookite'][0],
         'go-', label = r'$\Delta$E$_{R-B}$')

plt.text(0.595, 2.75, 'B = Ti',
        style = 'italic', color = 'white', fontsize = 24,
    transform = ax.transAxes,
    bbox={'facecolor':'b', 'alpha':0.5, 'pad':5})

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.1) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.27)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/TiO2-epitax-RACB-PBE' + '.' + ext, dpi=300)
plt.clf()


plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['V']['Rutile'][1], EBsiteMatch['V']['Rutile'][0], 'k-')
plt.plot(EBsiteMatch['V']['Anatase'][1], EBsiteMatch['V']['Anatase'][0],
         'bo-', label = r'$\Delta$E$_{R-A}$')
plt.plot(EBsiteMatch['V']['Columbite'][1], EBsiteMatch['V']['Columbite'][0],
         'ro-', label = r'$\Delta$E$_{R-C}$')
plt.plot(EBsiteMatch['V']['Brookite'][1], EBsiteMatch['V']['Brookite'][0],
         'go-', label = r'$\Delta$E$_{R-B}$')
```

```
plt.text(0.565, 2.75, 'B = V',
        style = 'italic', color = 'white', fontsize = 24,
    transform = ax.transAxes,
    bbox={'facecolor':'b', 'alpha':0.5, 'pad':5})

plt.xlabel( 'U value (eV)' )
plt.ylim( (0.0, 0.8) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.23)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/VO2-epitax-RACB-PBE' + '.' + ext, dpi=300)
plt.clf()


plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Ru']['Rutile'][1], EBsiteMatch['Ru']['Rutile'][0], 'k-')
plt.plot(EBsiteMatch['Ru']['Pyrite'][1], EBsiteMatch['Ru']['Pyrite'][0],
         'ro-', label = r'$\Delta$E$_{R-P}$')
plt.plot(EBsiteMatch['Ru']['Columbite'][1], EBsiteMatch['Ru']['Columbite'][0],
         'bo-', label = r'$\Delta$E$_{R-C}$')

plt.text(0.54, 2.75, 'B = Ru',
    style = 'italic', color = 'white', fontsize = 24,
    transform = ax.transAxes,
    bbox={'facecolor':'b', 'alpha':0.5, 'pad':5})

plt.xlabel( 'U value (eV)' )
plt.ylim( (-0.1, 0.5) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.25)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/RuO2-epitax-RPC-PBE' + '.' + ext, dpi=300)
```

```
plt.clf()


plt.figure(figsize=(3,4))
plt.plot(EBsiteMatch['Ir']['Rutile'][1], EBsiteMatch['Ir']['Rutile'][0], 'k-')
plt.plot(EBsiteMatch['Ir']['Pyrite'][1], EBsiteMatch['Ir']['Pyrite'][0],
         'ro-', label = r'$\Delta$E$_{R-P}$')
plt.plot(EBsiteMatch['Ir']['Columbite'][1], EBsiteMatch['Ir']['Columbite'][0],
         'bo-', label = r'$\Delta$E$_{R-C}$')

plt.text(0.580, 2.75, 'B = Ir',
    style = 'italic', color = 'white', fontsize = 24,
    transform = ax.transAxes,
    bbox={'facecolor':'b', 'alpha':0.5, 'pad':5})

plt.xlabel( 'U value (eV)' )
plt.ylim( (0.2, 0.45) )
plt.ylabel('Energy Difference (eV/f.u.)')
plt.legend(loc = 'upper left', prop={'size':10})

plt.gcf().subplots_adjust(left=0.25)
plt.gcf().subplots_adjust(bottom=0.11)

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('./figures/IrO2-epitax-RPC-PBE' + '.' + ext, dpi=300)
plt.clf()
```

```
PBE Functional, TiO2:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [-0.081062179999999984, -0.052122670000002813]

[-0.02370224500000262, 0.005027775000002066]

