

Universal Structure Predictor:

Evolutionary Xtallography

A.R. Oganov, S.V. Lepeshkin, S.V. Bakanov, D.V. Rybkovskiy, D.O. Poletaev, Z. Momenzadeh Abardeh, Carlo Nervi, V.N. Valmispild, C.W. Glass, A.O. Lyakhov, Q. Zhu, Z. Allahyari, G.-R. Qian, E. Trukhan, E.E. Vaneeva

with contributions from

H.T. Stokes, P. Bushlanov, P. Graf, V. Stevanovic, F. Therrien, A.I. Samtsevich, V.N. Baturin, E. Mazhnik, M.S. Rakitin, D. Dong *and others*

MANUAL

Version 26.5, November 15, 2026.

© A.R. Oganov, S.V. Lepeshkin and S.V.Bakanov

https://uspex-team.org



COMPUTATIONAL MATERIALS DISCOVERY LABORATORY

Contents

1 Features, aims and history of USPEX **4**

1.1 Overview 4

1.2 Features of USPEX 6

1.3 Key USPEX papers 8

1.4 Version history 10

2 Getting started **15**

2.1 How to obtain USPEX 15

2.2 Necessary citations 15

2.3 Supported platforms 15

2.4 Codes that can work with USPEX 15

2.5 How to install USPEX 16

2.6 How to run USPEX 16

3 Input options. The `INPUT.txt` file **17**

3.1 Minimal `INPUT.txt` example 17

3.2 Calculation folder 17

3.3 Relaxation with MatterSim 18

3.4 Type of calculation and system specification 18

3.5 Population 20

3.6 Survival of the fittest and selection 22

3.7 Structure generation 22

3.8 Variation Operators 24

3.8.1 Operator fractions 25

3.8.2 Mutation settings 27

3.8.3 Heredity and permutation settings 27

3.8.4 Transmutation 29

3.9 Constraints 30

3.10 Cell 32

3.11 Details of *ab initio* calculations 33

3.12 Fingerprint settings 36

3.13 Space group determination . . . 37
3.14 Variable-composition structure prediction . . . 37
3.15 Seldom used parameters . . . 40
3.15.1 Relaxation time limits and small random perturbations . . . 43
3.15.2 CPU settings . . . 43

4 Main output files **45**
4.1 Run summary and logs . . . 45
4.2 Structure information files . . . 45
4.3 Structures in POSCAR format . . . 46
4.4 Energies and relaxation diagnostics . . . 46
4.5 Variable-composition output . . . 46
4.6 Nanocluster output . . . 47
4.7 How to read structure information files . . . 47
4.8 How to analyze output . . . 48
4.9 Tool for vizualization . . . 49

5 Additional input for special cases **51**
5.1 Single-block calculations . . . 51
5.2 Molecular crystals . . . 51
5.2.1 Molecular crystals, `calculationType`=310/311 . . . 51
5.2.2 Additional inputs for classical forcefields . . . 54
5.2.3 How to prepare the `MOL` files . . . 55
5.3 Molecules and nanoclusters . . . 56
5.3.1 Composition input for each mode . . . 56
5.3.2 Atomic cluster generation . . . 56
5.4 ASE models and optimization settings . . . 58
5.5 User-defined programs . . . 59
5.5.1 External structure generators . . . 59
5.5.2 Structure checks and duplicate detection . . . 63
5.5.3 User-defined search property . . . 64
5.5.4 99 — Using a user-defined external code . . . 65

6 Online utilities **67**

6.1 USPEX-team codes and utilities 67

6.2 Structure characterization 68

6.3 Properties calculations 68

7 Appendices **69**

7.1 Sample `INPUT.txt` files 69

7.1.1 Fixed-composition USPEX calculation (`calculationType`=300): . . 69

7.1.2 Variable-composition USPEX calculation (`calculationType`=301): 71

7.1.3 MatterSim input for a fixed-composition crystal 73

7.1.4 Nanocluster inputs 74

7.1.5 Inputs for molecular crystal structure prediction 75

7.2 List of space groups 76

7.3 List of layer groups 77

7.4 List of plane groups 78

7.5 List of point groups 79

7.6 Table of univalent covalent radii used in USPEX 80

7.7 Table of default chemical `valences` used in USPEX 81

7.8 Table of default `goodBonds` used in USPEX 82

7.9 Examples 83

Bibliography **85**

1 Features, aims and history of USPEX

1.1 Overview

USPEX stands for *Universal Structure Predictor: Evolutionary Xtallography*... and in Russian "uspekh" means "success", which is appropriate given the high success rate and many useful results produced by this method! The USPEX code possesses many unique capabilities for computational materials discovery. Here is a list of features: ...

From the beginning in 2004, non-empirical crystal structure prediction was the main aim of the USPEX project. In addition to this, USPEX also allows one to predict a large set of robust metastable structures and perform several types of simulations using various degrees of prior knowledge. Starting from 2010, our code explosively expanded to other types of problems, and from 2012 includes many complementary methods.

The problem of crystal structure prediction is very old and does, in fact, constitute the central problem of theoretical crystal chemistry. In 1988 John Maddox[1] wrote that:

> "One of the continuing scandals in the physical sciences is that it remains in general impossible to predict the structure of even the simplest crystalline solids from a knowledge of their chemical composition... Solids such as crystalline water (ice) are still thought to lie beyond mortals' ken".

It is immediately clear that the problem at hand is that of global optimization, *i.e.*, finding the global minimum of the free energy of the crystal (per mole) with respect to variations of the structure. To get some feeling of the number of possible structures, let us consider a simplified case of a fixed cubic cell with volume V, within which one has to position N identical atoms. For further simplification let us assume that atoms can only take discrete positions on the nodes of a grid with resolution δ. This discretization makes the number of combinations of atomic coordinates C finite:

$$C = \frac{1}{(V/\delta^3)} \frac{(V/\delta^3)!}{[(V/\delta^3) - N]!N!} \tag{1}$$

If δ is chosen to be a significant fraction of the characteristic bond length (e.g., $\delta = 1$ Å), the number of combinations given by eq. 1 would be a reasonable estimate of the number of local minima of the free energy. If there are more than one type of atoms, the number of different structures significantly increases. Assuming a typical atomic volume $\sim$10 Å^3, and taking into account Stirling's formula ($n! \approx \sqrt{2\pi n}(n/e)^n$), the number of possible structures for an element A (compound AB) is 10^{11} (10^{14}) for a system with 10 atoms in the unit cell, 10^{25} (10^{30}) for a system with 20 atoms in the cell, and 10^{39} (10^{47}) for a system with 30 atoms in the unit cell. These numbers are enormous and practically impossible to deal with even for small systems with a total number of atoms $N \sim 10$.

Even worse, complexity increases exponentially with N. It is clear then, that point-by-point exploration of the free energy surface going through all possible structures is not feasible, except for the simplest systems with ∼1-5 atoms in the unit cell.

USPEX[2;3] employs an evolutionary algorithm devised by A.R. Oganov and C.W. Glass, with major subsequent contributions by A.O. Lyakhov, Q. Zhu, G.R. Qian, P. Bushlanov, Z. Allahyari, S. Lepeshkin and A. Samtsevich. Its efficiency draws from the carefully designed variation operators, while its reliability is largely due to the use of state-of-the-art *ab initio* simulations inside the evolutionary algorithm. The strength of evolutionary simulations is that they do not require any system-specific knowledge (except the chemical composition) and are self-improving, i.e. in subsequent generations increasingly good structures are found and used to generate new structures. This allows a "zooming in" on promising regions of the energy (or property) landscape (Fig. 1). Furthermore, by carefully designing variation operators, it is very easy to incorporate additional features into an evolutionary algorithm.

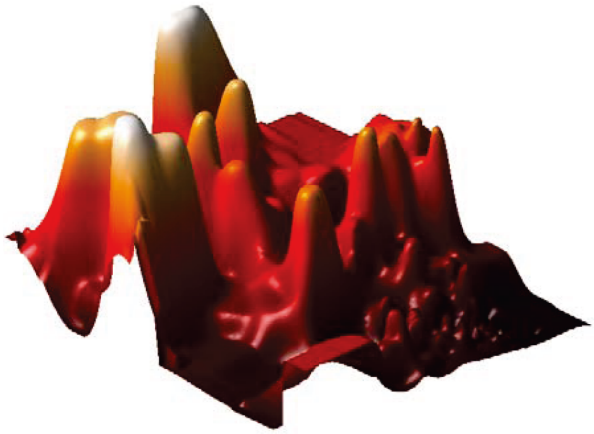


Figure 1: **2D projection of the reduced landscape of Au_8Pd_4, showing clustering of low-energy structures in one region.** The landscape was produced using the method of Oganov & Valle (2009).

A major motivation for the development of USPEX was the discovery of the post-perovskite phase of $MgSiO_3$ (Fig. 2), which was made in 2004[4;5] and has significantly changed models of the Earth's internal structure. In mid-2005 we had the first working version of USPEX. By September 2010, when USPEX was publicly released, the user community numbered nearly 200, over 800 users in May 2012, over 2100 in December 2014, and over 4500 in December 2018.

The popularity of USPEX is due to its extremely high efficiency and reliability. This was shown in the First Blind Test for Inorganic Crystal Structure Prediction[6], where USPEX outperformed the other methods it was tested against (simulated annealing and random sampling). Random sampling is the simplest, but much weaker and computationally the most expensive strategy. Even for small systems, such as GaAs with 8 atoms/cell, these advantages are large (random sampling requires on average 500 structure relaxations to

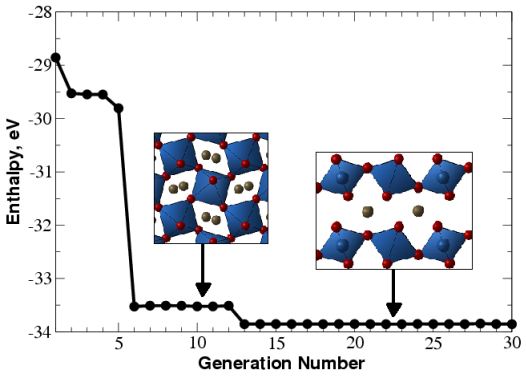


Figure 2: **Prediction of the crystal structure of $MgSiO_3$ at 120 GPa (20 atoms/cell).** Enthalpy of the best structure as a function of generation is shown. Between the 6^{th} and 12^{th} generations the best structure is perovskite, but at the 13^{th} generation the global minimum (post-perovskite) is found. This simulation was performed in 2005 using one of the first versions of USPEX combined with *ab initio* calculations. It used no experimental information and illustrates that USPEX can find both the stable and low-energy metastable structures in a single simulation. Each generation contains 30 structures. This figure illustrates the slowest of ∼10 calculations performed by the very first version of USPEX — and even that was pretty fast!

find the ground state in this case, while USPEX finds it after only ∼ 30 relaxations! (Fig. 3)). Due to the exponential scaling of the complexity of structure search (eq. 1), the advantages of USPEX increase exponentially with system size. For instance, 2 out of 3 structures of SiH_4 predicted by random sampling to be stable[7], turned out to be unstable[8]; and similarly random sampling predictions were shown[9] to be incorrect for nitrogen[10] and for SnH_4 (compare predictions[11] of USPEX and of random sampling[12]).

For larger systems, random sampling tends to produce almost exclusively disordered structures with nearly identical energies, which decreases the success rate to practically zero, as shown in the example of $MgSiO_3$ post-perovskite with 40 atoms/supercell — random sampling fails to find the correct structure even after 120,000 relaxations, whereas USPEX finds it after several hundred relaxations (Fig. 4).

For detailed investigations of phase transition mechanisms, additional methods were implemented in earlier USPEX versions, including the variable-cell NEB method[13] and transition path method[14] in the version[15].

1.2 Features of USPEX

- Prediction of the stable and metastable structures knowing only the chemical composition. Simultaneous searches for stable compositions and structures are also possible.
- Incorporation of partial structural information is possible:
  - constraining search to fixed experimental cell parameters, or fixed cell shape,

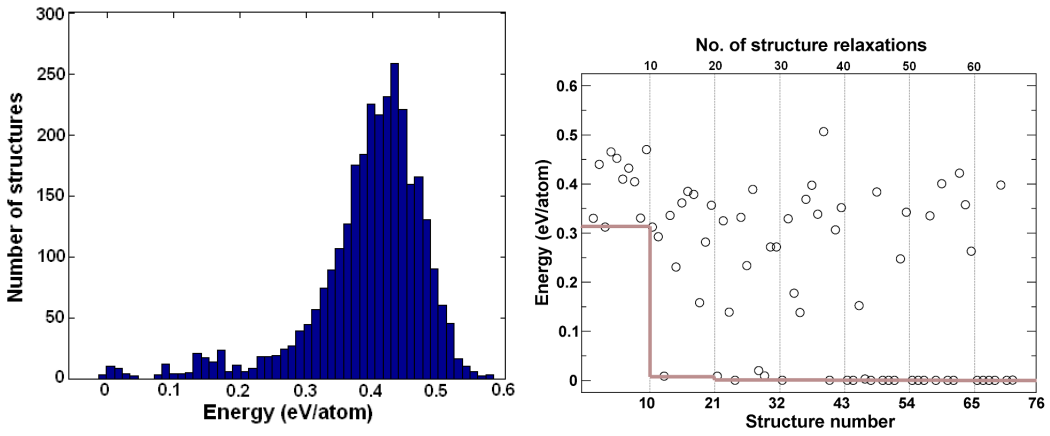


Figure 3: **Structure prediction for GaAs.** a) Energy distribution for relaxed random structures, b) progress of an evolutionary simulation (thin vertical lines show generations of structures, and the grey line shows the lowest energy as a function of generation). All energies are relative to the ground-state structure. The evolutionary simulation used 10 structures per generation. In addition, the lowest-energy structure of the previous generation survived into the next generation.

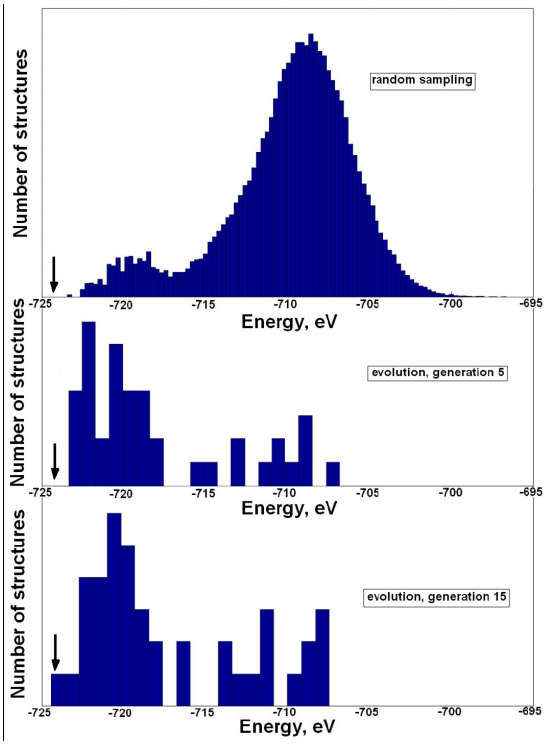


Figure 4: **Sampling of the energy surface: comparison of random sampling and USPEX for a 40-atom cell of $MgSiO_3$ with cell parameters of post-perovskite.** Energies of locally optimized structures are shown. For random sampling, 1.2×10^5 structures were generated (none of which corresponded to the ground state). For USPEX search, each generation included 40 structures and the ground-state structure was found within 15 generations. The energy of the ground-state structure is indicated by the arrow. This picture shows that "learning" incorporated in evolutionary search drives the simulation towards lower-energy structures.

or fixed cell volume (Subsection 3.10);

- starting structure search from known or hypothetical structures (Subsection 3.2);
- assembling crystal structures from predefined molecules, including flexible molecules (available in USPEX 26.0).

- Efficient constraint techniques, which eliminate unphysical and redundant regions of the search space. Cell reduction technique (Oganov & Glass, 2008).
- Niching using fingerprint functions (Oganov & Valle, 2009; Lyakhov, Oganov, Valle, 2010). Subsection 3.12 for details.
- Initialization using fully random approach, or using space groups and cell splitting techniques (Lyakhov, Oganov, Valle, 2010). Use of powerful topological structure generator (Bushlanov, Blatov, Oganov, 2018).
- On-the-flight analysis of results — determination of space groups (Subsection 3.13), calculation of the hardness, order parameters, *etc.*
- Prediction of isolated atomic clusters with fixed or variable composition. See Section 3.4.
- Powerful visualization and analysis techniques implemented in the STMng code (by M. Valle), fully interfaced with USPEX (Subsection 4.9).
- USPEX interfaces with external relaxation codes and supports MatterSim and DPA4 through `abinitioCode`=0. See the list in Subsection 2.4.
- Submission of jobs from local workstation to remote clusters and supercomputers is possible. USPEX 26.0 and subsequent versions can also be used on Windows PCs. See Section 3.11 for details.
- Options for structure prediction using the USPEX algorithm (default) and random sampling.
- Single-objective optimization and Pareto optimization of two or three objectives (Section 3.4).

1.3 Key USPEX papers

1. Oganov A.R., Glass C.W. (2006). Crystal structure prediction using evolutionary algorithms: principles and applications. *J. Chem. Phys.*, **124**, 244704.
2. Oganov A.R., Stokes H., Valle M. (2011). How evolutionary crystal structure prediction works — and why. *Acc. Chem. Res.*, **44**, 227–237.

3. Lyakhov A.O., Oganov A.R., Stokes H., Zhu Q. (2013). New developments in evolutionary structure prediction algorithm USPEX. *Comp. Phys. Comm.*, **184**, 1172–1182.

4. Zhu Q., Oganov A.R., Glass C.W., Stokes H. (2012). Constrained evolutionary algorithm for structure prediction of molecular crystals: methodology and applications. *Acta Cryst. B*, **68**, 215–226.

5. Zhu Q., Li L., Oganov A.R., Allen P.B. (2013). Evolutionary method for predicting surface reconstructions with variable stoichiometry. *Phys. Rev. B*, **87**, 195317.

6. Zhu, Q., Sharma V., Oganov A.R., Ramprasad R. (2014). Predicting polymeric crystal structures by evolutionary algorithms. *J. Chem. Phys.*, **141**, 154102.

7. Lyakhov A.O., Oganov A.R., Valle M. (2010). Crystal structure prediction using evolutionary approach. In: Modern methods of crystal structure prediction (ed: A.R. Oganov), Berlin: Wiley-VCH.

8. Oganov A.R., Ma Y., Lyakhov A.O., Valle M., Gatti C. (2010). Evolutionary crystal structure prediction as a method for the discovery of minerals and materials. *Rev. Mineral. Geochem.*, **71**, 271–298.

9. Duan, D., Liu, Y., Tian, F., Li, D., Huang, X., Zhao, Z., Yu, H., Liu, B., Tian, W., Cui, T. (2014). Pressure-induced metallization of dense $(H_2S)_2H_2$ with high-T_c superconductivity. *Sci. Rep.*, **4**, 6968.

10. Bushlanov, P. V., Blatov, V. A., Oganov, A. R. (2019). Topology-based crystal structure generator. Computer Physics Communications, 236, 1-7.

