General Utility Lattice Program

Version 1.3

Julian D. Gale
Nanochemistry Research Institute, Curtin University of Technology,
P.O. Box U1987, Perth 6845, Western Australia
email: j.gale@curtin.edu.au
gulp.curtin.edu.au

0.1 Introduction

The General Utility Lattice Program, or GULP to its friends, is designed to perform a variety of tasks relating to three dimensional solids. Although it started life as an attempt to produce an input file driven program for interatomic potential fitting, it has now expanded to encompass energy minimisation, phonon calculations and other hopefully useful facilities. More utilities are constantly being added, often at the request of end users. So if your favourite solid state property isn't currently available then speak up!

In many respects GULP parallels the suite of codes THBREL (METAPOCS), THBFIT, THBPHON, CASCADE and to some extent PARAPOCS. One major difference is that GULP tries to use the crystal symmetry both to make it easier to generate structures and where possible to speed up the calculations by only considering the asymmetric unit.

GULP will now also perform calculations on non-periodic systems subsuming what was once a separate program called CLUSTER. This facility is useful when calculating defect energies for molecular defects. It also allows the combined fitting of potentials to bulk and cluster information.

As with any large computer program (and GULP currently runs to about 160,000 lines) there is always the possibility of bugs. While every attempt is made to ensure that there aren't any and to trap incorrect input there can be no guarantee that a user won't find some way of breaking the program. So it is important to be vigilant and to think about your answers - remember GIGO! Immature optimising compilers can also be a common source of grief. As with most programs, the author accepts no liability for any errors but will attempt to correct any that are reported.

0.2 Overview of program

The following is intended to act as a brief summary of the capabilities of GULP to enable you to decide whether your required task can be performed without having to read the whole manual. Alternatively it may suggest some new possibilities for calculations!

Energy minimisation

- constant pressure / constant volume / unit cell only / isotropic
- thermal/optical calculations
- application of external pressure
- user specification of degrees of freedom for relaxation
- relaxation of spherical region about a given ion or point
- symmetry constrained relaxation
- unconstrained relaxation
- constraints for fractional coordinates and cell strains
- Newton/Raphson, conjugate gradients or Rational Function optimisers
- BFGS or DFP updating of hessian
- search for minima by genetic algorithms with simulated annealing
- free energy minimisation with analytic first derivatives

Transition states

- location of n-th order stationary points
- mode following

Crystal properties

- elastic constants
- bulk modulus
- Youngs modulus
- Poissons ratios
- piezoelectric stress and strain constants
- static dielectric constants
- high frequency dielectric constants
- static refractive indices
- high frequency refractive indices
- phonon frequencies

- phonon densities of states (total and projected)
- phonon dispersion curves
- zero point vibrational energies
- heat capacity (constant volume)
- entropy (constant volume)
- Helmholtz free energy

Defect calculations

- vacancies, interstitials and impurities can be treated
- explicit relaxation of region 1
- implicit relaxation energy for region 2
- energy minimisation and transition state calculations are possible
- defect frequencies can be calculated (assuming no coupling with 2a)

Fitting

- empirical fitting to elastic constants, piezoelectric constants, static and high frequency dielectric constants, lattice energies and structures
- fit to multiple structures simultaneously
- simultaneous relaxation of shell coordinates during fitting
- fit to structures by either minimising gradients or displacements
- variation of potential parameters, charges and core/shell charge splits
- constraints available for fitted parameters
- generate initial parameter sets by the genetic algorithm for subsequent refinement
- fit to quantum mechanically derived energy hypersurfaces

Structure analysis

- calculate bond lengths/distances
- calculate bond angles
- calculate torsion angles
- calculate out of plane distances
- calculation of the density and cell volume
- electrostatic site potentials
- electric field gradients

Structure manipulation

- convert centred cell to primitive form
- creation of supercells

Electronegativity equalisation method

- use EEM to calculate charges for systems containing H, C, N, O, F, Al, Si, P
- use QEq to calculate charges for any element
- new modified scheme for hydrogen within QEq that has correct forces