[0.033729065000002834, 0.062429919999999584, 0.090796440000001866]

Columbite: [-0.0041790349999999421, 0.012991809999999049]

[0.025531667499997468, 0.034880245000000087]
```

[0.042688772500000027, 0.049023697500000907, 0.054641072500000831]

Brookite: [-0.040741385000000463, -0.018720677500002836]

[4.6207499998729418e-05, 0.017773080000001329]

[0.035646842500000275, 0.052357034999999996, 0.069235472500000839]

PBE Functional, VO2:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Anatase: [0.086854330000001312, 0.088153319999999979]

[0.12906397499999755, 0.19266358999999866]

[0.61088731499999938, 0.69807349999999957, 0.7769986650000007]

Columbite: [0.019272935000000047, 0.010262215000000907]

[0.031495754999998127, 0.087489007500000326]

[0.11946426749999972, 0.13891211500000011, 0.15013168249999964]

Brookite: [0.035397034999999022, 0.026805509999999089]

[0.020633692499998801, 0.072716525000000587]

[0.11305117499999895, 0.1444874425000009, 0.16908419250000151]

PBE Functional, RuO2:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Pyrite: [0.2214406275000016, 0.25562430000000091]

[0.29623150499999795, 0.28924779500000142]

[0.29643102999999726, 0.37366771250000141, 0.47572785000000195]

Columbite: [0.15043195249999997, 0.18392020499999973]

[0.22020459749999688, 0.092317645000001392]

[-0.042534200000002187, -0.060944602499997558, -0.06880084499999839]

PBE Functional, IrO2:

Rutile: [0.0, 0.0]

[0.0, 0.0]

[0.0, 0.0, 0.0]

Pyrite: [0.28936491999999703, 0.29814440000000175]

[0.29797718000000017, 0.2933258725000023]

[0.31908643250000068, 0.37010447500000154, 0.44457876250000083]

Columbite: [0.25781293000000005, 0.25812473750000109]

[0.25408133750000061, 0.23397840250000002]

[0.22027735000000348, 0.21268551500000044, 0.20206573000000105]

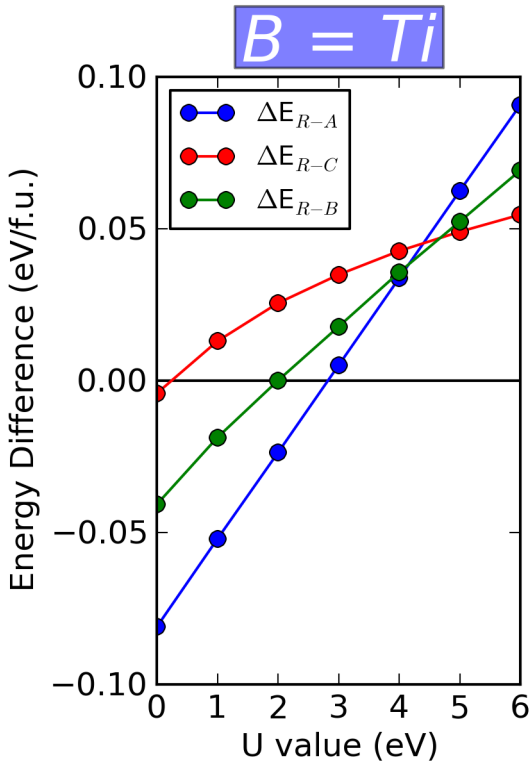

Figure S14

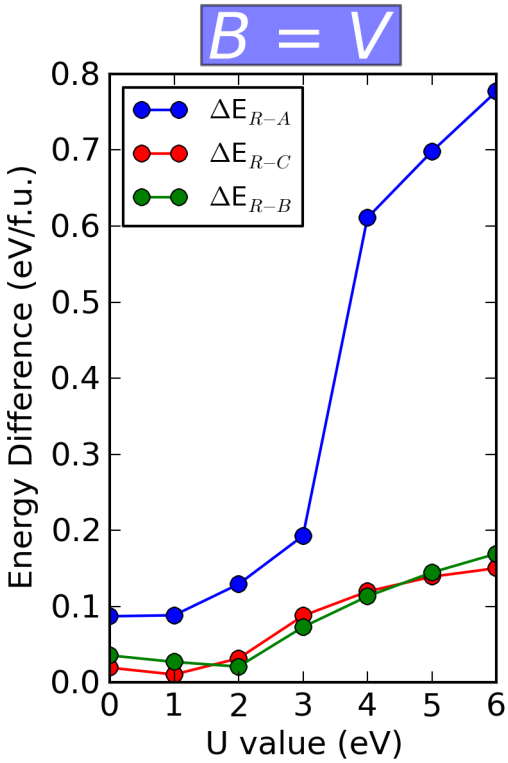

Figure S15

For the TiO_2 system depicted above, Rutile, Anatase, Brookite, and Columbite polymorphs are featured in formation energy calculations set with respect to Rutile. Given an epitaxial stabilization window of 0.1-0.2 eV, stabilization of Anatase, Columbite, and Brookite polymorphs from a Rutile precursor is highly possible in the TiO_2 system. The errors represented above, which range from 0.068-0.127 eV, are within an order of magnitude of the epitaxial stabilization range. When using the U value of 3.0 eV resolved in this work, the Rutile-Anatase, Rutile-Columbite, and Rutile-Brookite formation energies are approximately +0.005, +0.035, and +0.018 eV/f.u. TiO_2, respectively. Given the error ranges above and the formation energy magnitudes, selection of an arbitrary value of U could not only affect epitaxial stabilization candidacy predictions but also relative energetic ordering predictions, thus first-principles calculation of U values is essential to predicting relative energetics in the TiO_2 system.