11. Lepeshkin S.V., Baturin V.S., Uspenskii Yu.A., Oganov A.R. (2019). Method for simultaneous prediction of atomic structure of nanoclusters in a wide area of compositions. J. Phys. Chem. Lett. 10, 102-106.

12. Lepeshkin S.V., Baturin V.S., Uspenskii Yu.A., Oganov A.R. (2019). Method for simultaneous prediction of atomic structure of nanoclusters in a wide area of compositions. J. Phys. Chem. Lett. 10, 102-106.

13. Allahyari, Z., Oganov, A. R. (2018). Multi-objective optimization as a tool for material design. Handbook of Materials Modeling, 1-15.

1.4 Version history

v.1 — Evolutionary algorithm without local optimization. Real-space representation, interface with VASP. Experimental version. October 2004.

v.2 — CMA-ES implementation (CMA-ES is a powerful global optimization method developed by N. Hansen). Experimental version. January 2005.

v.3 — Evolutionary algorithm with local optimization.

v.3.1 — Working versions, sequential. Major basic developments.

3.1.4-3.1.5 — First production version. Based largely on heredity with slice-shifting and with minimum-parent contribution (hard-coded to be 0.25). May 2005.

3.1.8 — Adaptive k-point grids. 15/10/2005.

3.1.12 — Production version based on v.3.1.11, variable slice-shift mutation. 11/11/2005.

3.1.13 — Adaptive scaling volume. 29/11/2005.

3.1.14 — Basic seed technique. 29/11/2005 (debugged 6/12/2005).

v.3.2 — Massively parallel version.

v.4 — Unified parallel/sequential version.

4.1.1 — Lattice mutation. 20/12/2005 (debugged 10/01/2006).

4.2.1 — Interfaced with SIESTA. Initial population size allowed to differ from the running population size. 24/01/2006 (debugged 20/04/2006).

4.2.3 — Relaxation of best structures made optional. Version with fully debugged parallelism. 25/04/2006.

4.4.1 — Interfaced with GULP. 08/05/2006.

v.5 — Completely rewritten and debugged version, clear modular structure of the code.

5.1.1 — Atom-specific permutation, code interoperability, on-the-fly reading of parameters from `INPUT_EA.txt`. 20/12/2006.

5.2.1 — SIESTA-interface for Z-matrix, rotational mutation operator. 01/03/2007.

v.6 — Production version.

6.1.3 — To efficiently fulfill hard constraints for large systems, an optimizer was implemented within USPEX. 07/06/2007.

6.2 — Development version.

6.3.1–6.3.2 — Introduced angular constraints for cell diagonals. Completely rewritten remote submission. Improved input format. Further extended standard tests. 07/12/2007.

6.3.3 — X-com grid interface (with participation of S. Tikhonov and S. Sobolev). 05/03/2008.

6.4.1 — Fingerprint functions for niching. 07/04/2008.

6.4.4 — Space group recognition. Fast fingerprints (from tables). 05/05/2008.

6.5.1 — Split-cell method for large systems. Easy remote submission. Variable number of best structures (clustering). 16/07/2008.

6.6.1 — A very robust version — improved fingerprint and split-cell implementations. 13/08/2008.

6.6.3 — Heredity with multiple parents implemented. 01/10/2008.

6.6.4 — Added a threshold for parents participating in heredity (niching). 03/10/2008.

6.6.6 — First implementation of multicomponent fingerprints. 04/12/2008.

6.6.7, 6.7.1 and 6.7.2 — Implemented quasi-entropy to measure the diversity of the population. 10/12/2008.

v.7 — Production version, written to include variable composition.

7.1.1–7.1.7 — Series of improved versions. Version 7.1.7 has been distributed to ∼200 users. Variable composition partly coded, most known bugs fixed, improved tricks based on energy landscapes. Improved cell splitting, implemented pseudo-subcells. Implemented multicomponent fingerprints (much more sensitive to the structure than one-component fingerprints). 28/04/2009 (version finalized 28/05/2009).

7.2.5 — First fully functional version of the variable-composition method. Introduced transmutation operator and compositional entropy. 06/09/2009.

7.2.7 — Thoroughly debugged, improved seeding, introduced perturbations within structure relaxation. 25/09/2009, further improved in versions 7.2.8/9.

7.3.0 — Full fingerprint support in the variable-composition code, including niching. "Fair" algorithm for producing the first generation of compositions. 22/10/2009.

7.4.1 — Introduced coordinate mutation based on local order[16]. Heredity and transmutation are also biased by local order. Introduced computation of the hardness and new types of optimization by hardness and density. 04/01/2010.

7.4.2 — Implementation of multiple-parents heredity biased by local order. 15/01/2010.

7.4.3 — Implementation of new types of optimization (to maximize structural order and diversity of the population). Implemented antiseeds, eliminated parameters `volTimeConst`, `volBestHowMany`. 24/01/2010.

v.8 — Production version, written to include new types of optimization.

8.1.1–8.2.8 — Development versions. Local order and coordinate mutation operator. Softmutation operator. Calculation and optimization of the hardness. Optimization of the dielectric susceptibility. Prediction of the structure of nanoparticles and surfaces. Implementation of point groups. Greatly improved overall performance. Option to perform PSO simulations (not recommended for applications, due to PSO's inferior efficiency — so use only for testing purposes). Parameter `goodBonds` transformed into a matrix and used for building nanoparticles. 22/09/2010.

8.3.1 — Optimization of dielectric constants, cleaned-up input. 08/10/2010.

8.3.2 — For clusters, introduced a check on connectivity (extremely useful), `dynamicalBestHM=2` option improved, as well as mechanism for producing purely softmutated generations. Improved fingerprints for clusters. Interface to Quantum Espresso and CP2K codes. 11/10/2010.

8.4 — Improved antiseed functionalities and several improvements for nanoparticles. Development branches for surface reconstructions, pseudo-metadynamics, molecular crystals.

8.5.0 — Initialization of the first random generation using the space group code of H. Stokes added. New formulation of metadynamics implemented and finalized, for now in a separate code. Several debugs for varcomp, antiseeds, nanoparticles, computation of hardness. 18/03/2011.

8.5.1 — Space group initialization implemented for cases of fixed unit cell, variable composition, and subcells. 20/04/2011.

8.6.0 — Added space group determination program from H. Stokes. Merger with the updated code for molecular crystals (including space group initialization). Fixed a bug for SIESTA (thanks to D. Skachkov). 06/05/2011.

8.6.1–8.7.2 — Improved symmetric initialization for the case of a fixed cell. Implemented optimization of dielectric constants (using GULP and VASP), band gap (using VASP), and DOS at the Fermi level (VASP). Graphical output enabled. Improved softmutation (by using better criteria for mode and directional degeneracies) and heredity (by using energy-order correlation coefficient and cosine formula for the number of trial slabs) operators. Most variables now have default values, which enables the use of very short input files. Shortened and improved the format of log-files. 13/11/2011.

8.7.5 — Graphical output now includes many extra figures. Added utility to extract all structures close to convex hull for easier post-processing. 21/03/2012.

v.9 — Production version, made more user-friendly and written to include new types of functionality and to set the new standard in the field.

9.0.0 — Evolutionary metadynamics and VCNEB codes added to USPEX package, added tensor version of metadynamics, added additional figures and post-processing tools, cleaned the code output. A few parameters removed from the input. Improved softmutation. April 2012.

9.1.0 — Release version. Cleaned up, documented. The user community is >800 people. Released 28/05/2012.

9.2.0 — Working GEM. Constant development of the GEM code. Space group determination tolerance is now an input parameter. Improved default for number of permutations. July-August 2012.

9.2.1–9.2.3 — Improved GEM, more diverse populations and supercell sizes, improved mode selection. September-October 2012.

9.2.4–9.2.6 — (9.2.4 is a release version). Intelligent defaults for most input parameters. Improved symmetric initialization for clusters. Order-enhanced heredity for nanoparticles. New parameter to tune the tolerance for the space group determination. New property (quasientropy) can be optimized. Fully integrated VCNEB code. November-December 2012.

9.2.7. — Release version. Enabled optimization of order for alloys, without structure relaxation (for easy creation of quasirandom structures, based on the more general definition than the so-called "special quasi-random structures"). Symmetry generation was improved (particularly important for fixed-cell calculations). For fixed-cell calculations, one can now specify the cell parameters, not only in the form of a 3×3 matrix, but also as a row of six values (three lengths in Angstroms and three angles in degrees). For the maximum number of permutation swaps (parameter `howManySwaps`), we have introduced an intelligent default. Added new tests, and cleaned and reran the old ones. Added interface to CASTEP (thanks to Z. Raza, X. Dong and AL). User community 1160 people. 30/12/2012.

9.3.0–9.3.3 — Fixed a bug in generation of random symmetric structures (this bug appeared in 9.2.7). Significantly simplified input and output. Created file `OUTPUT.txt` with the most important information. Enabled split-cell trick for molecular crystals. Improved variable-composition calculations by allowing one to specify initial compositions. Added interface to CASTEP and LAMMPS. Added new test cases. 20/03/2013.

9.3.4 — Release version, cleaned up. 25/03/2013.

9.3.5 — Added code for prediction of 2D-crystals. 19/04/2013.

9.3.6 — Incorporated plane groups for 2D-crystals. 29/04/2013.

9.3.8 — Incorporated plane groups for 1D-polymer crystals, improved variables of stoichiometry for surfaces. 19/06/2013.

9.3.9 — Released version. Significantly improved version, improved user-friendliness, new functionalities (2D-crystals, GEM) made more robust, improvements in the variable-composition algorithm (and enabled support for single-block calculations, *i.e.* fixed-composition searches with variable number of atoms in the cell), fully functional surface calculations, new optimization types (can optimize band gaps, dielectric constants, and newly invented figure of merit of dielectric materials). Interfaces with LAMMPS and ATK are documented in new test cases. Continuously updated with minor debugs (last debug 10/02/2014). 19/07/2013.

9.4.1 — A major upgrade, greatly improved user-friendliness (automatic estimate of volumes and of percentages of variation operators for each case), new functionalities (optimization of elastic properties and Chen's model of hardness, prediction of polymeric structures, anti-compositions, automatic analysis of statistics, improved seed technique), first release of GEM (generalized evolutionary metadynamics), provided a set of real-life examples of USPEX calculations, test cases, documentation. More than 2100 users. Released 30/12/2014.

9.4.2 — Release version, compatible with Octave 3.4. Convex hull code rewritten. Interface with MOPAC implemented. Default values for `goodBonds`, `valences`, `IonDistances` enabled. More robust for ternary, quaternary, and more complex variable-composition searches. Robust TPS implementation (only for developers, will be available for users soon). More than 2200 users. Released 21/03/2015.

9.4.3 — Release version. It includes fixing a number of bugs (which should slightly speed up performance), interface with MOPAC, improved documentation. Released 10/08/2015.

9.4.4 — Release version. It includes fix for space group determination and other problems reported by users, improved documentation and examples, full Octave 3.4 compatibility and partial Octave 3.6/3.8/4.0 support. This version should be nearly bug-free and is a milestone towards a very major upgrade, which will be made available in version 10. Released 05/10/2015.

10.1 - Experimental version, released as a virtual machine on 01/08/2018.

10.2 - Release version. Distributed as a compiled code, so users no longer need to have Matlab. Has no known bugs and contains a very large number of new features and improvements. Random topological structure generator and automatic parameter control greatly speed up calculations. Prediction of (collinear) magnetic materials is enabled. Many new types of fitness implemented: magnetization, birefringence, thermoelectric figure of merit ZT, fracture toughness. Fitness can be minimized or maximized, and we can input mathematical expressions as fitness. Pareto optimization of several fitnesses is enabled. USPEX has been interfaced with

Gaussian, MOPAC, DFTB, ORCA, FHI-aims, ABINIT. Symmetry determination is now done using SPGLIB. Symmetrization can also be optionally performed during calculation of physical properties, making it cheaper and more robust. 80 layer symmetry groups are utilized for generating initial population of 2D-structures. Variable-composition prediction of 2D-crystals is enabled. For surface structure prediction, we have enabled all possible surface supercells and output the surface phase diagram. Variable-cell NEB method has been greatly improved (made a few times faster). Utility "pmpaths" of V. Stevanovic has been added to predict the likeliest phase transition mechanisms (which can then be directly input into the VCNEB code). Released 19/01/2019.

10.3 - Similar to version 10.2, with some critical fixed-bugs in job submission, and some minor fixed-bugs fixes.

10.4 - Several improvements, bug fixes, new features. Local and remote submission files are improved. VCNEB and PSO codes are improved. Interfaces to QE, DMACRYS, VASP, and DFTB+ are improved. Interfaces to Abinit and CRYSTAL codes are added. Half-metallicity fitness is added. Added Mazhnik-Oganov model for calculation of hardness and fracture toughness. All python scripts are translated to python3 - no more python2 needed for USPEX.

10.5 - minor bug fixes. Added optimization of more than one quantity expressed by formula. Improved interface with DFTB+ and GULP5.2. Added "USPEX -u" feature, which allows one to update the package to the latest version without downloading the entire USPEX distribution. Enabled machine learning calculation of the elastic moduli (their optimization is available as optType=1201-1207), using graph convolutional neural network, and added Example35 to show how to use this feature.

10.6 - added user-defined fitness properties, allowing up to two arbitrary properties to be optimized together with standard objectives. Added an interface for arbitrary external codes through `abinitioCode`=99 and the `commandExecutable` block. Added importing candidate structures from the DAICS database through `fracDB` for selected atomic calculation types. Website: Added the POSCAR2CIF+XRD utility and updated output and examples.

25.0 (October 2025) — Rewrote the code and added Linux and Windows builds with fixed-composition, single-block, and variable-composition crystal structure prediction.

26.0 (June 2026) — Added molecular crystal structure prediction, parallel random structure generation, anti-compositions, and more detailed `OUTPUT.txt` summaries. Improved compatibility with the 10.5/10.6 heredity operators.

26.5

- **Molecular crystals.** Version 26.5 improves support for molecular crystal structure prediction (Section 5.2).
- **Molecules and nanoclusters.** See Section 5.3 for input examples for modes `000`, `001`, and `002`.
- Added 11 interfaces for external programs. See Sections 5.5.1, 5.5.2, 5.5.3, and 5.5.4.

2 Getting started

2.1 How to obtain USPEX

To obtain USPEX, download the binary package from the official USPEX website:

https://uspex-team.org

Use the executable appropriate for your operating system:

- `uspex` for Linux
- `uspex.exe` for Windows

Run the executable in the directory containing your calculation input files.

2.2 Necessary citations

Whenever using USPEX, in all publications and reports you must cite the original papers, for example, in the following way:

> "Crystal structure prediction was performed using the USPEX code[2;17;18], based on an evolutionary algorithm developed by Oganov, Glass, Lyakhov and Zhu and featuring local optimization, real-space representation and flexible physically motivated variation operators".

2.3 Supported platforms

USPEX runs on Windows and Linux, on a local machine or a remote cluster.

2.4 Codes that can work with USPEX

USPEX uses the following codes for structure relaxation and energy or property evaluation.

- `0` — MatterSim[19] (default) or DPA4. Select the model with `potential`.
- `1` — VASP[20;21] — `https://www.vasp.at/`
- `2` — ORCA[22] — `https://orcaforum.cec.mpg.de`
- `3` — GULP[23;24] — `http://nanochemistry.curtin.edu.au/gulp/`

- `13` — MOPAC[25] — `https://openmopac.net/`
- `-103` — GULP with space-group preservation[23;24].
- `20` — ASE interface[26] — `https://ase-lib.org`
- `99` — user-defined external relaxation using the file protocol in Section 3.11.

Interfaces to several more codes will appear very soon.

2.5 How to install USPEX

Extract the binary package for Linux or Windows. The package includes MatterSim and DPA4 for structure relaxation, and the PyXtal random symmetric structure generator[27;28;29].

2.6 How to run USPEX

Prepare `INPUT.txt` using Section 3. Choose the code for structure relaxation from Section 2.4. Place any required input files in `Specific/`, for example the VASP `INCAR` files and `POTCAR` files for the elements in the calculation.

For scheduler runs, prepare the job scripts for the local or remote scheduler. For direct runs, see `whichCluster` and `commandExecutable` in Section 3.11.

Run USPEX directly in the calculation folder:

```
./uspex
```

or, on Windows:

```
uspex.exe
```

`uspex.log` records progress and errors. See Section 4 for the files in the results directory.

3 Input options. The INPUT.txt file

See Appendix 7.9 for a list of input examples.

3.1 Minimal INPUT.txt example

A compact fixed-composition crystal prediction input can look as follows:

```
USPEX : calculationMethod

300 : calculationType

% atomType
Mg Si O
% EndAtomType

% numSpecies
4 4 12
% EndNumSpecies

% abinitioCode
0 0 0 0
% ENDabinit
```

This input specifies prediction of Mg–Si–O crystal structures at fixed composition with 4 MatterSim relaxation steps. USPEX stores the output in the results folder.

3.2 Calculation folder

Run USPEX from the directory where the calculation is performed. The calculation directory contains the following input files and folders:

- `INPUT.txt` — the main input file for the USPEX search.
- `Specific/` — input files for external relaxation codes.
- `Specific/INCAR_1`, `Specific/INCAR_2`, ... — VASP relaxation stages.
- `Specific/POTCAR_*` — VASP pseudopotentials for all elements.
- `Specific/goptions_*`, `Specific/ginput_*` — GULP relaxation templates.
- `Seeds/POSCARS_generation` — seed structures in VASP POSCAR format for the selected generation.

- `Seeds/compositions` and `Seeds/Anti-compositions` — composition controls for variable-composition or single-block calculations. `Seeds/Anti-compositions` can explicitly forbid selected compositions, including specific unary, binary, ternary, ratio-based, or exact compositions.

See Section 4 for the contents of the results folder.

3.3 Relaxation with MatterSim

Set the relaxation codes in the `abinitioCode` block to `0` and select `mattersim : potential`. See Appendix 7.1.3 for a complete `INPUT.txt` and local execution settings.

3.4 Type of calculation and system specification

▷ *variable* `calculationMethod`

Meaning: Selects the algorithm for structure prediction. Currently, only the evolutionary algorithm is enabled. More algorithms will be re-added soon.

Default: `USPEX`

Format:

```
USPEX : calculationMethod
```

▷ *variable* `calculationType`

Meaning: Specifies dimensionality, molecularity, and composition mode, in that order.

First digit `0` for isolated molecules and nanoclusters. `3` for crystals (other dimensionalities will be re-added soon).

Second digit `0` for atom-based prediction. `1` for prediction using molecular units.

Third digit `0` for fixed composition, `1` for variable composition, and `2` for compositional ranges in isolated molecules and nanoclusters.

Default: `300`

Format:

```
300 :  calculationType
```

Use `300/301` for atomic crystals and `310/311` for molecular crystals (Section 5.2). For isolated molecules and nanoclusters, see Section 5.3. For a variable number of copies of one formula unit, see Section 5.1.

▷ *variable* dimensionality

Meaning: Sets the periodicity. Use `0` for isolated systems or `3` for bulk crystals.

Default: `3`

Format:

```
0 :  dimensionality
```

If you also set `calculationType`, use a compatible mode. See Section 5.3 for compositional ranges.