Generation of input files for other programs

- THBREL/THBPHON/CASCADE (.thb)
- MARVIN (.mvn)
- Insight (.xtl file)
- Insight (.arc/.car files)
- G-Vis (.xr)
- Cerius (.xtl/.cssr)
- SIESTA (.fdf)

Interatomic potentials available

- Buckingham
- Four range Buckingham
- Lennard-Jones (with input as A and B)
- Lennard-Jones (with input in ε and σ format)
- Lennard-Jones (with ESFF combination rules)
- Morse potential (with or without Coulomb subtract)
- Harmonic (with or without Coulomb subtract)
- General potential (Del Re) with energy and gradient shifts
- Spline
- Spring (core-shell)
- Coulomb subtract
- Coulomb with erfc
- Coulomb with short range taper
- Inverse Gaussian

- Damped dispersion (Tang-Toennies)
- Rydberg potential
- Covalent exponential form
- Breathing shell
- Three body potentials - harmonic with or without exponential decay
- Exponential three-body potential
- Urey-Bradley three-body potential
- Stillinger-Weber two- and three-body potentials
- Axilrod-Teller potential
- Four-body torsional potential
- Ryckaert-Bellemans cosine expansion for torsional potential
- Out of plane distance potential
- Embedded atom method for metals (Sutton-Chen potentials and others)
- Two body potentials can be intra- or inter-molecular, or both

Molecular dynamics

- Shell model (dipolar and breathing) molecular dynamics
- Finite mass or adiabatic algorithms
- NVE or NVT (Nose-Hoover) or NPT (Variable cell shape)

0.3 Background

In this section some of the theory behind GULP is explained and references are supplied for those who require a more detailed description of the methods involved.

0.3.1 Lattice energies

The calculation of the energetics of a three-dimensional system theoretically involves the evaluation of interactions between all species, be they cores, shells or united atom units, within the unit cell and their periodic replications to infinity. As this is clearly unfeasible, some finite cutoff must be placed on computation of the interactions. We can decompose the components of the lattice energy into two classes - long- and short-range potentials. These categories can then be treated differently.

The summation of the short-range forces can normally be readily converged directly in real space until the terms become negligible within the desired accuracy. However, other terms may decay slowly with distance, particularly since the number of interactions increases as $4\pi r^2 N_\rho$, where N_ρ is the particle number density. In particular, the electrostatic energy is conditionally convergent since the number of interactions increases more rapidly with distance than the potential (which is proportional to $1/r$) decays. Hence, the two classes of energy components will be considered separately.

Long-range potential

The electrostatic energy is the dominant term for many inorganic materials, particularly oxides, and therefore it is important to evaluate it accurately. For small- to moderate-sized systems this is most efficiently achieved through the Ewald summation [1,2] in which the inverse distance is rewritten as its Laplace transform and then split into two rapidly convergent series, one in reciprocal-space and one in real-space. The distribution of the summation between real- and reciprocal-space is controlled by a parameter η. The resulting expression for the energy is:

$$E_{\text{recip}} = \left(\frac{1}{2}\right)\frac{4\pi}{V}\sum_G \frac{\exp(-\frac{G^2}{4\eta})}{G^2}\sum_i\sum_j q_i q_j \exp(-iG.r_{ij})$$

$$E_{\text{real}} = \left(\frac{1}{2}\right)\sum_i\sum_j \frac{q_i q_j \operatorname{erfc}(\eta^{\frac{1}{2}} r_{ij})}{r_{ij}}$$

where G is a vector of the reciprocal lattice.

These expressions are strictly valid for the case where the material is hypothetically embedded in a metal to ensure that there is no dipole moment overall. This is normally the case, even for materials with an apparently dipolar unit cell as the surfaces will tend to reconstruct to remove the dipole moment. For solids which are genuinely dipolar then there is an additional term which depends on the dipole moment per unit cell. However, as this quantity can only be defined unambiguously when the structure of the entire crystal is known, including the surfaces, this term is not included in GULP.