For the VO_2 system depicted above, Rutile, Anatase, Brookite, and Columbite poly-

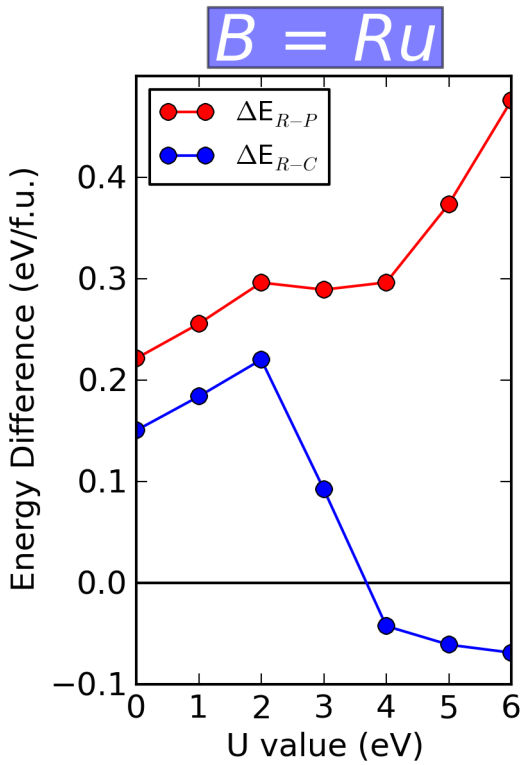

Figure S16

morphs are featured in formation energy calculations set with respect to Rutile. Given an epitaxial stabilization window of 0.1-0.2 eV and the errors mentioned above, stabilization of Anatase, Columbite, and Brookite polymorphs from a Rutile precursor is possible in the VO_2 system and error in calculations involving them can affect their predicted energetic stabilities. For VO_2, an appropriate selection of a U parameter for all polymorphs is more ambiguous, with previous research suggesting a U value between 3.0 and 4.0 eV. However, given the formation energy trends shown above, this level of precision is inadequate for suggesting the candidacy of VO_2 Anatase, though adequate for suggesting that of Columbite and Brookite. Values of U suggested for VO_2 Rutile, which given the calculated U values of TiO_2 polymorphs in this article may directly correspond to the U values of other VO_2 polymorphs, include 3.46 eV (fitted),[S56] 3.1-3.3 eV (fitted),[S57] 4.0 eV (via comparison to V_2O_5 calculated densities of state and other experimental data),[S58] 4.00 eV (U = 4.00 eV, J = 0.68 eV, $U_{eff} \neq U$ - J),[S59] and 3.32 eV (U = 4.00 eV, J = 0.68 eV, $U_{eff} = U$ - J).[S60]

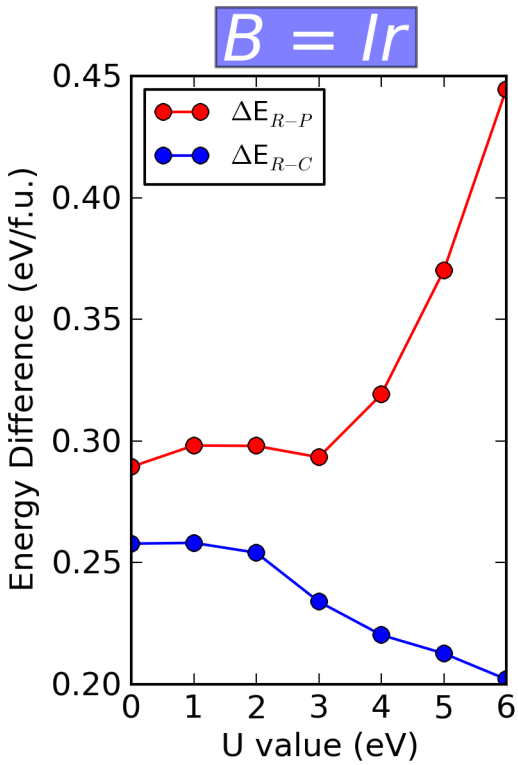

Figure S17

Given a U value of 3.0 eV for all species, the error ranges shown above could impact epitaxial stabilization candidate prediction for Anatase, Columbite, and Brookite polymorphs, which have corresponding formation energies of approximately +0.611, +0.119, and +0.113 eV/f.u. VO_2 with respect to Rutile. In contrast, applying a U value of 4.0 eV shows that the error ranges shown above could impact epitaxial stabilization candidate prediction for Columbite and Brookite polymorphs, which have corresponding formation energies of approximately +0.119 and +0.113 eV/f.u. VO_2 with respect to Rutile. Thus, first-principles calculation of U values is important for predicting relative energetics in the VO_2 system.

For the RuO_2 system depicted above, Rutile, Pyrite, and Columbite polymorphs are featured in formation energy calculations set with respect to Rutile. Given an epitaxial stabilization window of 0.1-0.2 eV and the errors mentioned previously, stabilization of the Columbite polymorph from a Rutile precursor is arguably possible in the RuO_2 system. For RuO_2, an appropriate selection of a U parameter for both Rutile and Columbite polymorphs