▷ *variable* optType

Meaning: Specifies the objectives of the search.

Default: `enthalpy`

Format:

```
% optType
enthalpy
% EndOptType
```

By default, USPEX minimizes the specified properties. To maximize a property, put a minus sign before its name, for example `-density`. `volume` refers to the volume per atom.

Use one objective for single-objective optimization or two or three for Pareto optimization. Separate objectives with spaces. An objective can be a property name or a mathematical expression in property names. Write expressions without spaces. The example minimizes energy and maximizes the user-defined property from Section 5.5.3.

```
% optType
energy -external_property
% EndOptType
```

▷ *variable* atomType

Meaning: Describes the identity of each type of atom.

Default: none, must specify explicitly

Format:

```
% atomType
Mg Si O
% EndAtomType
```

▷ *variable* numSpecies

Meaning: Lists element counts in `atomType` order. For fixed composition, give one row. For variable composition, give one row per compositional block. The block format is the

same for crystals and clusters.

Default: None (you must specify this block)

Format:

```
% numSpecies
4 4 12
% EndNumSpecies
```

For elements A, B, and C, rows `2 0 3` and `0 1 1` define blocks A_2C_3 and BC. The following block lets all three element counts vary independently.

```
% numSpecies
1 0 0
0 1 0
0 0 1
% EndNumSpecies
```

▷ *variable* ExternalPressure

Meaning: Specifies external pressure at which you want to find structures, in GPa.

Default: `0`

Format:

```
100 :  ExternalPressure
```

NOTE: As of USPEX version 9.4.1 pressure value (in GPa) is set by the tag `ExternalPressure` in the `INPUT.txt` file. **Please:** do not specify it in relaxation files in the `Specific/` folder.

3.5 Population

▷ *variable* populationSize

Meaning: The number of structures in each generation; size of initial generation can be set separately, if needed.

Default: Set dynamically in USPEX versions 25–26 (see Table 1).

Format:

```
20 :  populationSize
```

▷ *variable* initialPopSize

Meaning: The number of structures in the initial generation.

Default: Set dynamically in USPEX versions 25–26 (see Table 1).

Format:

```
20 :  initialPopSize
```

▷ *variable* numGenerations

Meaning: Maximum number of generations allowed for the simulation. The simulation can terminate earlier, if the same best structure remains unchanged for `stopCrit` generations.

Default: Set dynamically in USPEX versions 25–26 (see Table 1).

Format:

```
50 :  numGenerations
```

▷ *variable* stopCrit

Meaning: In fixed-compositional mode the simulation is stopped if the best structure did not change for stopCrit generations, or when `numGenerations` have expired — whichever happens first.

Default: Set dynamically in USPEX versions 25–26 (see Table 1).

Format:

```
20 :  stopCrit
```

In the table, P denotes `populationSize` and G denotes `numGenerations`. Each symbol uses the value set in `INPUT.txt` or its default from the table if omitted. N is the maximum number of structural units allowed by the composition settings. For variable composition, n_b counts the compositional blocks, one per row of `numSpecies`. For example, if each row represents one element, a two-element system has two blocks.

$$\mathrm{round10}(x) = 10\,\mathrm{round}\left(\frac{x}{10}\right).$$

Table 1: Defaults for omitted population parameters

Search mode	`populationSize`	`initialPopSize`	`numGenerations`	`stopCrit`
Fixed composition	$\min(60, \max(10, \mathrm{round10}(2N)))$	P	$2N$	N
Single block	$\min(90, \mathrm{round10}(N+60))$	P	$2N$	N
Variable composition				
2 blocks	120	200	60	30
3 blocks	200	300	70	35
4 blocks	300	400	80	40
5 or more blocks	$100(n_b-1)$	$100n_b$	$10n_b+40$	$G/2$

For five or more blocks, the default `stopCrit` is half the generation limit, including when you set that limit yourself.

Important note. If `INPUT.txt` does not exist, the template generator writes these four settings explicitly.

```
20 :  populationSize
20 :  initialPopSize
10 :  numGenerations
20 :  stopCrit
```

Internally, `stopCrit` is written to the `stop_crit` entry of `chang_pool` and also updates the `num_generations` entry of `best_envelope_not_changes`.

3.6 Survival of the fittest and selection

▷ *variable* `bestFrac`

Meaning: Fraction of the current generation that shall be used to produce the next generation.

Default: `0.7`

Format:

```
0.7 :  bestFrac
```

NOTE: This is an important parameter, values between 0.5–0.8 are reasonable.

See also `keepBestHM` in Section 3.15.

3.7 Structure generation

▷ *variable* `symmetries`

Meaning: Selects space groups for RandSym and PyXtal in crystal structure prediction (accepted numbers 1–230). During generation, special Wyckoff sites can be produced from general positions (Fig. 5). For molecules and nanoclusters, give point-group names using the accepted Schönflies spellings shown below.

Default: For 3D crystals: 2–230. For molecules and nanoclusters, PyXtal uses 32 crystallographic and 23 selected noncrystallographic point groups.

Format:

```
% symmetries
195-198 200 215-230
% EndSymmetries
```

The crystal example uses space-group numbers. The point-group options for molecules and nanoclusters are listed below.

▷ *variable* pyxtalSymmetries

The pyxtalSymmetries block takes point-group IDs from the PyXtal catalog for atomic cluster modes 000, 001 and 002. By default, PyXtal uses 32 crystallographic and 23 selected noncrystallographic point groups. Its catalog has 56 entries because IDs 17 and 51 refer to the same group. The following selects all 55 distinct point groups.

```
% pyxtalSymmetries
1-50 52-56
% EndPyxtalSymmetries
```

1	C_1	15	D_{4h}	29	T_h	43	D_{5h}
2	C_i	16	C_3	30	O	44	D_{7h}
3	C_2	17	$C_{3i} = S_6$	31	T_d	45	D_{8h}
4	C_s	18	D_3	32	O_h	46	D_{4d}
5	C_{2h}	19	C_{3v}	33	C_5	47	D_{5d}
6	D_2	20	D_{3d}	34	C_7	48	D_{6d}
7	C_{2v}	21	C_6	35	C_8	49	D_{7d}
8	D_{2h}	22	C_{3h}	36	D_5	50	D_{8d}
9	C_4	23	C_{6h}	37	D_7	52	S_8
10	S_4	24	D_6	38	D_8	53	S_{10}
11	C_{4h}	25	C_{6v}	39	C_{5v}	54	S_{12}
12	D_4	26	D_{3h}	40	C_{7v}	55	I
13	C_{4v}	27	D_{6h}	41	C_{8v}	56	I_h
14	D_{2d}	28	T	42	C_{5h}		

The 23 noncrystallographic groups in the PyXtal catalog are C_5, C_7, C_8, D_5, D_7, D_8, C_{5v}, C_{7v}, C_{8v}, C_{5h}, D_{5h}, D_{7h}, D_{8h}, D_{4d}, D_{5d}, D_{6d}, D_{7d}, D_{8d}, S_8, S_{10}, S_{12}, I, and I_h.

Point-group names in symmetries. For molecules and nanoclusters, symmetries takes the case-sensitive RandSym spellings of Schönflies symbols. Separate names with spaces. Numeric group IDs are accepted in pyxtalSymmetries. Use E for the identity group and Cv2, Ch2, Dh2 and Dd2 for C_{2v}, C_{2h}, D_{2h} and D_{2d}, respectively. Named ranges such as C2-C6 are unsupported. An omitted or empty block uses RandSym's built-in list, which excludes C7 and C8. Use pyxtalSymmetries separately for cluster PyXtal.

```
% symmetries
E C2 Cv2 Td Oh Ih
% EndSymmetries
```

For crystal structure prediction, use space-group numbers 1–230, for example 1 2 3-15. The block applies to both RandSym and PyXtal.

▷ *variable* splitInto

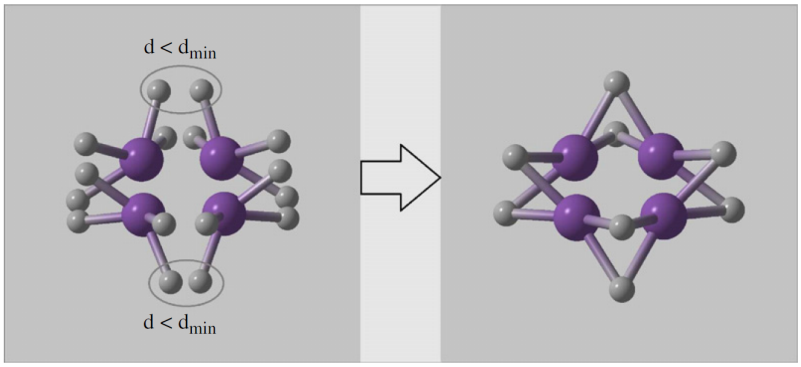


Figure 5: **Example of random symmetric structure generation and merging atoms onto special Wyckoff positions (for detail, see Ref.[18]).**

Meaning: Defines the number of identical subcells or pseudosubcells in the unit cell. If you do not want to use splitting, just use the value 1, or delete the block. Use splitting only for systems with >25–30 atoms/cell.

Default: `1`

Format:

```
% splitInto (number of subcells into which the unit cell is split)
1 2 4
% EndSplitInto
```

Subcells introduce extra translational (pseudo)symmetry. In addition to this, each subcell can be built using a special space-groups algorithm developed by A.R. Oganov and H.T. Stokes and implemented by H.T. Stokes (see Ref.[18]).

In USPEX 26.0, split-cell generation is available for atomic RandSym/PyXtal structure generation. The implemented mode is the exact split-cell mode; the pseudo-subcell mode is not used in this branch yet. Cell-size and cell-shape filters are applied to avoid unsuitable random cells.

3.8 Variation Operators

This section describes the parameters that control how USPEX creates new candidate structures from the current population. The main variation operators are heredity, random symmetric generation, topological random generation, PyXtal generation, permutation, softmutation, coordinate mutation, translational mutation, lattice mutation, and transmutation. For molecular crystals, the corresponding operators work with molecular units and include heredity, mutation, permutation, rotation, lattice mutation, and softmutation.

3.8.1 Operator fractions

The following parameters set relative weights for the operators. USPEX normalizes explicit weights and assigns zero weight to unspecified operators. See Section 5.3.2 for cluster operators and Section 5.5.1 for external generators.

▷ *variable* AutoFrac

Meaning: Set to 1 to adapt the operator weights during the run. USPEX favors operators that successfully generate distinct, low-energy structures, promoting diversity in the population. Set to 0 to keep the initial weights of the operators.

Default: 0

Format:

```
1 :  AutoFrac
```

▷ *variable* fracGene

Meaning: Percentage of structures obtained by heredity; 0.5 means 50%, *etc.*

Default: 0.4

Format:

```
0.40 :  fracGene (fixed, variable)
```

▷ *variable* fracRand

Meaning: Fraction of the generation produced by the random symmetric structure generator of Stokes and Oganov.

Default: 0.0

Format:

```
0.00 :  fracRand (fixed, variable)
```

▷ *variable* fracTopRand

Meaning: Percentage of structures obtained by the topological random generator of Bushlanov, Blatov and Oganov.

Default: 0.1

Format:

```
0.10 :  fracTopRand (fixed, variable)
```

▷ *variable* fracPyxtal

Meaning: Specifies the percentage of structures obtained by PyXtal, which uses another variant of random symmetric structure generation[27;28;29].

Default: 0.1

Format:

```
0.10 :  fracPyxtal (fixed, variable)
```

RandSym, RandTop, and PyXtal random structure generation can run in parallel in USPEX 26.0, which is especially useful on local multicore workstations and Windows PCs.

▷ *variable* fracPerm

Meaning: Percentage of structures obtained by permutation; 0.1 means 10%, *etc.*

Default: 0.1

Format:

```
0.10 :  fracPerm (fixed); 0.00 :  fracPerm (variable)
```

▷ *variable* fracAtomsMut

Meaning: Specifies the percentage of structures obtained by softmutation or coordinate mutation.

Default: 0.2

Format:

```
0.20 :  fracAtomsMut (fixed, variable)
```

▷ *variable* fracTransMut

Meaning: Specifies the percentage of structures obtained by translational mutation.

Default: 0.0

Format:

```
0.00 :  fracTransMut (fixed); 0.20 :  fracTransMut (variable)
```

▷ *variable* fracLatMut

Meaning: Specifies the percentage of structures obtained by lattice mutation.

Default: 0.0

Format:

```
0.00 :  fracLatMut (fixed, variable)
```

3.8.2 Mutation settings

▷ *variable* mutationRate

Meaning: Standard deviation of the strain matrix components for lattice mutation. The strain matrix components are selected randomly from the Gaussian distribution and are only allowed to take values between -1 and 1. Lattice mutation essentially incorporates the ideas of metadynamics into our method[6;30], where new structures are found by building up cell distortions of some known structure. Unlike in metadynamics, the distortions are not accumulated in our method, so the strain components should be large enough to obtain new structures.

Default: 0.5

Format:

```
0.5 :  mutationRate
```

It is a good idea to combine lattice mutation with a weak softmutation:

▷ *variable* mutationDegree

Meaning: The maximum displacement in softmutation in Å. The displacement vectors for softmutation or coordinate mutation are scaled so that the largest displacement magnitude equals mutationDegree.

Default: 3×(average atomic radius)

Format:

```
2.5 :  mutationDegree
```

▷ *variable* ordering_active

Meaning: Switch on the biasing of variation operators by local order parameters.

Default: 1

Format:

```
1 :  ordering_active
```

3.8.3 Heredity and permutation settings

USPEX 26.0 includes the full heredity/crossover operator set for atomic fixed-composition, single-block, and variable-composition calculations, including multi-parent crossover.

▷ *variable* percSliceShift

Meaning: Probability of shifting slabs (used in heredity) in all dimensions, 1.0 means 100%.

Default: 1.0

Format:

```
0.5 :  percSliceShift
```

▷ *variable* howManySwaps

Meaning: For permutation, the number of pairwise swaps will be randomly drawn from a uniform distribution between 1 and howManySwaps.

Default: 0.5×(maximum number of possible swaps). If atoms Na and Nb, and atoms Nc and Nd are swappable, then the total number of possible swaps is $\min(Na, Nb) + \min(Nc, Nd)$, and the default for howManySwaps is $0.5 \times [\min(Na, Nb) + \min(Nc, Nd)]$. In most cases, it is a good idea to rely on this default.

Format:

```
5 :  howManySwaps
```

▷ *variable* specificSwaps

Meaning: Specifies which atom types you allow to swap in permutation.

Default: blank line, which means no specific swaps and all atoms are permutable.

Format:

```
% specificSwaps
1 2
% EndSpecific
```

NOTE: In this case, atoms of type 1 can be swapped with atoms of type 2. If you want to try all possible swaps, just leave a blank line inside this keyblock, or delete the block.

▷ *variable* maxDistHeredity

Meaning: Specifies the maximal fingerprint cosine distances between structures that participate in heredity. This specifies the radius on the landscape within which structures can mate. Use with care (or do not use at all).

Default: 0.5

Format:

```
0.5 :  maxDistHeredity
```

▷ *variable* manyParents

Meaning: Specifies whether more than two slices (or more than two parent structures) should be used for heredity. This may be beneficial for very large systems.

Possible values (integer):

0 — only 2 parents are used, 1 slice each.

1 — many structures are used as parents, 1 slice each.

2 — two structures are used as parents, many slices (determined dynamically using parameters `minSlice` and `maxSlice`) are chosen independently from each one.

3 — two structures are used as parents, many slices (determined dynamically using parameters `minSlice` and `maxSlice`) are cut from the cell with a fixed offset. This is the preferred option for large systems. For example, we cut both structures into slices of approximately the same thickness and then choose the even slices from parent 1 and odd slices form parent 2, making a multilayered "sandwich", or a "zebra".

Default: `0`

Format:

```
3 :  manyParents
```

minSlice, maxSlice: Determines the minimal and maximal thickness of the slices in Å that will be cut out of the parent structures to participate in creation of the child structure. We want the slices to be thick enough to carry some information about the parent (but not too thick to make heredity ineffective). Reasonable values for these parameters are around 1 and 6 Å, respectively.

For clusters, you can directly specify the number of parents participating in heredity (but we found this to be of little use):

▷ *variable* numberparents

Meaning: Defines the number of parents in heredity for clusters.

Default: `2`

Format:

```
2 :  numberparents
```

3.8.4 Transmutation

▷ *variable* fracTrans

Meaning: Percentage of structures obtained by transmutation. In this operator, a randomly selected atom is transmuted into another chemical species present in the system — the new chemical identity is chosen randomly by default.

Default: `0.1`

Format:

```
0.1 :  fracTrans
```

▷ *variable* howManyTrans

Meaning: Maximum percentage of atoms in the structure that are being transmuted (0.1 = 10%). The fraction of atoms that will be transmuted is drawn randomly from a homogeneous distribution bounded from 0 to the fractional parameter `howManyTrans`.

Default: `0.2`

Format:

```
0.2 :  howManyTrans
```

3.9 Constraints

The same structure can be represented in an infinite number of coordinate systems ("modular invariance"). Many of these equivalent choices will lead to very flat unit cells, which creates problems for structure relaxation and energy calculation (*e.g.*, a very large number of k-points are needed). The constraint, well known in crystallography, that the cell angles be between 60° and 120°, does not remove all redundancies and problematic cells (*e.g.*, thus allowed cells with $\alpha = \beta = \gamma \sim 120°$ are practically flat). Therefore, we developed[31;32] a scheme to obtain special cell shapes with the shortest cell vectors. This transformation can be performed if there is at least one lattice vector whose projection onto any other cell vector or the diagonal vector of the opposite cell face is greater (by modulus) than half the length of that vector, *i.e.*, for pairs $\mathbf{a}$ and $\mathbf{b}$, or $\mathbf{c}$ and $(\mathbf{a}+\mathbf{b})$ these criteria are:

$$\left|\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{b}|}\right| > \frac{|\mathbf{b}|}{2} \tag{2}$$

$$\left|\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}|}\right| > \frac{|\mathbf{a}|}{2} \tag{3}$$

$$\left|\frac{\mathbf{c}\cdot(\mathbf{a}+\mathbf{b})}{|\mathbf{c}|}\right| > \frac{|\mathbf{c}|}{2} \tag{4}$$

$$\left|\frac{\mathbf{c}\cdot(\mathbf{a}+\mathbf{b})}{|\mathbf{a}+\mathbf{b}|}\right| > \frac{|\mathbf{a}+\mathbf{b}|}{2} \tag{5}$$

For instance, for the criterion 2 the new vector $\mathbf{a}^*$ equals:

$$\mathbf{a}^* = \mathbf{a} - \text{ceil}\left(\frac{|\mathbf{a}\cdot\mathbf{b}|}{|\mathbf{b}|^2}\right)\text{sign}(\mathbf{a}\cdot\mathbf{b})\mathbf{b} \tag{6}$$

This transformation is performed iteratively, completely avoids pathological cell shapes, and thus solves the problem. During this transformation, the atomic fractional coordinates are transformed so that the original and transformed structures are identical (during the transformation, the Cartesian coordinates of the atoms remain invariant).