The Ewald sum has a scaling with system size of $N^{\frac{3}{2}}$. This is achieved when the optimal value of η is chosen. Selection of this value can be made based on the criterion of minimising the total number of terms to be evaluated in real- and reciprocal-space, weighted by the relative computational expense for the operations involved, w :

$$\eta_{\text{opt}} = \left(\frac{nw\pi^3}{V^2}\right)^{\frac{1}{3}}$$

where n is the number of species in the unit cell, including shells, and V is the unit cell volume.

The derivation of the above formula is given by Jackson and Catlow [3], except that the value of w is implicitly assumed to be unity. It generally is found that the parameter, w, which reflects the ratio of the computational expense in reciprocal- and real-space, is not a constant as a function of system size due to implementational factors. In GULP the value of w can be adjusted using the `rspeed` option.

Because the summation of the real-space terms is performed concurrently with the short-range potentials, it can be beneficial to match the real-space cutoff to the short-range cutoff and also to keep it at less than the shortest unit cell vector for moderate to large systems as this leads to greater efficiency in the search for translational image interactions.

The maximum electrostatic cut-offs in real- and reciprocal-space can then be written in terms of the optimum value of η:

$$R_{\max} = \frac{f}{\eta_{\text{opt}}^{\frac{1}{2}}}$$

$$G_{\max} = 2f\eta_{\text{opt}}^{\frac{1}{2}}$$

$$f = (-\ln A)^{\frac{1}{2}}$$

where A is an accuracy parameter which controls the magnitude of terms to be neglected in the Ewald sum. A value of 10^{-8} for A is found to give sufficiently accurate results for most systems, though those with large unit cells may require an increased value.

Recently there has been increasing interest in many techniques which achieve linear or $N \log N$ scaling for the evaluation of the electrostatic contributions, such as the fast multipole method [4] and particle mesh approaches [5]. These methods are clearly beneficial for very large systems, but have a larger prefactor and there is some debate as to where the crossover point with the Ewald sum occurs. The best estimates indicate that this happens at close to 10000 ions. Since GULP is currently aimed at crystalline materials most systems to be studied will be considerably smaller than this and so the Ewald technique represents the most efficient solution.

In the case of finite systems, a simplified implementation of the cell multipole method has been included in GULP to accelerate the calculation of the electrostatic energy. In this implementation the molecular crystal is divided up into a series of boxes each of which has a side equal to the short-range potential cut-off. Hence all interatomic potentials need only be evaluated within a box and between neighbouring boxes - this reduces the expense of searching for valid distances for large systems. The Coulomb terms are also evaluated explicitly within a box and between nearest neighbours. For more remote boxes the charge distribution within each box is expanded as a series of multipoles at the mid point of the atoms within the box. Electrostatic interactions are then calculated from the multipole-multipole terms. The level of multipole can currently be anything up to an octopole.

In the full implementation of the cell multipole method there is a hierarchical structure of boxes on several levels and the size of the box need not be equal to the potential cut-off. The method currently used in GULP is a trade off between complexity and speed-up which is suitable for systems containing of the order of a few thousand species, but for very large systems the full implementation would be beneficial.

The only remaining issue is how to select the charges for the electrostatic energy. For the majority of ionic inorganic materials, particularly oxides and halides, formal charges are a natural choice. Even for materials which are clearly not fully ionic based on the results of *ab initio* electronic structure calculations, such as silicates, formal charges work well in practice provided that a shell model [6] is employed. For low symmetry structures, a dipolar shell model is sufficient to absorb most of the effects of partial covalency, whereas for high symmetry systems a breathing shell (where the shell has a finite variable radius) may be needed in conjunction with formal charges [7].

For molecular crystals, the charges may be determined independently, for example by fitting to a quantum mechanical electrostatic potential energy surface for the isolated species [8] or may be empirically fitted if there is sufficient experimental data for the crystal. An attractive alternative is to use electronegativity equalisation methods [9] to determine the charges *in situ*. This option has been implemented within GULP (see the keyword `eem`).