was not available within the literature to the knowledge of the authors of this manuscript, with some authors of previous work relating to RuO_2 bulk and shear moduli calculation[S61] contending that applying U to Ru cations is not necessary. Though past work suggests a U value of 6.73 eV for RuO_2 Rutile,[S34] the plot demonstrated above shows that RuO_2 Columbite would be predicted (via extrapolation) to be more favorable than RuO_2 Rutile, which is not consistent with past experimental and theoretical results.[S61] Furthermore, this U value was calculated using Ultrasoft rather than PAW pseudopotentials, thus possible discrepancies similar to those demonstrated in the article for TiO_2 systems could result from applying such a U value to the formation energetic calculations shown above. Nevertheless, a proposed U value of 2.9 eV was calculated for Ru impurities in Rb,[S62] which was successfully applied to RuO_2 in past work.[S63] Application of $U = 2.9$ eV to RuO_2 Rutile and Columbite would not contradict experimental expectations, thus this result is implemented in this analysis. Using $U = 2.9$ eV, the effective Rutile-Columbite formation energy in the RuO_2 system is +0.10 eV/f.u. RuO_2 with respect to Rutile, as is shown in the plot above. Given the epitaxial stabilization window and TiO_2 energetic error range shown above, first-principles calculation of U values is important for predicting relative energetics in the RuO_2 system.

For the IrO_2 system depicted above, Rutile, Pyrite, and Columbite polymorphs are featured in formation energy calculations set with respect to Rutile. Given a value of $U = 2.0$ eV resolved for IrO_2 Rutile via qualitative first-principles assessment[S64] and the error ranges specified above, candidacy for the epitaxial stabilization of either Pyrite or Columbite using a Rutile precursor is not feasible, as their respective formation energies are +0.26 and +0.29 eV eV/f.u. IrO_2 with respect to Rutile. Thus, first-principles calculation of U values is not necessarily important for predicting relative energetics in this IrO_2 system.

References

(S1) Dominik, C. *The Org-Mode 7 Reference Manual: Organize Your Life with GNU Emacs*; Network Theory: UK, 2010; with contributions by David O'Toole, Bastien

Guerry, Philip Rooke, Dan Davison, Eric Schulte, and Thomas Dye.

(S2) Python. `http://python.org/`.

(S3) Mehta, P.; Salvador, P. A.; Kitchin, J. R. Identifying Potential BO_2 Oxide Polymorphs for Epitaxial Growth Candidates. *ACS Appl. Mater. Interfaces* **2014**, *6*, 3630–3639.