Commonly used computational methods (pseudopotentials, PAW, LAPW, and many parametric forcefields) fail when the interatomic distances are too small. This situation needs to be avoided by specifying the minimum distances between each pair of atoms using the `IonDistances` square matrix:

▷ *variable* `IonDistances`

Meaning: Sets the minimum interatomic distance matrix between different atom types. Structures with distances lower than `IonDistances` will be strictly discarded.

Default: the `IonDistances` between atom A and B are estimated as $0.22 \times (V_A^{1/3} + V_B^{1/3})$ but not larger than 1.2 Å, and $0.45 \times (V_A^{1/3} + V_B^{1/3})$ in molecular calculations, where V_A and V_B are the default volumes of atom A and B estimated in USPEX.

Format:

```
% IonDistances
1.0 1.0 0.8
1.0 1.0 0.8
0.8 0.8 1.0
% EndDistances
```

NOTE: The dimensions of this matrix must be equal to the number of atomic species. If the compound in the example above is $MgSiO_3$, the matrix reads as follows: the minimum Mg–Mg distance allowed in a newly generated structure is 1.0 Å, the minimum Mg–Si, Si–Si and O–O distances are also 1.0 Å, and the minimum Mg–O and Si–O distances are 0.8 Å. You can use this keymatrix to incorporate further system-specific information: *e.g.*, if you know that Mg atoms prefer to be very far apart and are never closer than 3 Å in your system, you can specify this information. Beware, however, that the larger these minimum distances, the more difficult it is to generate structures fulfilling these constraints (especially for large systems), so strive for a compromise and remember that `IonDistances` must be **much** smaller than the actual distances in the crystal: realistic distances will be achieved by structure relaxation. What `IonDistances` trick does is to avoid structures which cannot be relaxed correctly.

3.10 Cell

It is useful to create all new structures (before relaxing them) with a unit cell volume appropriate for given conditions. This can be specified in the `Latticevalues` keyblock:

▷ *variable* `Latticevalues`

Meaning: Specifies the initial volume of the unit cell or known lattice parameters.

Default: For cell volumes you don't have to specify values — USPEX has a powerful algorithm to make reasonable estimates at any pressure.

Format:

```
% Latticevalues
125.00
% Endvalues
```

Notes: (1) This volume provides an initial guess for structure relaxation. A variable-cell relaxation can change it. An ASE stage configured with `aseFilter` set to `none` keeps the cell fixed. This keyblock also has another use: when you know the lattice parameters (*e.g.*, from experiment), you can enter the three lattice vectors as a 3×3 matrix in the `Latticevalues` block for a 3D crystal. For example:

```
% Latticevalues
7.49 0.0 0.0
0.0 9.71 0.0
0.0 0.0 7.07
% Endvalues
```

Alternatively, you can specify unit cell parameters just by listing `a`, `b`, `c`, α, β, and γ values:

```
% Latticevalues
10.1 8.4 12.5 90.0 101.3 90.0
% Endvalues
```

Attention: if you do a calculation with a fixed monoclinic cell, please use setting with special angle β (standard setting).

(2) For variable-composition crystal structure prediction, you can supply initial volumes for the end members of the compositional search space. For example:

```
% Latticevalues
12.5 14.0 11.0
% Endvalues
```

(3) Users no longer need to specify the unit cell or atomic volumes in the keyblock `Latticevalues` — a special algorithm has been implemented that accurately estimates it at the pressure of interest, without the need for the user to specify it. This option works well and is available for any `calculationType` where input volumes are required. You

can also use the online program https://uspex-team.org/online_utilities/volume_estimation. The users can also input the volumes manually.

(4) If you study molecular crystals under pressure, you might sometimes need to increase the initial volumes somewhat, in order to be able to generate initial random structures.

3.11 Details of *ab initio* calculations

USPEX employs a powerful two-level parallelization scheme, making its parallel scalability exemplary. The first level of parallelization is performed within structure relaxation codes, the second level of parallelization distributes the calculation over the individuals in the same population (since structures within the same generation are independent of each other).

First, you must specify which code(s) you want to use for structure relaxation and fitness calculation:

▷ *variable* abinitioCode

Meaning: Defines the code used for every optimization step.

Default: 1 for every optimization step (VASP)

Format:

```
% abinitioCode
3 1 1 1
% ENDabinit
```

Alternative Format:

```
% abinitioCode
1 1 1 1 1 (14)
% ENDabinit
```

The difference is that here one (or more) stage(s) of calculation for each structure is shown in parentheses. This is very useful when optimizing physical properties: for calculations in parentheses, the structure is symmetrized and represented in the standard crystallographic setting; this allows us to fully use crystal symmetry and make property calculations cheaper and more numerically robust. Before symmetrization and property calculations, the structure must be well relaxed (so the user must make sure that relaxation is well done).

Note 1: the enthalpy is taken from the last stage **BEFORE** parentheses.

Note 2: Numbers indicate the code used at each step of structure relaxation:

0 — built-in MLIP backend	2 — ORCA
1 — VASP	3 — GULP

`-103` — GULP with space-group preservation

`13` — MOPAC

`20` — ASE interface

`99` — user-defined external relaxation

▷ *variable* potential

Meaning: Selects the potential used with `abinitioCode=0`. The alias `mlip` is also accepted.

Default: `mattersim`

Format:

```
dpa4 :  potential
```

Use `mattersim` for MatterSim or `dpa4` for DPA4. See Section 5.4 for model settings.

▷ *variable* d3

Meaning: Enables D3 dispersion correction.

Default: `0`

Format:

```
1 :  d3
```

Use `1/0`, `true/false`, `yes/no`, or `on/off`. For example, `1 :  d3` adds a D3 dispersion correction to the force field you use.

▷ *variable* KresolStart

Meaning: Specifies the reciprocal-space resolution for k-point generation (units: $2\pi\text{Å}^{-1}$).

Default: from 0.2 to 0.08 linearly

Format:

```
% KresolStart
0.2 0.16 0.12 0.08
% Kresolend
```

You can specify one value per relaxation step. Larger values give coarser meshes. Use coarser meshes for early steps and a finer mesh for the final step.

▷ *variable* numParallelCalcs

Meaning: Sets how many structure relaxations can run in parallel. Active parallel structure generators also use this setting as a shared limit.

Default: Sequential relaxation

Format:

```
10 :  numParallelCalcs
```

When this parameter is omitted, USPEX can use the CPU count available where structure generation runs to choose the number of parallel generation tasks.

▷ *variable* GeneratorTimeoutSeconds

Meaning: Timeout, in seconds, for one structure generation task. This common generator limit is used for atomic and molecular crystal workflows, and is especially important for parallel RandSym, PyXtal, and RandTop jobs: if a generator task exceeds the limit, USPEX can discard it and continue the run instead of waiting indefinitely.

Default: 120

Format:

```
120 :  GeneratorTimeoutSeconds
```

You need to supply the job submission files or the names of executable files for each code/mode you are using.

▷ *variable* commandExecutable

Meaning: Specifies the name of the job submission files or executables for a given code.

Default: no default, has to be specified by the user.

Format:

```
% commandExecutable
gulp < input > output
mpirun -np 8 vasp > out
mpirun -np 8 vasp > out
mpirun -np 8 vasp > out
% EndExecutable
```

NOTE: Every line corresponds to a stage of relaxation — the first line describes the execution of the first stage of relaxation, *etc.* For example, `abinitioCode` equal to "`3 1 1 1`" means that the first relaxation step will be performed with GULP, while the subsequent steps will be performed using VASP via the command "`mpirun -np 8 vasp > out`". If only one line is present in `commandExecutable`, then the same execution will be performed for all steps of relaxation. For `abinitioCode`=99, this block defines the command for the user-selected external code.

For `99`, see Section 5.5.4 for the input example and the files your program reads and writes.

▷ *variable* whichCluster

Meaning: Selects how relaxation jobs run.

- `-1` — run locally without a scheduler or CPU binding by default.
- `0` — run locally without a scheduler.
- `1` — submit locally through a job script.
- `2` — submit to a remote computing cluster.
- `3` — run on a Linux cluster over SSH without a scheduler.

Default: `0`

Format:

```
0 :  whichCluster
```

See Section 3.15.2 for assigning jobs to specific CPUs.

`sleepSeconds` sets the wait in seconds between sequential structure relaxations.

3.12 Fingerprint settings

Please read (Oganov & Valle, 2009[16]) for details on fingerprint functions.

▷ *variable* `RmaxFing`

Meaning: Distance cutoff (in Å).

Default: `10.0`

Format:

```
10.0 :  RmaxFing
```

▷ *variable* `deltaFing`

Meaning: Discretization (in Å) of the fingerprint function.

Default: `0.08`

Format:

```
0.10 :  deltaFing
```

▷ *variable* `sigmaFing`

Meaning: Gaussian broadening of interatomic distances.

Default: `0.03`

Format:

```
0.05 :  sigmaFing
```

`toleranceFing` (default `0.008`) is the minimum cosine distance between fingerprints for USPEX to treat two structures as distinct. When checking for duplicates, you can use a custom program in place of the fingerprint check or run it after that check (Section 5.5.2).

3.13 Space group determination

▷ *variable* doSpaceGroup

Meaning: Determines space groups and also writes output in the crystallographic `*.CIF`-format (this makes your life easier when preparing publications, but beware that space groups may often be under-determined if relaxation was not very precise and if very stringent tolerances were set for the symmetry finder). This option is enabled thanks to the spglib-library https://atztogo.github.io/spglib/ code.

Default: `1,` if `calculationType=3**` (`300`, `301`, `310`, `311` — bulk crystals) and `0` otherwise.

Format:

```
1 :  doSpaceGroup (0 - no space groups, 1 - determine space groups)
```

▷ *variable* SymTolerance

Meaning: Precision for symmetry determination using the symmetry finder code. Can be specified either as a number (in Å) or as `high | medium | low` (= `0.05 | 0.10 | 0.20`)

Default: `medium`

Format:

```
medium :  SymTolerance
```

3.14 Variable-composition structure prediction

Use `301` for atomic crystals, `311` for molecular crystals, or `001` for isolated molecules and nanoclusters. Define the compositional blocks in `numSpecies`. `minAt` and `maxAt` set the minimum and maximum numbers of atoms in the primitive cell of a crystal or in a cluster. In mode `311`, they count molecules in the primitive cell. For mode `002`, see Section 5.3.

Atomic variable-composition examples are EX06 and EX07. For molecular variable-composition calculations, use the same molecular-input layout as in EX09 or EX10, but set `calculationType`=311 and define the molecular composition blocks.

▷ *variable* firstGeneMax

Meaning: How many different compositions are sampled in the first generation. If 0, then the number is equal to initialPopSize/4. For binaries, we recommend firstGeneMax=11, for ternaries a higher value is needed, *e.g.* 30.

Default: 11

Format:

```
10 :  firstGeneMax
```

▷ *variable* minAt

Meaning: Minimum number of atoms (for modes 001, 300 and 301) or molecules (mode 311) in the primitive cell or cluster in the first generation.

Default: No default

Format:

```
10 :  minAt
```

▷ *variable* maxAt

Meaning: Maximum number of atoms (for modes 001, 300 and 301) or molecules (mode 311) in the primitive cell or cluster in the first generation. **Note.** Keep maxAt at or below 2–3 times minAt.

Default: No default

Format:

```
20 :  maxAt
```

The transmutation operator parameters fracTrans and howManyTrans are described in Section 3.8.

DAICS database import. DAICS import is available only on Linux. DAICS https://daics.net can be used as an additional source of trial structures for atomic crystal structure prediction. Instead of generating a candidate from scratch, USPEX requests database structures for the composition defined by atomType and numSpecies, imports compatible entries, and then relaxes and ranks them together with structures produced by the other variation operators. DAICS candidates are used by the DAICSFirstGen generator in the first generation and by the DAICS generator in later generations. Python USPEX keeps the DAICS mat_id values that have already been used and passes them back to the DAICS client, so repeated requests do not keep returning the same structures.

▷ *variable* fracDB

Meaning: Generator fraction assigned to structures imported from the DAICS database in each generation. The alias `fracDAICS` is also accepted. This option can enhance structure prediction by adding database-derived candidates; if no database entries match the system, USPEX falls back to random symmetric generation. Currently this option is intended for atomic `calculationType`=300 and 301 runs, and for unary and binary systems. It must not be used for molecular-crystal runs.

Default: `0.10 for atomic crystal runs; 0 disables DAICS`

Format:

```
0.10 :  fracDB
```

This imports 10% of the structures in each generation from the database. If at least one generator fraction is explicitly specified (*e.g.*, `fracRand`, `fracPyxtal`, `fracDB`, `fracGene`), unspecified generator fractions are set to 0 and the specified fractions are normalized to sum to 1.

▷ *variable* daicsCompositionCheckMode

Meaning: Controls how Python USPEX checks the composition of structures imported from DAICS before accepting them. This check is useful because DAICS returns complete database structures, while USPEX must make sure that the imported candidate is compatible with the requested search composition. DAICS is supported only for atomic crystal calculations; do not use `fracDB`/`fracDAICS` in molecular runs. The `symmetries` block does not control DAICS structures, because their space group comes from the database entry.

Possible values (integer):

- 0 — ratio-only check: the stoichiometric ratio must match the requested composition;
- 1 — strict check using the full compositional-space rules. Use this mode when the search uses compositional constraints and you want DAICS imports to pass the same composition filter as internally generated candidates.

Default: `0`

Format:

```
0 :  daicsCompositionCheckMode
```

When the convex hull is available in variable-composition crystal structure prediction, `keepBestHM` retains the structures on the hull (*i.e.*, thermodynamically stable states of the multicomponent system) survive, along with a few metastable states closest to the convex hull — the total number is `keepBestHM`. Also, parameter `stopCrit` has a new meaning in variable-composition calculations: its value (we recommend to set `stopCrit`=20 in

variable-composition runs) indicates the maximum number of generations during which a structure can appear as a parent of new structures.

For variable-composition runs, it is particularly important to set up the first generation wisely. Choose a suitably large initial population size `initialPopSize`. Choose a reasonably large number of different compositions `firstGeneMax` to be sampled in the first generation (but not too large — each composition needs to be sampled several times at least). Keep `maxAt` at or below 2–3 times `minAt`. You may need several runs with different sizes: *e.g.*, 4–8, 8–16, 16–30 atoms, *etc.*

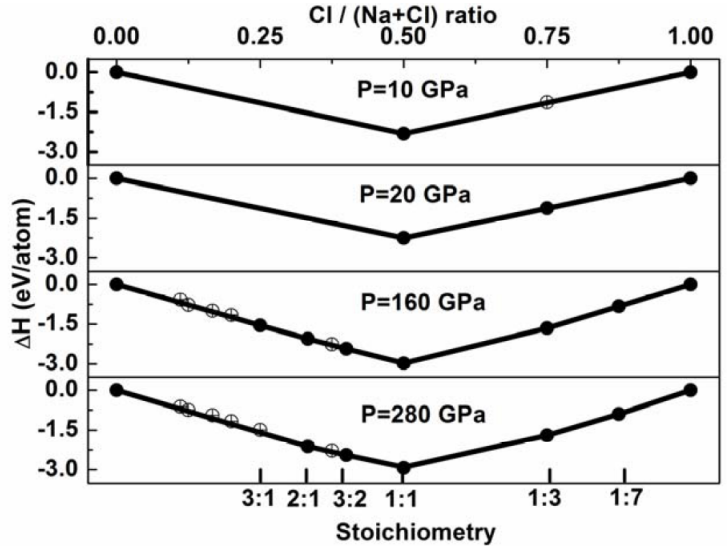


Figure 6: **Convex hull diagram for Na-Cl system at selected pressures.** Solid circles represent stable compounds; open circles — metastable compounds.

An additional comment for VASP users — if you want to perform a variable-composition run, let's say for the Na-Cl system, you should make sure the atomic types are given correctly in `INPUT.txt`, and put pseudopotential files `POTCAR_Na` and `POTCAR_Cl` in the folder `~/StructurePrediction/Specific/`. USPEX will then recognize each atom and take each atom's `POTCAR` file appropriately for the calculations. Convex hull in Fig. 6 shows stable sodium chlorides discovered using USPEX and confirmed by experiment[33].

3.15 Seldom used parameters

▷ *variable* `keepBestHM`

Meaning: Defines how many best structures will survive into the next generation.

Default: `0.15×populationSize`

Format:

```
3 :  keepBestHM
```

▷ *variable* valences

Meaning: Describes the valences of each type of atom. Used only to evaluate bond hardnesses, which are used for computing the approximate dynamical matrix (for softmutation) and hardness of the crystal.

Default: USPEX has a table of default valences (see Appendix 7.7). Beware, however, that for some elements (*e.g.*, N, S, W, Fe, Cr, *etc.*) many valence states are possible. Unless you calculate hardness, just use the default values by not specifying valences. If you do calculate the hardness, you need to carefully and explicitly specify the valence.

Format:

```
% valences
2 4 2
% EndValences
```

▷ *variable* goodBonds

Meaning: Specifes, in the matrix form, the minimum bond valences for contacts that will be considered as important bonds. Like the `IonDistances` matrix (see below), this is a square matrix. This is only used in calculations of hardness and in softmutation. One can estimate these values for a given bond type taking `goodBonds`$=\frac{valence}{max_coordination_number}$ or slightly smaller.

Default: USPEX can make a reasonable default estimation of `goodBonds`. This should be sufficient for most purposes, but for hardness calculations, using Lyakhov-Oganov model, you may need to carefully examine these values and perhaps set them manually. For more details, see Appendix 7.8

Format:

```
% goodBonds
10.0 10.0 0.2
10.0 10.0 0.5
0.2 0.5 10.0
% EndGoodBonds
```

Notes: The dimensionality of this matrix must be equal to either the number of atomic species or unity. If only one number is used, the matrix is filled with this number. The above matrix reads as follows: to be considered a bond, the Mg–Mg distance should be short enough to have bond valence of 10 or more, the same for Mg–Si, Si–Si, and O–O bonds (by using such exclusive criteria, we effectively disregard these interactions from the softmutation and hardness calculations), whereas Mg–O bonds that will be considered for hardness and softmutation calculations will have a bond valence of 0.2 or more, and the Si–O bonds will have a bond valence of 0.5 or more.

▷ *variable* checkConnectivity

Meaning: Switches on/off hardness calculation and connectivity-related criteria in soft-mutation.

Possible values (integer):

- 0 — connectivity is not checked, no hardness calculations;
- 1 — connectivity is taken into account, hardness is calculated.

Default: 0

Format:

```
1 :  checkConnectivity
```

▷ *variable* symmetrize

Meaning: Switches on a transformation of all structures to standard symmetry-adapted crystallographic settings.

Default: 0

Format:

```
1 :  symmetrize
```

▷ *variable* valenceElectr

Meaning: Number of valence electrons for each type of atoms.

Default: these numbers are constants for all atoms, and we have tabulated them, no need to specify explicitly.

Format:

```
% valenceElectr
2 6
% EndValenceElectr
```

▷ *variable* minVectorLength

Meaning: Sets the minimum length of a cell parameter of a newly generated structure.

Default: 1.8 × covalent diameter of the largest atom. For molecular crystals (calculationType = 310, 311) default value is 1.8 × max(MolCenters).