(S4) Codd, E. F. *Relational completeness of data base sublanguages*; IBM Corporation, 1972.

(S5) Fagin, R. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Data. Sys.* **1977**, *2*, 262–278.

(S6) MySQL. `http://www.mysql.com/`.

(S7) sqlite. `https://sqlite.org/`.

(S8) PlantUML. `http://plantuml.sourceforge.net/`.

(S9) Curnan, M. T.; Kitchin, J. R. Effects of Concentration, Crystal Structure, Magnetism, and Electronic Structure Method on First-Principles Oxygen Vacancy Formation Energy Trends in Perovskites. *J. Phys. Chem. C* **2014**, *118*, 28776–28790.

(S10) Murnaghan, F. The compressibility of media under extreme pressures. *Proc. Natl. Acad. Sci. U. S. A.* **1944**, *30*, 244.

(S11) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865–3868.

(S12) Perdew, J. P.; Ruzsinszky, A.; Csonka, G. I.; Vydrov, O. A.; Scuseria, G. E.; Constantin, L. A.; Zhou, X.; Burke, K. Restoring the Density-Gradient Expansion for Exchange in Solids and Surfaces. *Phys. Rev. Lett.* **2008**, *100*, 136406.

(S13) Perdew, J. P.; Chevary, J. A.; Vosko, S. H.; Jackson, K. A.; Pederson, M. R.; Singh, D. J.; Fiolhais, C. Atoms, molecules, solids, and surfaces: Applications of

the generalized gradient approximation for exchange and correlation. *Phys. Rev. B* **1992**, *46*, 6671–6687.

(S14) Armiento, R.; Mattsson, A. E. Functional designed to include surface effects in self-consistent density functional theory. *Phys. Rev. B* **2005**, *72*, 085108.

(S15) Kresse, G.; Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **1996**, *54*, 11169–11186.

(S16) Hammouchi, M.; El Boudouti, E. H.; Nougaoui, A.; Djafari-Rouhani, B.; Lahlaouti, M. L. H.; Akjouj, A.; Dobrzynski, L. Acoustic waves in finite superlattices: Influence of buffer layers. *Phys. Rev. B* **1999**, *59*, 1999–2010.

(S17) Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I.; Corso, A. D.; de Gironcoli, S.; Fabris, S.; Fratesi, G.; Gebauer, R.; Gerstmann, U.; Gougoussis, C.; Kokalj, A.; Lazzeri, M.; Martin-Samos, L.; Marzari, N.; Mauri, F.; Mazzarello, R.; Paolini, S.; Pasquarello, A.; Paulatto, L.; Sbraccia, C.; Scandolo, S.; Sclauzero, G.; Seitsonen, A. P.; Smogunov, A.; Umari, P.; Wentzcovitch, R. M. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys.: Condens. Matter* **2009**, *21*, 395502.

(S18) Mozilla Firefox. `http://www.mozilla.org/en-US/firefox/new/`, Last visited April 20, 2014.

(S19) Muscat, J.; Swamy, V.; Harrison, N. M. First-principles calculations of the phase stability of TiO_2. *Phys. Rev. B* **2002**, *65*, 224112.

(S20) Liechtenstein, A. I.; Anisimov, V. I.; Zaanen, J. Density-functional theory and strong interactions: Orbital ordering in Mott-Hubbard insulators. *Phys. Rev. B* **1995**, *52*, R5467–R5470.

(S21) Zhou, F.; Cococcioni, M.; Marianetti, C. A.; Morgan, D.; Ceder, G. First-principles prediction of redox potentials in transition-metal compounds with LDA+U. *Phys. Rev. B* **2004**, *70*, 235121.

(S22) Swamy, V.; Wilson, N. C. First-Principles Calculations of the Pressure Stability and Elasticity of Dense TiO_2 Phases Using the B3LYP Hybrid Functional. *J. Phys. Chem. C* **2014**, *118*, 8617–8625.

(S23) Taggart, J. E.; Foord, E. E.; Rosenzweig, A.; Hanson, T. Scrutinyite, natural occurrences of $\alpha - PbO_2$ from Bingham, New Mexico, U.S.A., and Mapimi, Mexico. *Can. Min.* **1988**, *26*, 905–910.

(S24) Ma, X. G.; Liang, P.; Miao, L.; Bie, S. W.; Zhang, C. K.; Xu, L.; Jiang, J. J. Pressure-induced phase transition and elastic properties of TiO_2 polymorphs. *Phys. Status Solidi B* **2009**, *246*, 2132–2139.

(S25) Wu, X.; Holbig, E.; Steinle-Neumann, G. Structural stability of TiO_2 at high pressure in density-functional theory based calculations. *J. Phys.: Condens. Matter* **2010**, *22*, 295501.

(S26) Ohsaka, T.; Yamaoka, S.; Shimomura, O. Effect of hydrostatic pressure on the Raman spectrum of anatase (TiO_2). *Solid State Commun.* **1979**, *30*, 345 – 347.

(S27) Mammone, J.; Sharma, S.; Nicol, M. Raman study of rutile (TiO_2) at high pressures. *Solid State Commun.* **1980**, *34*, 799 – 802.

(S28) Arashi, H. Raman spectroscopic study of the pressure-induced phase transition in TiO_2. *J. Phys. Chem. Solids* **1992**, *53*, 355 – 359.

(S29) Lagarec, K.; Desgreniers, S. Raman study of single crystal anatase TiO_2 up to 70 GPa. *Solid State Commun.* **1995**, *94*, 519 – 524.

(S30) Olsen, J. S.; Gerward, L.; Jiang, J. On the rutile/$\alpha - PbO_2$-type phase boundary of TiO_2. *J. Phys. Chem. Solids* **1999**, *60*, 229 – 233.

(S31) Sekiya, T.; Ohta, S.; Kamei, S.; Hanakawa, M.; Kurita, S. Raman spectroscopy and phase transition of anatase TiO_2 under high pressure. *J. Phys. Chem. Solids* **2001**, *62*, 717 – 721.

(S32) Mattesini, M.; de Almeida, J. S.; Dubrovinsky, L.; Dubrovinskaia, N.; Johansson, B.; Ahuja, R. High-pressure and high-temperature synthesis of the cubic TiO_2 polymorph. *Phys. Rev. B* **2004**, *70*, 212101.

(S33) Finazzi, E.; Di Valentin, C.; Pacchioni, G.; Selloni, A. Excess electron states in reduced bulk anatase TiO_2: Comparison of standard GGA, GGA+U, and hybrid DFT calculations. *J. Chem. Phys.* **2008**, *129*, 154113–1:9.

(S34) Xu, Z.; Rossmeisl, J.; Kitchin, J. R. A Linear Response DFT+U Study of Trends in the Oxygen Evolution Activity of Transition Metal Rutile Dioxides. *J. Phys. Chem. C* **2015**, *119*, 4827–4833.