Format:

```
2.0 :  minVectorLength
```

3.15.1 Relaxation time limits and small random perturbations

▷ *variable* relaxationTimeLimit

Sets the time limit for one relaxation step of one structure. Give a positive finite number of seconds or 0 (default) for no limit. Choose a unit explicitly with one of the options below.

▷ *variable* relaxationTimeLimitSeconds

▷ *variable* relaxationTimeLimitMinutes

▷ *variable* relaxationTimeLimitHours

▷ *variable* relaxationTimeLimitDays

The clock starts at the first scientific command, after preparation and any initial queue wait. Retries and restarts use the same budget. Set 30 minutes explicitly to reproduce the former implicit Linux ASE limit.

▷ *variable* relaxationReadResultsOnTimeout

The default 0 treats a timed-out step as failed. Set to 1 to recover a consistent geometry and energy from built-in ASE or VASP. USPEX continues only if recovery succeeds. Both time-limit options accept a scalar or a block with one value for all steps or one value per step.

▷ *variable* relaxationPerturbations

Supply one flag per relaxation step. A 1 enables small random perturbations of the lattice, coordinates, or orientations before that step, as applicable to the structure. This can help a highly symmetric structure relax to a lower-energy geometry. A 0 disables the perturbations. The four-step ase_2 profile uses 0 0 1 0. With explicit abinitioCode, the default enables perturbations on inner steps 2 through $N - 1$, except for code -103.

```
% relaxationTimeLimitMinutes
2 5 15 30
% EndRelaxationTimeLimitMinutes
% relaxationReadResultsOnTimeout
1 1 0 0
% EndRelaxationReadResultsOnTimeout
% relaxationPerturbations
0 1 0 1
% EndRelaxationPerturbations
```

3.15.2 CPU settings

▷ *variable* bindCPUs

Controls whether USPEX assigns jobs to specific logical CPUs. 0 disables this assignment. 1 accounts for CPUs occupied by USPEX's own jobs. 2 additionally selects CPUs by load and is available for local Linux runs. By default, direct runs

with whichCluster 0 or 3 use CPU binding, while scheduler modes and -1 leave it disabled. Scheduler modes disallow explicit binding with 1 or 2. Direct SSH runs support 1.

▷ *variable* allowedCPUs

Block of unique nonnegative logical CPU IDs. Requires binding. If omitted, the pool consists of the CPUs allowed by the execution environment.

▷ *variable* storageMode

0 keeps state in RAM and prevents restart in a new process. 1 (default) uses SQLite FULL. 2 uses SQLite EXTRA for additional write synchronization and can make writes slower.

```
1 :  bindCPUs
1 :  storageMode
% allowedCPUs
0 1 2 3
% EndAllowedCPUs
```

Choose IDs from the CPUs available to your jobs. Set OpenMP and BLAS thread counts and MPI process counts separately. SQLite state is stored in resultsN/uspex.sqlite. Restart also requires compatible settings and external calculation files. The storage backend (RAM or SQLite) must remain the same when resuming a calculation.

4 Main output files

After the run starts, USPEX creates one or more `results`-folders. The example calculations distributed with USPEX contain a `reference/` folder with representative output files. These files are useful for comparing your own run with a known result.

The SQLite state file is `resultsN/uspex.sqlite` when `storageMode` is `1` or `2`. For restart requirements and the RAM mode, see Section 3.15.2.

4.1 Run summary and logs

- `uspex.log` — progress messages and error diagnostics.
- `RELAXATION_STATUS.txt` — status of relaxation jobs at the latest observation.
- `USPEX_IS_DONE` — marker written at the end of a completed run.
- `OUTPUT.txt` — a compact summary of input variables, structures produced by USPEX, and their characteristics. In USPEX 26.0, this file follows the generation-by-generation style of the 10.5/10.6 branch, including structure tables, counts of structures produced by each generator, and runtime volumes. Some example `reference/` folders may omit this file, but the same information is represented in the files below.
- `Parameters.txt` — a copy of the `INPUT.txt` file used in the calculation. This is the first file to check when reproducing or comparing a run.

4.2 Structure information files

- `Individuals` — the main table of generated structures by generation. It includes structure number, composition, energy, cell volume, space group, origin, parents, fitness, and order metrics.
- `Individuals_all` — a compact table of all structures, including structures that did not enter the filtered good set.
- `Individuals_extended` — an extended table with formulas, density, space-group symbols, formation energy, energy above the hull, and order metrics.
- `BESTIndividuals` — the best structure from each generation. Use it to see when the minimum was found and whether the search progressed or stagnated.
- `best_structure` — a one-line summary of the final best structure: generation, id, energy, origin, parents, and energy above the hull.
- `generation_properties` — generated and relaxed population sizes and generation-level correlation coefficients.

4.3 Structures in POSCAR format

- `gatheredPOSCARS` — all relaxed structures in VASP POSCAR format, written as concatenated POSCARs. This file is convenient for visualization and post-processing in STMng, VESTA, or ASE.
- `gatheredPOSCARS_unrelaxed` — unrelaxed structures before relaxation for the good set; these show what USPEX actually generated.
- `gatheredPOSCARS_unrelaxed_all` — all unrelaxed structures, including those not retained by filters or uniqueness selection.
- `BESTgatheredPOSCARS` — concatenated POSCARs for the best structures from each generation.
- `best_structure_POSCAR` — POSCAR of the final best structure. This is usually the fastest file to open when visualizing the found minimum.
- `goodStructures` — unique good structures sorted by stability from best to worst.
- `goodStructures_POSCARS` — the same good structures written as concatenated POSCARs.

4.4 Energies and relaxation diagnostics

- `enthalpies_nospace.dat` — final energy or enthalpy for each good structure, one line per structure and without a header.
- `enthalpies_compete.dat` — energies by relaxation stage. The number of columns follows the number of stages in `abinitioCode`.
- `E2_vs_E1.png`, `E3_vs_E2.png`, ... — scatter plots comparing energies between neighboring relaxation stages. Poor correlation or many outliers can indicate problems in the relaxation setup or external-code input files.

4.5 Variable-composition output

When the standard convex hull can be computed, variable-composition crystal calculations add the files below.

- `convex_hull` — thermodynamically stable compositions on the convex hull; structures on the hull have `e_above_hull`=0.
- `extended_convex_hull` — stable and near-hull metastable structures.

- `convex_hull_POSCARS` — concatenated POSCARs for stable hull structures.
- `extended_convex_hull_POSCARS`
  Concatenated POSCARs for stable and near-hull metastable structures.
- `convex_hull.png` and `convex_hull.pdf`
  Convex-hull plots without labels.
- `convex_hull_labeled.png` and `convex_hull_labeled.pdf`
  Convex-hull plots with labels.
- `convex_hull_with_metastable.png`
  `convex_hull_with_metastable.pdf`
  Hull plots including metastable points.
- `convex_hull_with_metastable_labeled.png`
  `convex_hull_with_metastable_labeled.pdf`
  Labeled hull plots with metastable points; these are often the most useful figures for reports.

4.6 Nanocluster output

For a fixed composition such as EX11 (Si_{12}), start with `best_structure_POSCAR`. For a search over compositional ranges such as EX12 (Si–O), use the composition-resolved outputs below.

- `best_clusters` — the lowest total energy found for each exact composition, with the structure ID from `Individuals`.
- `magic_clusters` — compositions with a nonnegative minimum second difference in energy over available one-atom changes. An asterisk marks missing neighbors. Such an entry gives an incomplete test of stability.
- `best_clusters_POSCARS` and `magic_clusters_POSCARS` — the structures in those tables as concatenated POSCARs.
- `stability_map`, `stability_cells`, `fragmentation_map`, and `fragmentation_cells` — plots of cluster stability or fragmentation by composition. The Si–O example provides PNG and PDF copies.

4.7 How to read structure information files

Common columns in `Individuals`, `goodStructures`, and convex-hull files include:

- `generation`, `number` — generation and global structure id.
- `num_atoms_all` — composition as atom counts in the `atomType` order.
- `energy` — final energy or enthalpy after the last relaxation stage.
- `cell_volume`, `density` — unit-cell volume and density, when printed.
- `space_group` — space-group number; symmetry may be under-determined if relaxation accuracy is low.
- `origin`, `parents` — how the structure was created: random generation, heredity, softmutation, transmutation, or survival from a previous generation.
- `formation_energy` — formation energy relative to end members, mainly used in variable-composition searches.
- `e_above_hull` — distance above the best envelope or convex hull. In a fixed single-composition run it is effectively the distance from the best found minimum; in a variable-composition run it is the energy above the convex hull.
- `opt_property` — objective value used for ranking.
- `quasi_entropy` — a measure of disorder for a structure.
- `a_order`, `s_order` — measures of structural simplicity used by USPEX for analysis and selection.

4.8 How to analyze output

1. Open `Parameters.txt` and confirm that the run used the intended `INPUT.txt`.
2. For fixed-composition calculations, open `best_structure_POSCAR`, then compare the first rows of `goodStructures`.
3. For variable-composition crystal calculations with a computed hull, inspect the labeled metastable convex-hull plot, then read `convex_hull` and `extended_convex_hull`.
4. Check `BESTIndividuals`: when was the best candidate found, and did the search stagnate too early?
5. Check `E*_vs_E*.png` and `enthalpies_compete.dat`: relaxation energy outliers can flag candidates for inspection. Match the structure ID in the unrelaxed and relaxed POSCAR collections, then compare the geometries. Look for unexpectedly short contacts or molecular fragments that have separated. The energy plots alone do not show geometry changes.

6. Visualize the structures, including with STMng, which is designed to read and analyze USPEX output files (see Section 4.9). VESTA or ASE can also be used for POSCAR files. For publication-quality results, re-relax the best structures at higher precision.

4.9 Tool for vizualization

USPEX produces a large set of numbers (structures, energies, *etc*). Post-processing, or analysis of the data, is extremely important. Analysis of these data "by hand" can be quite tedious and time-consuming. USPEX benefits from an interface specifically developed for USPEX by Mario Valle to read and visualize USPEX output files using his STMng visualization toolkit[34], which includes analysis of thousands of structures in a matter of a few minutes, determination of structure-property correlations, analysis of algorithm performance, quantification of the energy landscapes, state-of-the-art visualization of structures, determination of space groups, *etc.*, including preparation of movies showing the progress of the simulation! Fig. 7 shows typical figures produced by STMng. Download it from Mario Valle's STMng project page (`https://github.com/crystalfp/STMng`). As an alternative, see the USPEX downloads page (`https://uspex-team.org/en/codes/`).

Alternatively, you can visualize USPEX results with other software, *e.g.*, VESTA, which can read USPEX structure files directly.

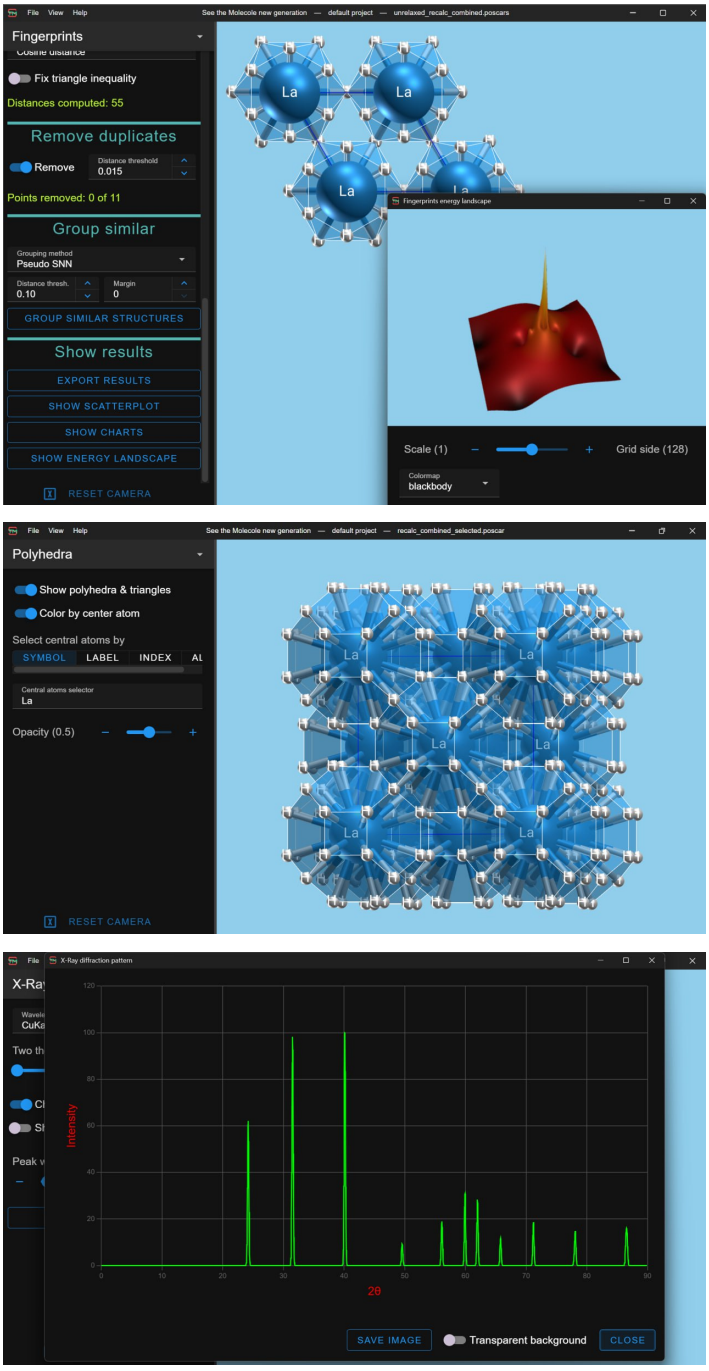


Figure 7: **STMng interface for USPEX.** Top to bottom: structure and energy-landscape views, polyhedral view, and X-ray diffraction spectrum.

5 Additional input for special cases

5.1 Single-block calculations

The single-block mode is a fixed-composition search with a variable number of formula units. In this case, `numSpecies` describes one compositional building block and `minAt`/`maxAt` set the allowed total number of atoms in the unit cell. For example:

```
% atomType
Si O
% EndAtomType

% numSpecies
1 2
% EndNumSpecies

12 : minAt
24 : maxAt
```

This means that USPEX samples SiO_2 structures with the atomic ratio 1:2 and with 12–24 atoms in the unit cell. Starting from USPEX 9.4.1, single-block calculations are specified using `calculationType`=300 for crystals and `calculationType`=310 for molecular crystals. The key settings are `numSpecies`, `minAt`, and `maxAt`.

5.2 Molecular crystals

5.2.1 Molecular crystals, `calculationType`=310/311

To run a molecular-crystal calculation, provide an `XYZ` file with the molecular geometry. However, if you want to use GULP, you must prepare the `MOL` file yourself. See Appendix 7.1.5 for sample `INPUT.txt` files. Molecular-crystal example folders are listed in Appendix 7.9: see EX09 for a GULP/Dreiding `MOL_1` example and EX10 for a MatterSim+D3 `XYZ` example.

USPEX supports molecular crystal structure prediction in two bulk `calculationType` modes: 310 for fixed-composition and single-block molecular calculations, and 311 for variable-composition molecular calculations. Molecular random structures can be generated with RandSym or PyXtal, and molecular variation operators include heredity, mutation, permutation, rotation, lattice mutation, and softmutation. For molecular crystals, USPEX also uses molecular minimum-distance checks and molecular-center checks; the `MolCenters` keyblock can be used when the molecular-center distances need to be controlled explicitly.

Molecular random-generation parameters. ▷ *variable* RandSymTimeLimit

Meaning: Hard timeout, in seconds, for one molecular RandSym generation call. Use this parameter when molecular random generation takes too long for difficult compositions or restrictive distance constraints.

Default: `100.0`

Format:

```
100 :  RandSymTimeLimit
```

▷ *variable* MolCenterShrinkFraction

Meaning: Runtime adaptation factor for molecular center distances. After an unsuccessful molecular generation attempt, USPEX can reduce the target minimum distances between molecular centers by this fraction; the lower bound is 0.8 of the original `MolCenters` matrix.

Default: `0.05`

Format:

```
0.05 :  MolCenterShrinkFraction
```

▷ *variable* VolumeGrowthFraction

Meaning: Controls how much the estimated molecular block volumes increase after an unsuccessful structure generation attempt. USPEX can multiply these estimates by the factor `1 + VolumeGrowthFraction`.

Default: `0.10`

Format:

```
0.10 :  VolumeGrowthFraction
```

▷ *variable* MolCenters

Meaning: Sets the minimum-distance matrix between centers of molecular structural units. For one molecular component this keyblock contains one number; for several components it is a square matrix in the order of the `Species` block.

Default: estimated automatically from molecular components

Format:

```
% MolCenters
7.1
% EndMol
```

▷ *variable* IonDistances

Meaning: Sets atom–atom minimum distances used in molecular structure checks. This keyblock is a square matrix in the `atomType` order and complements `MolCenters`: `MolCenters` controls center–center separations, while `IonDistances` controls direct atomic contacts.

Default: estimated automatically if omitted

Format:

```
% IonDistances
1.0 1.2
1.2 1.0
% EndDistances
```

▷ *variable* relaxedVolumesWeight

Meaning: Controls how much the volumes of relaxed molecules influence the volume estimates used to generate new structures. Smaller values update those estimates more gradually.

Default: `0.1`

Format:

```
0.1 :  relaxedVolumesWeight
```

▷ *variable* PyxtalOneSymmetryTimeLimit

Meaning: Time limit, in seconds, for processing one symmetry in molecular PyXtal generation. This is useful when a particular space group is difficult for the current molecular composition.

Default: `20.0`

Format:

```
20 :  PyxtalOneSymmetryTimeLimit
```

For GULP-based molecular-crystal relaxation, USPEX uses a molecular relaxation workflow that preserves molecular integrity during geometry optimization. The example EX09 shows this workflow for urea with two molecules in the unit cell and Dreiding force-field input for GULP. For a MatterSim+D3 relaxation of molecular CO_2, see EX10 and the `d3` setting in Section 3.11.

The `MOL_1` file lists Cartesian coordinates and the references used to build a Z-matrix. Columns 5–7 identify the atoms that define the bond length, bond angle, and torsion angle, respectively. Use zero for an unused reference. In the benzene example, the first four atoms use `0 0 0`, `1 0 0`, `2 1 0`, and `3 2 1`. The final column marks flexible torsions with `1`. For a rigid molecule, use `1` for the first three atoms and `0` for the rest.

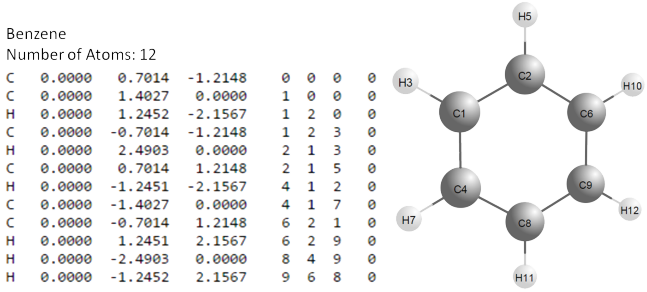


Figure 8: **Benzene geometry and its `MOL_1` file.**

5.2.2 Additional inputs for classical forcefields

The above `MOL_1` files can be used for general cases in USPEX. However, some classical forcefield-based codes need additional information. For instance, GULP needs chemical labels and charges to be specified. The `MOL_1` file for aspirin can be written in the following way:

```
Aspirin_charge
Number of atoms: 21
H_1      0.2310      3.5173      4.8778      0   0   0    1      0.412884
O_R      0.7821      4.3219      4.9649      1   0   0    1     -0.676228
C_R      0.4427      5.0883      6.0081      2   1   0    1      0.558537
O_2     -0.5272      4.5691      6.6020      3   2   1    0     -0.658770
C_R      1.0228      6.3146      6.3896      3   2   4    0      0.116677
C_R      2.1330      6.8588      5.6931      5   3   2    0      0.311483
C_R      0.4810      7.0546      7.4740      5   3   6    0     -0.119320
O_R      2.8023      6.2292      4.6938      6   5   3    0     -0.574557
C_R      2.6211      8.1356      6.0277      6   5   8    0     -0.083091
C_R      0.9966      8.3146      7.8237      7   5   3    0     -0.103442
H_2     -0.3083      6.6848      8.0128      7   5  10    0      0.198534
C_R      3.6352      5.1872      4.9079      8   6   5    0      0.609295
C_R      2.0623      8.8613      7.0940      9   6   5    0     -0.119297
H_2      3.3963      8.5283      5.4906      9   6  13    0      0.174332
H_2      0.5866      8.8412      8.6013     10   7  13    0      0.205960
O_2      3.9094      4.7941      6.0632     12   8   6    0     -0.588433
C_3      4.2281      4.5327      3.7638     12   8  16    0     -0.271542
H_2      2.4227      9.7890      7.3367     13   9  10    0      0.196738
H_2      3.4269      4.1906      3.1183     17  12   8    0      0.151315
H_2      4.8283      3.6848      4.0792     17  12  19    0      0.131198
H_2      4.8498      5.2464      3.2337     17  12  19    0      0.127726
```

Here, the keyword `charge` in the title tells the program to read the charge in the additional (last) column.

5.2.3 How to prepare the MOL files

There are plenty of programs that can generate Z-matrix-style files, such as Molden, Avogadro, and so on. Experienced users might have their own way to prepare these files. For the users' convenience, we have created an online utility to generate a USPEX-style MOL file from a file in XYZ format. Please try this utility at https://uspex-team.org/online_utilities/zmatrix.

5.3 Molecules and nanoclusters

For isolated molecules and nanoclusters, set `calculationType` to `000` for fixed composition, `001` for variable composition, or `002` for compositional ranges.

5.3.1 Composition input for each mode

▷ *variable* `compositionRanges`

For `000`, list the atom counts in element order. For `001`, use `minAt` and `maxAt` to set the minimum and maximum numbers of atoms in the molecule.

For `002`, specify the range of numbers of atoms of each type. For example, to study C–H molecules with 5–10 C atoms and 5–25 H atoms, use the following input.

```
002 :  calculationType
% atomType
C H
% EndAtomType
% numSpecies
1 0
0 1
% EndNumSpecies
% compositionRanges
5 10
5 25
% EndCompositionRanges
```

Give one inclusive integer pair per `numSpecies` row, in the same order, with $0 \leq$ minimum $\leq$ maximum. Use single-atom blocks. `compositionRanges` (alias `blockRanges`) replaces `minAt` and `maxAt`. USPEX excludes the empty composition.

See Appendix 7.1.4 for examples for all three modes. Choose a relaxator configured for nonperiodic structures. Code 99 supports isolated clusters at zero pressure (see Section 5.5.4).

5.3.2 Atomic cluster generation

▷ *variable* `fracGeneCloseCompositions`

Weight for crossover between parents whose atom counts differ by at most one for each element. For example, $C_{10}O_5$ and $C_{11}O_6$ qualify. `fracGeneConstrained` is an alias and must have the same value if both names are supplied.

▷ *variable* `fracAddAtom`

Weight for adding an atom to a cluster.

▷ *variable* `fracRemoveAtom`

Weight for removing an atom from a cluster.

For modes `001` and `002`, each operator gets 15% of the planned new structures in generation 2 by default. With `populationSize` of 100 the plan assigns 15 to each. Failed attempts, fallback operators, seeds and retained structures can change the observed proportions. `AutoFrac` adapts the weights starting in generation 3 when enabled. Set it to `0` to keep the initial plan.

Explicit weights must be finite and nonnegative. A zero weight disables the operator. See Section 3.8 for normalization. For example, the following gives a 50:25:25 weight ratio:

```
0.30 :  fracRand
0.15 :  fracAddAtom
0.15 :  fracRemoveAtom
```

See Section 3.7 for point groups and input examples.

5.4 ASE models and optimization settings

▷ *variable* potentialFamily

Selects `omat24` or `omol25` for DPA4. Defaults to `omat24` when no family is set in `Specific` and no custom checkpoint is provided. Alias `dpa4Family`.

▷ *variable* potentialTier

Selects `nano`, `mini`, `neo`, `air` or `plus` for DPA4. The standard DPA4 tier without a custom checkpoint is `air`. MatterSim tiers are `1M` and `5M`, with the default determined by the profile. `dpa4Tier` and `mattersimTier` are model-specific aliases.

▷ *variable* potentialCheckpoints

Block with model weight paths, one path per line. Use one path for all steps or one entry per step. Paths are relative to the directory containing `INPUT.txt` unless absolute. Use `default` to preserve the standard weight selection. For mixed ML and non-ML chains, use a full list with `default` at non-ML positions. Custom DPA4 weights require the prepared checkpoint format and its accompanying metadata file.

```
dpa4 :  potential
omat24 :  potentialFamily
air :  potentialTier
```

`potentialFamily` and `potentialTier` also accept blocks for step-specific choices.

▷ *variable* aseFmax

Finite force threshold greater than zero, in eV/Å.

▷ *variable* aseMaxSteps

Integer iteration limit of at least one.

▷ *variable* aseOptimizer

`LBFGS`, `BFGS` or `FIRE`.

▷ *variable* aseFilter

`none`, `FrechetCellFilter`, `UnitCellFilter`, `ExpCellFilter` or `StrainFilter`. Use `none` to keep the cell fixed.

Use blocks for these four settings. Supply one value for all ASE steps or one value per step, with `default` at non-ASE positions. `default` preserves the profile setting. For example, the following sets the threshold only at the final stage of a four-step workflow:

```
% aseFmax
default default default 0.005
% EndASEFmax
```

With the blocks omitted, USPEX reads `Specific` and the profile defaults. For the automatically generated `ase_2` workflow, the force thresholds are `0.1 0.02 0.01 0.002` and the optimizer is `LBFGS`. A conflict with an existing ASE profile causes an error.

5.5 User-defined programs

USPEX can call your programs to generate structures, reject individual candidates, compare pairs for duplicates, calculate a search objective, or relax a structure. For each program, specify a command in `INPUT.txt`. The examples below show the input settings, the files USPEX supplies, and the results your program must write.

5.5.1 External structure generators

Choose a command block according to what your generator needs from USPEX: no parent, one or two parents, a target composition, or a target composition and space group.

Supply a nonempty command block to register an external generator. Each command has an optional scalar weight, which defaults to `0.1`. Weights are relative. A zero weight disables the generator. For example:

```
% externalMutationCommand
/absolute/path/to/python /absolute/path/to/mutate.py
% EndExternalMutationCommand
0.2 :  fracExternalMutation
4 :  externalMutationNumChildren
120 :  externalMutationTimeoutSeconds
```

▷ *variable* `externalCreationCommand`

Runs your own generator without a parent structure. Your program chooses the composition.

▷ *variable* `fracExternalCreation`

Relative weight for this command.

▷ *variable* `externalMutationCommand`

Applies your mutation operator to one parent structure after generation 1.

▷ *variable* `fracExternalMutation`

Relative weight for this command.

▷ *variable* `externalCrossoverCommand`

Crosses two parents after generation 1.

▷ *variable* fracExternalCrossover

Relative weight for this command.

▷ *variable* externalCompositionStructureCreationCommand

USPEX supplies the number of each structural unit. Your program generates a structure with those counts.

▷ *variable* fracExternalCompositionStructureCreation

Relative weight for this command.

▷ *variable* externalCompositionSymmetryStructureCreationCommand

USPEX supplies the unit counts and space group. Your program generates a structure that meets both constraints. This generator supports 3D crystals.

▷ *variable* fracExternalCompositionSymmetryStructureCreation

Relative weight for this command.

▷ *variable* externalGeneratorCommand

Calls your program once for its entire allocation of new structures.

▷ *variable* fracExternalGenerator

Relative weight for this command.

Creation and both composition-controlled generators can run from generation 1.

▷ *variable* externalGeneratorNumParents

Defaults to zero, allowing generation 1 use. A positive integer selects that many distinct systems from the start of the prepared parent pool in later generations. This hook supports atomic crystals in modes 300/301 without constLattice.

The first five generators accept the optional batch-size and timeout settings below.

▷ *variable* externalCreationNumChildren

▷ *variable* externalCreationTimeoutSeconds

▷ *variable* externalMutationNumChildren

▷ *variable* externalMutationTimeoutSeconds

▷ *variable* externalCrossoverNumChildren

▷ *variable* externalCrossoverTimeoutSeconds

▷ *variable* externalCompositionStructureCreationNumChildren

▷ *variable* externalCompositionStructureCreationTimeoutSeconds

▷ *variable* externalCompositionSymmetryStructureCreationNumChildren

▷ *variable* externalCompositionSymmetryStructureCreationTimeoutSeconds

`NumChildren` is a positive integer per call, default `1`. Short batches are accepted as returned. Oversized batches are trimmed to the generator's remaining overall quota. USPEX can call the command again while quota and attempts remain. All children in a composition-controlled batch must use the requested composition and, where applicable, symmetry.

▷ *variable* `externalGeneratorTimeoutSeconds`

Sets the timeout for the single call to `externalGeneratorCommand`. USPEX accepts the returned batch without refill or trimming. Use `externalGeneratorNumParents` to set its parent count. `externalGeneratorNumChildren` is unsupported.

All generator timeouts are finite nonnegative seconds, including fractions, and default to `0` for unlimited runtime. They apply to one command invocation. Reading and validating the output occur afterward. Providing a batch size or timeout requires the corresponding nonempty command block, including when its weight is zero.

USPEX runs each command in a separate directory under `Calculation/Generation/<generator>/CalcFold_N/`. It writes `request.json`, the input structures, and the logs `stdout.log` and `stderr.log`. Composition-controlled variants use generator names ending in `FirstGen` in the first generation. For other shortcuts the name identifies the hook, for example `ExternalCreation` or `ExternalMutation`. `N` is a directory allocation counter. The working directory provides all files needed for the call. Standard input is unused. Write results to files and exit with status zero. Standard output and standard error provide logs. This convention also applies to the external checks, comparison and property calculator below.

All six requests contain `num_children`, `generation_number` (counting from 1), `structural_units`, `pbc` (three JSON booleans) and `fixed_lattice` (null or a 3×3 matrix with one lattice vector per row in angstroms). The catalog contains all configured units, in order, with one-based `unit_id`. Atom entries contain `type: "atom"` and `symbol`. Other entries contain `name`, `type` and `template_atoms`, a list of atomic symbols and three-component positions in angstroms (fields `symbol` and `position`). For older scripts, read the atomic unit's `symbol` in place of `name` or `template_atoms`.

```
{
  "operation": "creation",
  "num_children": 1,
  "generation_number": 1,
  "structural_units": [{"unit_id": 1, "type": "atom", "symbol": "Cu"}],
  "pbc": [true, true, true],
  "fixed_lattice": null
}
```

For external mutation and crossover operators, USPEX writes `"operation": "mutation"` or `"operation": "crossover"` to `request.json`, respectively. The `parents` array contains one entry for mutation and two for crossover. `ExternalGenerator` uses `"operation": "generation"` and a `parents` array that may

be empty. Each parent entry gives its `directory`, `system_number` and `composition`. A parent directory such as `parents/parent_1/` contains `POSCAR` and, when needed, `POSCAR_components.json`. The parent composition gives counts of structural units in that parent.

```
{
  "directory": "parents/parent_1",
  "system_number": 7,
  "composition": [{"unit_id": 1, "count": 4}]
}
```

The composition-controlled requests use `composition` as their additional field. The composition is an array such as `[{"unit_id": 1, "count": 4}]`. The symmetry variant also supplies `"symmetry": {"kind": "space_group", "number": 221}`.

Returned structures. Write one `POSCAR` per child, in `children/child_N/`. USPEX reads these directories in numeric order. Numbering can skip values. Use numbers without leading zeros. An optional `results.json` overrides discovery and ordering, for example:

```
{"children": ["children/child_2", "children/child_1"]}
```

The sole key is `children`. Its entries must be unique relative directory paths inside the working directory, without parent traversal or symlinks. An empty array returns no children. An invalid manifest prevents automatic discovery. When one child is requested, the first five generators also accept a top-level `POSCAR` if neither a manifest nor standard child directories exist. `ExternalGenerator` always reads children from separate directories. A nonzero exit status or timeout rejects the batch.

Use POSCAR scale `1.0`, element and count lines, and Direct or Cartesian coordinates. Omit Selective dynamics. USPEX obtains periodicity from `pbc`. For molecular or other composite units and ambiguous atomic units, place a `POSCAR_components.json` beside each POSCAR:

```
{"components": [{"unit_id": 1, "atom_indices": [1, 2, 3]}]}
```

Each entry describes one instance of a unit. Atom indices are one-based POSCAR atom rows in template order, and each atom must appear exactly once. An atomic unit takes a single index of the specified element. This file is optional for unambiguous atomic structures. Additional scientific checks on external children are disabled by default.

5.5.2 Structure checks and duplicate detection

A structure check accepts or rejects one candidate. A duplicate check compares two candidates and reports whether they describe the same structure. Both return a Boolean decision in `results.json`.

▷ *variable* structureCheckAfterGenerationCommand

Runs on generated structures and imported seeds before relaxation.

▷ *variable* structureCheckAfterGenerationTimeoutSeconds

Sets the per-call timeout.

▷ *variable* structureCheckAfterRelaxationCommand

Runs after successful relaxation, before property calculation and selection.

▷ *variable* structureCheckAfterRelaxationTimeoutSeconds

Sets the per-call timeout.

▷ *variable* structureComparisonCommand

Decides whether two candidates are duplicates.

▷ *variable* structureComparisonTimeoutSeconds

Sets the per-call timeout.

The hooks are independent. Supply a nonempty command block for each hook you want to enable. USPEX uses its built-in comparison by default. For example:

```
% structureCheckAfterRelaxationCommand
/absolute/path/to/python /absolute/path/to/check_structure.py
% EndStructureCheckAfterRelaxationCommand
60 :  structureCheckAfterRelaxationTimeoutSeconds
```

Time limits require the corresponding command. They accept finite nonnegative seconds and default to `0` (unlimited). ▷ *variable* structureComparisonReplaceFingerprint

Defaults to `1`, which replaces the built-in fingerprint comparison when the external command is configured. Set it to `0` to run the external comparison after the fingerprint check. Composition and energy checks still apply.

Checking a single structure. Each check gets a `CalcFold_N` directory under one of these paths:

```
Calculation/StructureCheck/AfterGeneration/
Calculation/StructureCheck/AfterRelaxation/
```

USPEX writes `POSCAR`, `POSCAR_components.json` if needed, and `request.json` with just `pbc` and `structural_units`. Write `{"passed": true}` or `{"passed": false}` to `results.json`. A false response rejects the structure. A technical failure skips this hook for the current structure.

Comparing a pair of structures. Each comparison uses a fresh directory under `Calculation/StructureComparison/`, also named `CalcFold_N`. It contains `POSCAR_1` and `POSCAR_2`, optional matching `POSCAR_1_components.json` and `POSCAR_2_components.json`, and a `request.json` with `pbc_1`, `pbc_2` and `structural_units`. Write `{"duplicate": true}` or `{"duplicate": false}` to `results.json`. A technical failure leaves the pair unconfirmed as a duplicate. Use a single JSON boolean field in both response formats. All three hooks run locally, process one structure or one pair per call, and return their decisions without geometry.

5.5.3 User-defined search property

Use this interface when you want USPEX to optimize a quantity calculated by your program. Your program reads the relaxed structure and returns one number.

▷ *variable* externalPropertyCalculatorCommand

Registers an external command to compute a scalar property for the final structure.

▷ *variable* externalPropertyCalculatorFailureValue

Required with the command. Sets the finite fallback value for a failed calculation.

▷ *variable* externalPropertyCalculatorTimeoutSeconds

Sets the per-call timeout in finite nonnegative seconds. The default `0` allows unlimited runtime.

```
% externalPropertyCalculatorCommand
/absolute/path/to/python /absolute/path/to/calculate_property.py
% EndExternalPropertyCalculatorCommand
1000000 :  externalPropertyCalculatorFailureValue
600 :  externalPropertyCalculatorTimeoutSeconds
% optType
external_property
% EndOptType
```

Files sent to your program. For system 42, the command runs in `Calculation/PropertiesCalculation/Calcfold_42/`. USPEX can reuse this directory for the same system. It contains `POSCAR`, optional `POSCAR_components.json`, `request.json` with `pbc` and `structural_units`, and `properties.json`. The latter contains `composition` (present unit counts), `pressure_gpa`, `properties` and `units`. Available values depend on the structure and calculation. For example:

```
{
  "composition": [{"unit_id": 1, "count": 2}],
  "pressure_gpa": 0.0,
  "properties": {"energy": -8.4, "total_atoms": 2},
  "units": {"pressure_gpa": "GPa", "energy": "eV", "total_atoms": "count"}
}
```

Result to return. Write a single finite numeric `external_property` field in `results.json`, for example `{"external_property": 3.14}`. A successful command can return `{"external_property": null}` to request the configured failure value. USPEX also substitutes this value on timeout, invalid result or command failure. Set `optType` to `external_property` to minimize this property, or `-external_property` to maximize it. Either form can be used in a multi-objective search. Choose the failure value for the direction of the objective. The positive value in the example is a penalty for minimization.

5.5.4 99 — Using a user-defined external code

▷ *variable* abinitioCode

▷ *variable* commandExecutable

Set `abinitioCode` to **99** for each step handled by your program. Specify the command in `commandExecutable`, as shown below.

```
% abinitioCode
99
% ENDabinit
% commandExecutable
/absolute/path/to/python /absolute/path/to/relax.py
% EndExecutable
```

Input files written by USPEX

- `geom.in` contains the starting geometry in POSCAR format.
- `parameters.json` describes the run. `schema_version` is `1` and `stage_name` identifies the relaxation stage. `pressure_gpa` gives the pressure in GPa and `pbc` gives the periodicity along the three cell directions. `kpoints` gives the mesh and `k_resol` its target resolution in inverse angstroms.

USPEX uses the cell and `KresolStart` to compute the mesh for crystals. With no block, the resolution goes from 0.2 to 0.08 over the relaxation steps (0.2 for one step). Isolated clusters use `[1, 1, 1]`.