(S35) Tsuneda, T.; Hirao, K. Self-interaction corrections in density functional theory. *J. Chem. Phys.* **2014**, *140*.

(S36) Aroyo, M. I.; Perez-Mato, J. M.; Capillas, C.; Kroumova, E.; Ivantchev, S.; Madariaga, G.; Kirov, A.; Wondratschek, H. Bilbao Crystallographic Server: I. Databases and Crystallographic Computing Programs. *Z. Kristallogr.* **2006**, *221*, 15–27.

(S37) Cococcioni, M.; de Gironcoli, S. Linear response approach to the calculation of the effective interaction parameters in the LDA+U method. *Phys. Rev. B* **2005**, *71*, 035105.

(S38) Kulik, H. J.; Cococcioni, M.; Scherlis, D. A.; Marzari, N. Density Functional Theory

in Transition-Metal Chemistry: A Self-Consistent Hubbard U Approach. *Phys. Rev. Lett.* **2006**, *97*, 103001.

(S39) Kulik, H. J.; Marzari, N. Systematic study of first-row transition-metal diatomic molecules: A self-consistent DFT+U approach. *J. Chem. Phys.* **2010**, *133*, 114103–1:15.

(S40) VASP - FORUMS. `http://cms.mpi.univie.ac.at/vasp-forum/viewtopic.php?f=4&t=16400`, Last visited March 24, 2015.

(S41) Atomic Simulation Environment (ASE). `https://wiki.fysik.dtu.dk/ase/`, Last visited March 24, 2015.

(S42) Lefebvre, M.; Keeler, R.; Sobie, R.; White, J. Propagation of errors for matrix inversion. *Nucl. Instrum. Methods Phys. Res., Sect. A* **2000**, *451*, 520 – 528.

(S43) Ghaoui, L. E. Inversion error, condition number, and approximate inverses of uncertain matrices. *Lin. Alg. Appl.* **2002**, *343 - 344*, 171 – 193, Special Issue on Structured and Infinite Systems of Linear equations.

(S44) Kulik, H. J.; Marzari, N. A self-consistent Hubbard U density-functional theory approach to the addition-elimination reactions of hydrocarbons on bare FeO^+. *J. Chem. Phys.* **2008**, *129*, 134314–1:12.

(S45) Kulik, H. J.; Marzari, N. Accurate potential energy surfaces with a DFT+$U(R)$ approach. *J. Chem. Phys.* **2011**, *135*, 194105–1:9.

(S46) Arroyo-de Dompablo, M. E.; Morales-García, A.; Taravillo, M. DFT+U calculations of crystal lattice, electronic structure, and phase stability under pressure of TiO_2 polymorphs. *J. Chem. Phys.* **2011**, *135*, –.

(S47) Han, X.; Shao, G. Electronic Properties of Rutile TiO_2 with Nonmetal Dopants from First Principles. *J. Phys. Chem. C* **2011**, *115*, 8274–8282.

(S48) Islam, M. M.; Calatayud, M.; Pacchioni, G. Hydrogen Adsorption and Diffusion on the Anatase TiO_2(101) Surface: A First-Principles Investigation. *J. Phys. Chem. C* **2011**, *115*, 6809–6814.

(S49) Han, X.; Song, K.; Lu, L.; Deng, Q.; Xia, X.; Shao, G. Limitation and extrapolation correction of the GGA+ U formalism: a case study of Nb-doped anatase TiO_2. *J. Phys. Chem. C* **2013**, *1*, 3736–3746.