Output files written by your program

- `geom.out` contains the relaxed structure in POSCAR format.
- `results.json` contains exactly one key, `energy` or `enthalpy`. Give the total for the whole structure in eV, for example `{"energy": -10.643}`.

For 3D crystals, USPEX adds pV to a returned energy using the final volume and requested pressure. A returned enthalpy is used directly. The external program applies the requested pressure during relaxation. The protocol supports fully periodic 3D structures at finite scalar pressure and isolated clusters at zero pressure.

A JSON null value signals a scientific failure. Supplying both keys, a nonnumeric result or invalid output geometry prevents success. Code `99` does not recover partial results after a timeout.

To migrate a script that writes `energy.txt`, replace that file with `results.json`. Use `enthalpy` if the old value already included pV, or `energy` for the total energy alone.

6 Online utilities

We have created a number of useful online utilities, which can be used for preparing USPEX input and for post-processing. The utilities are available at:

https://uspex-team.org/online_utilities/

Below you can find information about each one of them.

6.1 USPEX-team codes and utilities

The USPEX-team codes page also lists standalone codes and utilities developed in the laboratory:

- USPEX — the main code for computational materials discovery and structure prediction.
- STMng — visualization and analysis of USPEX results, including fingerprints, matching, clustering, and structure-property analysis. Also available from the USPEX downloads page.
- AICON — fast calculations of lattice thermal conductivity and transport properties.
- FPTE — tools for finite-pressure and finite-temperature elastic constants, including stress-strain calculations and temperature-dependent elastic constants.
- MendS — coevolutionary Mendelevian search for compounds with optimal properties across possible compounds of the chemical elements.
- GTTP — a LAMMPS-based implementation of a fast general two- and three-body machine-learning interatomic potential.
- Automag — automatic search for the most stable magnetic state of a given structure.
- xrpostprocessing — scripts for X-ray post-processing with USPEX.
- a2f — Python scripts for Eliashberg-function calculations and superconducting-state analysis.
- SUPERHEX — supercell optimization for Heisenberg exchange calculations and mapping *ab initio* results to the Heisenberg Hamiltonian.

6.2 Structure characterization

The structure-characterization utilities are:

- Fingerprints — the utility calculates and plots fingerprint function, which is a crystal structure descriptor, a 1D-function related to the pair correlation function and diffraction patterns. It does not depend on absolute atomic coordinates, but only on interatomic distances. Small deviations in atomic positions will influence the fingerprint only slightly, *i.e.* it is numerically robust.
- Multifingerprint — the utility calculates average quasi-entropy, A-order(average atomic order parameter) and S-order(whole-structure order parameter) for a set of structures. Also it filters unique structures by cosine distances difference ≥ 0.003, identifies the symmetry of these structures and lists them in the `uniq_gatheredPOSCARS` file.
- POSCAR2CIF — determines space group and prepares a `CIF` file from a `POSCAR` file.
- POSCAR2CIF+XRD — symmetrizes a crystal structure, prepares a `CIF` file, and computes the X-ray diffraction pattern before and after symmetrization. This is useful for comparing predicted structures with experimental X-ray diffraction data.
- CIF2POSCAR — prepares a `POSCAR` file from a `CIF` file.
- XSF2POSCAR — prepares a `POSCAR` file from a `XSF` (XCRYSDEN) file.

6.3 Properties calculations

The property-calculation utilities are:

- Hardness — the utility is to calculate hardness based on the Mazhnik-Oganov model.
- EELS — the utility calculates the Electron Energy Loss Spectrum (EELS). Written by Priya Johari.

7 Appendices

7.1 Sample INPUT.txt files

7.1.1 Fixed-composition USPEX calculation (calculationType=300):

```
PARAMETERS EVOLUTIONARY ALGORITHM
% Example of the short input, using most options as defaults

########################################
#       TYPE OF RUN AND SYSTEM         #
########################################
USPEX   : calculationMethod (USPEX, VCNEB, META)
300     : calculationType
0       : AutoFrac

% optType
1
% EndOptType

% atomType
Mo B
% EndAtomType

% numSpecies
1 0
0 1
% EndNumSpecies

8       : minAt
18      : maxAt
11      : firstGeneMax
########################################
#              POPULATION              #
########################################
20      : populationSize
20      : initialPopSize
20       : numGenerations
30       : stopCrit
0.6     : bestFrac
########################################
#          VARIATION OPERATORS         #
########################################
0.40    : fracGene
0.05    : fracRand
```

```
0.05   : fracTopRand
0.10   : fracPyxtal
0.00   : fracPerm
0.20   : fracAtomsMut
0.20   : fracTransMut

****************************************
*    DETAILS OF AB INITIO CALCULATIONS    *
****************************************
% abinitioCode
0 0 0
% ENDabinit

8          : numParallelCalcs
2          : coresPerJob
0          : whichCluster

5        : sleepSeconds
```

7.1.2 Variable-composition USPEX calculation (`calculationType`=301):

```
USPEX   : calculationMethod (USPEX, VCNEB, META)
301     : calculationType
0       : AutoFrac

% optType
1
% EndOptType

% atomType
Mo B
% EndAtomType

% numSpecies
1 0
0 1
% EndNumSpecies

8       : minAt
18      : maxAt
11      : firstGeneMax
****************************************
*                POPULATION            *
****************************************
20      : populationSize
20      : initialPopSize
20       : numGenerations
30       : stopCrit
0.6     : bestFrac
****************************************
*          VARIATION OPERATORS         *
****************************************
0.40   : fracGene
0.05   : fracRand
0.05   : fracPyxtal
0.10   : fracTopRand
0.20   : fracPerm
0.20   : fracAtomsMut
0.00   : fracTransMut

****************************************
*   DETAILS OF AB INITIO CALCULATIONS  *
****************************************
% abinitioCode
0 0 0
```

```
% ENDabinit

8          : numParallelCalcs
2          : coresPerJob
0          : whichCluster

5         : sleepSeconds
```

7.1.3 MatterSim input for a fixed-composition crystal

This `INPUT.txt` specifies a silicon crystal search with 8 atoms per cell and 4 MatterSim relaxation steps at zero pressure. Each `0` in the `abinitioCode` block selects the built-in relaxation backend.

```
USPEX : calculationMethod
300 : calculationType
0 : ExternalPressure
% atomType
Si
% EndAtomType
% numSpecies
8
% EndNumSpecies
% abinitioCode
0 0 0 0
% ENDabinit
mattersim : potential
```

To run locally, use `whichCluster=0` (the default) with CPU binding enabled by default, or `-1` with it disabled by default. For the CPU pool, see Section 3.15.2. Add the following settings to run up to 4 relaxations in parallel with one CPU core allocated per job. If you omit `numParallelCalcs`, relaxations run sequentially.

```
0 :  whichCluster
4 :  numParallelCalcs
1 :  coresPerJob
```

Run USPEX in the calculation folder. On the first run, the results are stored in `results1/`. For reference output for a silicon search using MatterSim, see EX02.

7.1.4 Nanocluster inputs

These examples use a custom relaxator at zero pressure. Replace `/path/to/relax.py` with a script that handles isolated clusters and implements the code 99 file protocol (Section 5.5.4).

Fixed composition (000)
Predicts C_6O_2 structures.

```
USPEX : calculationMethod
0 : dimensionality
000 : calculationType
0 : ExternalPressure
% atomType
C O
% EndAtomType
% numSpecies
6 2
% EndNumSpecies
% abinitioCode
99
% ENDabinit
% commandExecutable
python /path/to/relax.py
% EndExecutable
```

Variable composition (001)
Samples clusters with 5–12 atoms in total.

```
USPEX : calculationMethod
0 : dimensionality
001 : calculationType
0 : ExternalPressure
% atomType
C O
% EndAtomType
% numSpecies
1 0
0 1
% EndNumSpecies
5 : minAt
12 : maxAt
% abinitioCode
99
% ENDabinit
% commandExecutable
python /path/to/relax.py
% EndExecutable
```

Compositional ranges (002). In the preceding input, use `002 : calculationType` and replace the two total-count limits with the block below. It sets 5–10 C atoms and 0–2 O atoms.

```
% compositionRanges
5 10
0 2
% EndCompositionRanges
```

7.1.5 Inputs for molecular crystal structure prediction

These MatterSim+D3 inputs use molecular geometries in XYZ format. Place the XYZ files in the calculation folder. List their names without the `.xyz` extension in the `species` block.

Fixed composition (310). Predicts crystals with four CO_2 molecules per cell. Supply `CO2.xyz` as in EX10.

```
USPEX : calculationMethod
310 : calculationType
% species
CO2
% EndSpecies
% numSpecies
4
% EndNumSpecies
% abinitioCode
0 0 0 0
% ENDabinit
mattersim : potential
1 : d3
```

Variable composition (311). Lets the CO_2 and N_2 counts vary with a total of 2–6 molecules. Supply `CO2.xyz` and `N2.xyz`. The columns of `numSpecies` follow the order in `species`.

```
USPEX : calculationMethod
311 : calculationType
% species
CO2 N2
% EndSpecies
% numSpecies
1 0
0 1
% EndNumSpecies
2 : minAt
6 : maxAt
% abinitioCode
0 0 0 0
% ENDabinit
mattersim : potential
1 : d3
```

7.2 List of space groups

1	$P1$	2	$P\text{-}1$	3	$P2$	4	$P2_1$
5	$C2$ $(A2)^*$	6	Pm	7	Pc $(Pa)^*$	8	Cm $(Am)^*$
9	Cc $(Aa)^*$	10	$P2/m$	11	$P2_1/m$	12	$C2/m$ $(A2/m)^*$
13	$P2/c$ $(P2/a)^*$	14	$P2_1/c$ $(P2_1/a)^*$	15	$C2/c$ $(A2/a)^*$	16	$P222$
17	$P222_1$	18	$P2_12_12$	19	$P2_12_12_1$	20	$C222_1$
21	$C222$	22	$F222$	23	$I222$	24	$I2_12_12_1$
25	$Pmm2$	26	$Pmc2_1$	27	$Pcc2$	28	$Pma2$
29	$Pca2_1$	30	$Pnc2$	31	$Pmn2_1$	32	$Pba2$
33	$Pna2_1$	34	$Pnn2$	35	$Cmm2$	36	$Cmc2_1$
37	$Ccc2$	38	$Amm2$ $(C2mm)^*$	39	$Aem2$ $(C2mb)^*$	40	$Ama2$ $(C2cm)^*$
41	$Aea2$ $(C2cb)^*$	42	$Fmm2$	43	$Fdd2$	44	$Imm2$
45	$Iba2$	46	$Ima2$	47	$Pmmm$	48	$Pnnn$
49	$Pccm$	50	$Pban$	51	$Pmma$	52	$Pnna$
53	$Pmna$	54	$Pcca$	55	$Pbam$	56	$Pccn$
57	$Pbcm$	58	$Pnnm$	59	$Pmmn$	60	$Pbcn$
61	$Pbca$	62	$Pnma$	63	$Cmcm$	64	$Cmce$ $(Cmca)^*$
65	$Cmmm$	66	$Cccm$	67	$Cmme$ $(Cmma)^*$	68	$Ccce$ $(Ccca)^*$
69	$Fmmm$	70	$Fddd$	71	$Immm$	72	$Ibam$
73	$Ibca$	74	$Imma$	75	$P4$	76	$P4_1$
77	$P4_2$	78	$P4_3$	79	$I4$	80	$I4_1$
81	$P\text{-}4$	82	$I\text{-}4$	83	$P4/m$	84	$P4_2/m$
85	$P4/n$	86	$P4_2/n$	87	$I4/m$	88	$I4_1/a$
89	$P422$	90	$P42_12$	91	$P4_122$	92	$P4_12_12$
93	$P4_222$	94	$P4_22_12$	95	$P4_322$	96	$P4_32_12$
97	$I422$	98	$I4_122$	99	$P4mm$	100	$P4bm$
101	$P4_2cm$	102	$P4_2nm$	103	$P4cc$	104	$P4nc$
105	$P4_2mc$	106	$P4_2bc$	107	$I4mm$	108	$I4cm$
109	$I4_1md$	110	$I4_1cd$	111	$P\text{-}42m$	112	$P\text{-}42c$
113	$P\text{-}42_1m$	114	$P\text{-}42_1c$	115	$P\text{-}4m2$	116	$P\text{-}4c2$
117	$P\text{-}4b2$	118	$P\text{-}4n2$	119	$I\text{-}4m2$	120	$I\text{-}4c2$
121	$I\text{-}42m$	122	$I\text{-}42d$	123	$P4/mmm$	124	$P4/mcc$
125	$P4/nbm$	126	$P4/nnc$	127	$P4/mbm$	128	$P4/mnc$
129	$P4/nmm$	130	$P4/ncc$	131	$P4_2/mmc$	132	$P4_2/mcm$
133	$P4_2/nbc$	134	$P4_2/nnm$	135	$P4_2/mbc$	136	$P4_2/mnm$
137	$P4_2/nmc$	138	$P4_2/ncm$	139	$I4/mmm$	140	$I4/mcm$
141	$I4_1/amd$	142	$I4_1/acd$	143	$P3$	144	$P3_1$
145	$P3_2$	146	$R3$	147	$P\text{-}3$	148	$R\text{-}3$
149	$P312$	150	$P321$	151	$P3_112$	152	$P3_121$
153	$P3_212$	154	$P3_221$	155	$R32$	156	$P3m1$
157	$P31m$	158	$P3c1$	159	$P31c$	160	$R3m$
161	$R3c$	162	$P\text{-}31m$	163	$P\text{-}31c$	164	$P\text{-}3m1$
165	$P\text{-}3c1$	166	$R\text{-}3m$	167	$R\text{-}3c$	168	$P6$
169	$P6_1$	170	$P6_5$	171	$P6_2$	172	$P6_4$
173	$P6_3$	174	$P\text{-}6$	175	$P6/m$	176	$P6_3/m$
177	$P622$	178	$P6_122$	179	$P6_522$	180	$P6_222$
181	$P6_422$	182	$P6_322$	183	$P6mm$	184	$P6cc$
185	$P6_3cm$	186	$P6_3mc$	187	$P\text{-}6m2$	188	$P\text{-}6c2$
189	$P\text{-}62m$	190	$P\text{-}62c$	191	$P6/mmm$	192	$P6/mcc$
193	$P6_3/mcm$	194	$P6_3/mmc$	195	$P23$	196	$F23$
197	$I23$	198	$P2_13$	199	$I2_13$	200	$Pm\text{-}3$
201	$Pn\text{-}3$	202	$Fm\text{-}3$	203	$Fd\text{-}3$	204	$Im\text{-}3$
205	$Pa\text{-}3$	206	$Ia\text{-}3$	207	$P432$	208	$P4_232$
209	$F432$	210	$F4_132$	211	$I432$	212	$P4_332$
213	$P4_132$	214	$I4_132$	215	$P\text{-}43m$	216	$F\text{-}43m$
217	$I\text{-}43m$	218	$P\text{-}43n$	219	$F\text{-}43c$	220	$I\text{-}43d$
221	$Pm\text{-}3m$	222	$Pn\text{-}3n$	223	$Pm\text{-}3n$	224	$Pn\text{-}3m$
225	$Fm\text{-}3m$	226	$Fm\text{-}3c$	227	$Fd\text{-}3m$	228	$Fd\text{-}3c$
229	$Im\text{-}3m$	230	$Ia\text{-}3d$				

*In parentheses there non-standard space groups used in the code.

7.3 List of layer groups

Triclinic							
1	$p1$	2	$p\text{-}1$				
Monoclinic / inclined							
3	$p112$	4	$p11m$	5	$p11a$	6	$p112/m$
7	$p112/a$						
Triclinic / orthogonal							
8	$p211$	9	$p2_111$	10	$c211$	11	$pm11$
12	$pb11$	13	$cm11$	14	$p2/m11$	15	$p2_1/m11$
16	$p2/b11$	17	$p2_1/b11$	18	$c2/m11$		
Orthorhombic							
19	$p222$	20	$p2_122$	21	$p2_12_12$	22	$c222$
23	$pmm2$	24	$pma2$	25	$pba2$	26	$cmm2$
27	$pm2m$	28	$pm2_1b$	29	$pb2_1m$	30	$pb2b$
31	$pm2a$	32	$pm2_1a$	33	$pb2_1a$	34	$pb2n$
35	$cm2m$	36	$cm2e$	37	$pmmm$	38	$pmaa$
39	$pban$	40	$pmam$	41	$pmma$	42	$pman$
43	$pbaa$	44	$pbam$	45	$pbma$	46	$pmmn$
47	$cmmm$	48	$cmme$				
Tetragonal							
49	$p4$	50	$p\text{-}4$	51	$p4/m$	52	$p4/n$
53	$p422$	54	$p42_12$	55	$p4mm$	56	$p4bn$
57	$p\text{-}4m2$	58	$p\text{-}42_1m$	59	$p\text{-}4m2$	60	$p\text{-}4b2$
61	$p4/mmm$	62	$p4/nbm$	63	$p4/mbn$	64	$p4/nmm$
Trigonal							
65	$p3$	66	$p\text{-}3$	67	$p312$	68	$p321$
69	$p3m1$	70	$p31m$	71	$p\text{-}31m$	72	$p\text{-}3m1$
Hexagonal							
73	$p6$	74	$p\text{-}6$	75	$p6/m$	76	$p622$
77	$p6mm$	78	$p\text{-}m2$	79	$p\text{-}62m$	80	$p6/mmm$

7.4 List of plane groups

Number	Group
1	p1
2	p2
3	pm
4	pg
5	cm
6	pmm
7	pmg
8	pgg
9	cmm
10	p4
11	p4m
12	p4g
13	p3
14	p3m1
15	p31m
16	p6
17	p6m

7.5 List of point groups

List of all crystallographic and the most important non-crystallographic point groups in Schönflies and Hermann-Mauguin (international) notations.