(S50) Meredig, B.; Thompson, A.; Hansen, H. A.; Wolverton, C.; van de Walle, A. Method for locating low-energy solutions within DFT $+ U$. *Phys. Rev. B* **2010**, *82*, 195128.

(S51) Heyd, J.; Scuseria, G. E.; Ernzerhof, M. Hybrid functionals based on a screened Coulomb potential. *J. Chem. Phys.* **2003**, *118*, 8207–8215.

(S52) Heyd, J.; Scuseria, G. E.; Ernzerhof, M. Erratum:.

(S53) Morgan, B. J.; Watson, G. W. Intrinsic n-type Defect Formation in TiO_2: A Comparison of Rutile and Anatase from GGA+U Calculations. *J. Phys. Chem. C* **2010**, *114*, 2321–2328.

(S54) Rao, C. N. R. Kinetics and Thermodynamics of the Crystal Structure Transformation of Spectroscopically Pure Anatase to Rutile. *Can. J. Chem.* **1961**, *39*, 498–500.

(S55) Gorbenko, O. Y.; Samoilenkov, S. V.; Graboy, I. E.; Kaul, A. R. Epitaxial Stabilization of Oxides in Thin Films. *Chem. Mater.* **2002**, *14*, 4026–4043.

(S56) Aykol, M.; Wolverton, C. Local environment dependent GGA$+U$ method for accurate thermochemistry of transition metal compounds. *Phys. Rev. B* **2014**, *90*, 115105.

(S57) Wang, L.; Maxisch, T.; Ceder, G. Oxidation energies of transition metal oxides within the GGA+U framework. *Phys. Rev. B* **2006**, *73*, 195107.

(S58) Scanlon, D. O.; Walsh, A.; Morgan, B. J.; Watson, G. W. An ab initio Study of Reduction of V2O5 through the Formation of Oxygen Vacancies and Li Intercalation. *J. Phys. Chem. C* **2008**, *112*, 9903–9911.

(S59) Biermann, S.; Poteryaev, A.; Lichtenstein, A. I.; Georges, A. Dynamical Singlets and Correlation-Assisted Peierls Transition in VO_2. *Phys. Rev. Lett.* **2005**, *94*, 026404.

(S60) Xiao, B.; Sun, J.; Ruzsinszky, A.; Perdew, J. P. Testing the Jacob's ladder of density functionals for electronic structure and magnetism of rutile VO_2. *Phys. Rev. B* **2014**, *90*, 085134.

(S61) Hugosson, H. W.; Grechnev, G. E.; Ahuja, R.; Helmersson, U.; Sa, L.; Eriksson, O. Stabilization of potential superhard RuO_2 phases: A theoretical study. *Phys. Rev. B* **2002**, *66*, 174111.

(S62) Solovyev, I. V.; Dederichs, P. H. *Ab initio* calculations of Coulomb U parameters for transition-metal impurities. *Phys. Rev. B* **1994**, *49*, 6736–6740.

(S63) Dong, H.; Zhang, L.; Zhou, X. Theoretical investigation on RuO_2 nanoclusters adsorbed on TiO_2 rutile (110) and anatase (101) surfaces. *Theor. Chem. Acc.* **2014**, *133*.

(S64) Panda, S. K.; Bhowal, S.; Delin, A.; Eriksson, O.; Dasgupta, I. Effect of spin orbit coupling and Hubbard U on the electronic structure of IrO_2. *Phys. Rev. B* **2014**, *89*, 155102.