Crystallographic point groups:

Hermann-Maugin	Schönflies	In USPEX
1	C_1	C1 or E
2	C_2	C2
222	D_2	D2
4	C_4	C4
3	C_3	C3
6	C_6	C6
23	T	T
$\overline{1}$	S_2	S2
M	C_{1h}	Ch1
mm2	C_{2v}	Cv2
$\overline{2}$	S_4	S4
$\overline{3}$	S_6	S6
$\overline{6}$	C_{3h}	Ch3
m$\overline{3}$	T_h	Th
2/m	C_{2h}	Ch2
mmm	D_{2h}	Dh2
4/m	C_{4h}	Ch4
32	D_3	D3
6/m	C_{6h}	Ch6
432	O	O
422	D_4	D4
3m	C_{3v}	Cv3
622	D_6	D6
$\overline{4}$3m	T_d	Td
4mm	C_{4v}	Cv4
$\overline{3}$m	D_{3d}	Dd3
6mm	C_{6v}	Cv6
m$\overline{3}$m	O_h	Oh
$\overline{4}$2m	D_{2d}	Dd2
$\overline{6}$2m	D_{3h}	Dh3
4/mmm	D_{4h}	Dh4
6/mmm	D_{6h}	Dh6
m$\overline{3}$m	O_h	Oh

Important non-crystallographic point groups

Hermann-Maugin	Schönflies	In USPEX
5	C_5	C5
5/m	S_5	S5
$\overline{5}$	S_{10}	S10
5m	Cv_{5v}	Cv5
$\overline{10}$	Ch_{5h}	Ch5
52	D_5	D5
$\overline{5}$m	D_{5d}	Dd5
$\overline{10}$2m	D_{5h}	Dh5
532	I	I
5$\overline{3}$m	I_h	Ih

7.6 Table of univalent covalent radii used in USPEX

Table of covalent radii (in Å) used in USPEX (for hardness calculations, *etc.*):

Z	Element	radius	Z	Element	radius	Z	Element	radius
1	H	0.31	30	Zn	1.22	63	Eu	1.98
2	He	0.28	31	Ga	1.22	64	Gd	1.96
3	Li	1.28	32	Ge	1.20	65	Tb	1.94
4	Be	0.96	33	As	1.19	66	Dy	1.92
5	B	0.84	34	Se	1.20	67	Ho	1.92
6	Csp^3	0.76	35	Br	1.20	68	Er	1.89
	Csp^2	0.73	36	Kr	1.16	69	Tm	1.90
	Csp	0.69	37	Rb	2.20	70	Yb	1.87
7	N	0.71	38	Sr	1.95	71	Lu	1.87
8	O	0.66	39	Y	1.90	72	Hf	1.75
9	F	0.57	40	Zr	1.75	73	Ta	1.70
10	Ne	0.58	41	Nb	1.64	74	W	1.62
11	Na	1.66	42	Mo	1.54	75	Re	1.51
12	Mg	1.41	43	Tc	1.47	76	Os	1.44
13	Al	1.21	44	Ru	1.46	77	Ir	1.41
14	Si	1.11	45	Rh	1.42	78	Pt	1.36
15	P	1.07	46	Pd	1.39	79	Au	1.36
16	S	1.05	47	Ag	1.45	80	Hg	1.32
17	Cl	1.02	48	Cd	1.44	81	Tl	1.45
18	Ar	1.06	49	In	1.42	82	Pb	1.46
19	K	2.03	50	Sn	1.39	83	Bi	1.48
20	Ca	1.76	51	Sb	1.39	84	Po	1.40
21	Sc	1.70	52	Te	1.38	85	At	1.50
22	Ti	1.60	53	I	1.39	86	Rn	1.50
23	V	1.53	54	Xe	1.40	87	Fr	2.60
24	Cr	1.39	55	Cs	2.44	88	Ra	2.21
25	Mn l.s.	1.39	56	Ba	2.15	89	Ac	2.15
	h.s	1.61	57	La	2.07	90	Th	2.06
26	Fe l.s	1.32	58	Ce	2.04	91	Pa	2.00
	h.s.	1.52	59	Pr	2.03	92	U	1.96
27	Co l.s.	1.26	60	Nd	2.01	93	Np	1.90
	h.s.	1.50	61	Pm	1.99	94	Pu	1.87
28	Ni	1.24	62	Sm	1.98	95	Am	1.80
29	Cu	1.32				96	Cm	1.69

Source: Cordero *et al.*, Dalton Trans. 2832-2838, 2008[35].

7.7 Table of default chemical `valences` used in USPEX

Table of chemical `valences` used in USPEX (for hardness calculations, *etc.*):

Z	Element	valence	Z	Element	valence	Z	Element	valence
1	H	1	35	Br	1	69	Tm	3
2	He	0.5	36	Kr	0.5	70	Yb	3
3	Li	1	37	Rb	1	71	Lu	3
4	Be	2	38	Sr	2	72	Hf	4
5	B	3	39	Y	3	73	Ta	5
6	C	4	40	Zr	4	74	W	4
7	N	3	41	Nb	5	75	Re	4
8	O	2	42	Mo	4	76	Os	4
9	F	1	43	Tc	4	77	Ir	4
10	Ne	0.5	44	Ru	4	78	Pt	4
11	Na	1	45	Rh	4	79	Au	1
12	Mg	2	46	Pd	4	80	Hg	2
13	Al	3	47	Ag	1	81	Tl	3
14	Si	4	48	Cd	2	82	Pb	4
15	P	3	49	In	3	83	Bi	3
16	S	2	50	Sn	4	84	Po	2
17	Cl	1	51	Sb	3	85	At	1
18	Ar	0.5	52	Te	2	86	Rn	0.5
19	K	1	53	I	1	87	Fr	1
20	Ca	2	54	Xe	0.5	88	Ra	2
21	Sc	3	55	Cs	1	89	Ac	3
22	Ti	4	56	Ba	2	90	Th	4
23	V	4	57	La	3	91	Pa	4
24	Cr	3	58	Ce	4	92	U	4
25	Mn	4	59	Pr	3	93	Np	4
26	Fe	3	60	Nd	3	94	Pu	4
27	Co	3	61	Pm	3	95	Am	4
28	Ni	2	62	Sm	3	96	Cm	4
29	Cu	2	63	Eu	3	97	Bk	4
30	Zn	2	64	Gd	3	98	Cf	4
31	Ga	3	65	Tb	3	99	Es	4
32	Ge	4	66	Dy	3	100	FM	4
33	As	3	67	Ho	3	101	Md	4
34	Se	2	68	Er	3	102	No	4

7.8 Table of default `goodBonds` used in USPEX

Table of default `goodBonds` used in USPEX (for hardness calculations, *etc.*):

Z	Element	goodBonds	Z	Element	goodBonds	Z	Element	goodBonds
1	H	0.20	35	Br	0.10	69	Tm	0.20
2	He	0.05	36	Kr	0.05	70	Yb	0.20
3	Li	0.10	37	Rb	0.05	71	Lu	0.20
4	Be	0.20	38	Sr	0.10	72	Hf	0.30
5	B	0.30	39	Y	0.20	73	Ta	0.40
6	C	0.50	40	Zr	0.30	74	W	0.30
7	N	0.50	41	Nb	0.35	75	Re	0.30
8	O	0.30	42	Mo	0.30	76	Os	0.30
9	F	0.10	43	Tc	0.30	77	Ir	0.30
10	Ne	0.05	44	Ru	0.30	78	Pt	0.30
11	Na	0.05	45	Rh	0.30	79	Au	0.05
12	Mg	0.10	46	Pd	0.30	80	Hg	0.10
13	Al	0.20	47	Ag	0.05	81	Tl	0.20
14	Si	0.30	48	Cd	0.10	82	Pb	0.30
15	P	0.30	49	In	0.20	83	Bi	0.20
16	S	0.20	50	Sn	0.30	84	Po	0.20
17	Cl	0.10	51	Sb	0.20	85	At	0.10
18	Ar	0.05	52	Te	0.20	86	Rn	0.05
19	K	0.05	53	I	0.10	87	Fr	0.05
20	Ca	0.10	54	Xe	0.05	88	Ra	0.10
21	Sc	0.20	55	Cs	0.05	89	Ac	0.20
22	Ti	0.30	56	Ba	0.10	90	Th	0.30
23	V	0.30	57	La	0.20	91	Pa	0.30
24	Cr	0.25	58	Ce	0.30	92	U	0.30
25	Mn	0.30	59	Pr	0.20	93	Np	0.30
26	Fe	0.25	60	Nd	0.20	94	Pu	0.30
27	Co	0.25	61	Pm	0.20	95	Am	0.30
28	Ni	0.15	62	Sm	0.20	96	Cm	0.30
29	Cu	0.10	63	Eu	0.20	97	Bk	0.30
30	Zn	0.10	64	Gd	0.20	98	Cf	0.30
31	Ga	0.25	65	Tb	0.20	99	Es	0.30
32	Ge	0.50	66	Dy	0.20	100	FM	0.30
33	As	0.35	67	Ho	0.20	101	Md	0.30
34	Se	0.20	68	Er	0.20	102	No	0.30

7.9 Examples

The following example folders illustrate the main USPEX calculation types. In a release package, these folders may be placed in the examples directory.

Fixed-composition atomic crystals, `calculationType`=300

- `EX01-3D_Si_vasp/` — Silicon (8 atoms/cell) at zero pressure. Variable-cell DFT calculation using VASP with the PBE96 functional. The folder contains `INPUT.txt`, VASP `Specific/INCAR_*` files, `POTCAR_Si`, a job-submission template, and a `reference/` output snapshot.
- `EX02-3D_Si_mattersim/` — Silicon (8 atoms/cell) at zero pressure. Variable-cell calculation using the MatterSim universal machine-learning interatomic potential (`abinitioCode`=0). This is the compact MatterSim counterpart of the VASP silicon example.
- `EX03-3D_MgAl2O4_mattersim/` — $MgAl_2O_4$ (14 atoms/cell) at zero pressure. Variable-cell fixed-composition calculation using MatterSim.
- `EX04-3D_MgAl2O4_gulp/` — $MgAl_2O_4$ (28 atoms/cell) at 100 GPa. Variable-cell calculation using Buckingham potentials with GULP; the GULP templates are stored in `Specific/ginput_*` and `Specific/goptions_*`.
- `EX05-3D_Al2O3_mattersim/` — Al_2O_3 single-block calculation using MatterSim. The composition block defines one Al_2O_3 formula unit, while `minAt`/`maxAt` allow USPEX to vary the number of formula units in the cell.
- `EX08-USPEX-performance_SrTiO3_gulp/` — $SrTiO_3$ (50 atoms/cell) at zero pressure. Variable-cell calculation using Buckingham potentials with GULP; this larger fixed-composition example is useful for checking performance and success rate.
- `EX13 EX03-3D-const_cell_MgSiO3_gulp/` — $MgSiO_3$ with a fixed orthorhombic cell (20 atoms per cell, mode `300`). The example uses five GULP relaxation stages with the templates in `Specific/`. It requires a USPEX build with the fixed-cell input reader correction described in the example's `README.txt`. The saved `results1/` contains the final structure in `best_structure_POSCAR`.

Variable-composition atomic crystals, `calculationType`=301

- `EX06-3D_Mo-B_mattersim/` — Binary Mo–B variable-composition search using MatterSim. The `numSpecies` block defines Mo and B composition building blocks, while `firstGeneMax`, `minAt`, and `maxAt` control the first generation and allowed cell sizes.

- EX07-3D_varcomp_LJ_gulp/ — Variable-composition Lennard-Jones binary system with fake "Mo" and "B" atoms, relaxed with GULP. This is a lightweight example for convex-hull output and variable-composition analysis.

Molecular crystals, calculationType=310 and 311

- EX09-3D_urea_gulp/ — Urea (2 molecules/cell) at zero pressure. Variable-cell molecular-crystal calculation using the Dreiding force field and GULP. The folder contains INPUT.txt, a manually prepared MOL_1 file, GULP files in Specific/, and a reference/ output snapshot. This is the main example for a GULP molecular-crystal setup.
- EX10-3D_CO2_d3_mattersim/ — Carbon dioxide (4 molecules/cell) at zero pressure. Variable-cell molecular-crystal calculation using the MatterSim universal machine-learning interatomic potential with the D3 dispersion correction. The folder contains INPUT.txt, CO2.xyz, and a reference/ output snapshot. This is the main example for an XYZ-based molecular-crystal setup with abinitioCode=0 and d3=1.
- For molecular variable-composition calculations, use the same molecular-input layout. Set the calculation type to 311 and define a variable-composition numSpecies block for the molecular species. The same folder structure as in the EX09 and EX10 examples applies.

Nanoclusters, calculationType=000 and 002

- EX11 Si12_OMat24_Plus/ — Fixed-composition prediction for an isolated Si_{12} cluster (mode 000). Open results1/best_structure_POSCAR to inspect the lowest-energy structure found.
- EX12 SO_0505/ — Compositional ranges for Si–O clusters (mode 002). Each element ranges from 0 to 5 atoms. The empty composition is excluded. Start with results1/best_clusters to compare the lowest-energy structures found at each exact composition.

Both inputs use DPA4 omat24 plus with four ASE relaxation stages defined in Specific/. The examples include a saved results1/ folder. Before starting a fresh run, adapt numParallelCalcs and the CPU IDs in allowedCPUs to your machine (Section 3.15.2).

Bibliography

[1] J. Maddox. Crystals from first principles. *Nature*, 335:201, 1988.

[2] A.R. Oganov and C.W. Glass. Crystal structure prediction using ab initio evolutionary techniques: Principles and applications. *The Journal of Chemical Physics*, 124:244704, 2006.

[3] C.W. Glass, A.R. Oganov, and N. Hansen. USPEX — evolutionary crystal structure prediction. *Comp. Phys. Comm.*, 175:713–720, 2006.

[4] A.R. Oganov and S. Ono. Theoretical and experimental evidence for a post-perovskite phase of MgSiO3 in Earth's D" layer. *Nature*, 430(6998):445–448, July 2004.

[5] M. Murakami, K. Hirose, K. Kawamura, N. Sata, and Y. Ohishi. Post-perovskite phase transition in MgSiO3. *Science*, 304(5672):855–858, 2004.

[6] A.R. Oganov, J.C. Schon, M. Jansen, S.M. Woodley, W.W. Tipton, and R.G. Hennig. *Appendix: First Blind Test of Inorganic Crystal Structure Prediction Methods*, pages 223–231. Wiley-VCH Verlag GmbH & Co. KGaA, 2010.

[7] C.J. Pickard and R.J. Needs. High-pressure phases of silane. *Phys. Rev. Lett.*, 97:045504, Jul 2006.

[8] M. Martinez-Canales, A.R. Oganov, Y. Ma, Y. Yan, A.O. Lyakhov, and A. Bergara. Novel structures and superconductivity of silane under pressure. *Phys. Rev. Lett.*, 102:087005, Feb 2009.

[9] Y. Ma, A.R. Oganov, Y. Xie, Z. Li, and J. Kotakoski. Novel high pressure structures of polymeric nitrogen. *Phys. Rev. Lett.*, 102:065501, 2009.

[10] C.J. Pickard and R.J. Needs. High-pressure phases of nitrogen. *Phys. Rev. Lett.*, 102:125702, Mar 2009.

[11] G. Gao, A.R. Oganov, P. Li, Z. Li, H. Wang, T. Cui, Y. Ma, A. Bergara, A.O. Lyakhov, T. Iitaka, and G. Zou. High-pressure crystal structures and superconductivity of stannane (SnH4). *Proceedings of the National Academy of Sciences*, 107(4):1317–1320, 2010.

[12] C.J. Pickard and R.J. Needs. Structures at high pressure from random searching. *physica status solidi (b)*, 246(3):536–540, 2009.

[13] G.R. Qian, X. Dong, X.-F. Zhou, Y. Tian, A.R. Oganov, and H.-T. Wang. Variable cell nudged elastic band method for studying solid-solid structural phase transitions. *Computer Physics Communications*, 184(9):2111–2118, 2013.

[14] C. Dellago, P.G. Bolhuis, F.S. Csajka, and D. Chandler. Transition path sampling and the calculation of rate constants. *The Journal of Chemical Physics*, 108(5):1964–1977, 1998.

[15] S.E. Boulfelfel, A.R. Oganov, and S. Leoni. Understanding the nature of "superhard graphite"'. *Scientific Reports*, 2(471):1–9, 2012.

[16] A.R. Oganov and M. Valle. How to quantify energy landscapes of solids. *The Journal of Chemical Physics*, 130:104504, 2009.

[17] A.R. Oganov, A.O. Lyakhov, and M. Valle. How evolutionary crystal structure prediction works — and why. *Accounts of Chemical Research*, 44(3):227–237, 2011.

[18] A.O. Lyakhov, A.R. Oganov, H.T. Stokes, and Q. Zhu. New developments in evolutionary structure prediction algorithm USPEX. *Comp. Phys. Comm.*, 184:1172–1182, 2013.

[19] Han Yang, Chenxi Hu, Yichi Zhou, Xixian Liu, Yu Shi, Jielan Li, Guanzhi Li, Zekun Chen, Shuizhou Chen, Claudio Zeni, Matthew Horton, Robert Pinsler, Andrew Fowler, Daniel Zügner, Tian Xie, Jake Smith, Lixin Sun, Qian Wang, Lingyu Kong, Chang Liu, Hongxia Hao, and Ziheng Lu. Mattersim: A deep learning atomistic model across elements, temperatures and pressures. May 2024.

[20] G. Kresse and J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 6(1):15–50, 1996.

[21] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996.

[22] Frank Neese, Frank Wennmohs, Ute Becker, and Christoph Riplinger. The orca quantum chemistry program package. *The Journal of Chemical Physics*, 152(22):224108, 2020.

[23] Julian D. Gale. Gulp: A computer program for the symmetry-adapted simulation of solids. *Journal of the Chemical Society, Faraday Transactions*, 93(4):629–637, 1997.

[24] Julian D. Gale and Andrew L. Rohl. The general utility lattice program (gulp). *Molecular Simulation*, 29(5):291–341, 2003.

[25] Jonathan E. Moussa and James J. P. Stewart. MOPAC: An open-source semiempirical molecular orbital program. *Journal of Open Source Software*, 11(119):8025, 2026.

[26] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dulak, Jesper Friis, Michael N. Groves, Bjork Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jon Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiotz, Ole Schutt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W. Jacobsen. The atomic simulation environment – a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[27] Scott Fredericks, Kevin Parrish, Dean Sayre, and Qiang Zhu. Pyxtal: A python library for crystal structure generation and symmetry analysis. *Computer Physics Communications*, 261:107810, 2021.

[28] Daniel Levy, Siba Smarak Panigrahi, Sékou-Oumar Kaba, Qiang Zhu, Mikhail Galkin, Santiago Miret, and Siamak Ravanbakhsh. SymmCD: Symmetry-preserving crystal generation with diffusion models. In *AI for Accelerated Materials Design - NeurIPS 2024*, 2024.

[29] Qiang Zhu, Byungkyun Kang, and Kevin Parrish. Symmetry relation database and its application to ferroelectric materials discovery. *MRS Communications*, 12(5):686–691, 2022.

[30] R. Martoňák, A. Laio, M. Bernasconi, C. Ceriani, P. Raiteri, F. Zipoli, and M. Parrinello. Simulation of structural phase transitions by metadynamics. *Z. Krist.*, 220:489–498, 2005.

[31] A.R. Oganov and C.W. Glass. Evolutionary crystal structure prediction as a tool in materials design. *Journal of Physics: Condensed Matter*, 20(6):064210, 2008.

[32] A.O. Lyakhov, A.R. Oganov, and M. Valle. *Crystal Structure Prediction Using Evolutionary Approach*, pages 147–180. Wiley-VCH Verlag GmbH & Co. KGaA, 2010.

[33] W. Zhang, A.R. Oganov, A.F. Goncharov, Q. Zhu, S.E. Boulfelfel, A.O. Lyakhov, E. Stavrou, M. Somayazulu, V.B. Prakapenka, and Z. Konopkova. Unexpected stable stoichiometries of sodium chlorides. *Science*, 342(6165):1502–1505, 2013.

[34] M. Valle. STM3: a chemistry visualization platform. *Z. Krist.*, 220:585–588, 2005.

[35] B. Cordero, V. Gomez, A.E. Platero-Prats, M. Reves, J. Echeverria, E. Cremades, F. Barragan, and S. Alvarez. Covalent radii revisited. *Dalton Trans.*, 21:2832–2838, 2008.