Interatomic potentials

For many ionic materials the predominant short-range potential description used is the Buckingham potential, which consists of a repulsive exponential and an attractive dispersion term between pairs of species. For more general systems, such as molecular organics, semiconductors, metals and inert gases, a wider range of functional forms is required. GULP contains a variety of standard two-, three- and four-body potentials (Tables 1, 2 and 3, respectively). Additionally, there is the option to input potentials as a series of energies versus distance with a spline function to interpolate between the points. For the Lennard-Jones potential it is possible to input the parameters for each pair of atoms or combination rules can be used based on one-centre coefficients.

In the most commonly used interatomic potentials, the so called 'short-range' cut-off is controlled by the dispersion term as represented by $-C/r^{-6}$, as the exponential repulsion and terms dependant on higher powers of the distance decay more rapidly. Unfortunately, these dispersion terms can often be significant even when summed out to twice the distance needed to converge the repulsive terms; such truncation of the dispersion terms generally leads to small, but noticeable, discontinuities in the energy surface which can lead to termination of an optimisation before the gradient norm falls below the required tolerance.

As pointed out by Williams [10], it is straightforward to accelerate the convergence of the dispersion energy by the same procedure as for the electrostatic energy. When transformed partially into reciprocal space the resulting expressions for the dispersion energy are:

$$E^{C6}_{\text{recip}} = \frac{1}{2}\sum_i\sum_j -C_{ij}\left(\frac{\pi^{\frac{3}{2}}}{12V}\right)\sum_G \exp(iG.r)G^3\left(\pi^{\frac{1}{2}}\text{erfc}\left(\frac{G}{2\eta^{\frac{1}{2}}}\right)+\left(\frac{4\eta^{\frac{3}{2}}}{G^3}-\frac{2\eta^{\frac{1}{2}}}{G}\right)\exp\left(-\frac{G^2}{4\eta}\right)\right)$$

$$E^{C6}_{\text{self}} = \frac{1}{2}\sum_i\sum_j -\frac{C_{ij}}{3}\left((\pi\eta)^{\frac{3}{2}}\right)+\sum_i\frac{C_{ii}\eta^3}{6}$$

$$E^{C6}_{\text{real}} = \frac{1}{2}\sum_i\sum_j\sum_{cells} -\frac{C_{ij}}{r^6}\left(1+\eta r^2+\frac{\eta^2 r^4}{2}\right)\exp(-\eta r^2)$$

The additional computational overhead to perform this summation is small and, when combined with the reduction in the real-space cutoff, the CPU time taken to achieve a particular target accuracy should be greatly diminished. Two algorithms have been implemented, depending on whether all the dispersion C coefficients can be factorised into one centre parameters according to a simple geometric mean combination rule:

$$C_{ij} = (C_iC_j)^{\frac{1}{2}} \qquad \text{for all } i \text{ and } j$$

When such a factorisation can be performed there is a significant increase in efficiency of the calculation in reciprocal-space, since the loop over i and j can be transformed into a single sum:

$$\sum_i\sum_j C_{ij}\exp(iGr_{ij}) = \left(\sum_i C_i\exp\left(iG.r_i\right)\right)^2$$

Table 1: Functional forms for two-body interatomic potentials incorporated into GULP (where r represents the distance between two atoms i and j).

Potential Name	Formula	Units for input
Buckingham	$A\exp(-r/\rho) - Cr^{-6}$	A in eV, ρ in Å, C in eVÅ^6
Lennard-Jones[†]	$Ar^{-m} - Br^{-n}$	A in eVÅ^m, B in eVÅ^n
	or	
	$\varepsilon(c_1(\frac{\sigma}{r})^m - c_2(\frac{\sigma}{r})^n)$	ε in eV, σ in Å
	$c_1 = (n/(m-n))(m/n)^{(m/(m-n))}$	
	$c_2 = (m/(m-n))(m/n)^{(n/(m-n))}$	
Harmonic[⋆]	$\frac{1}{2}k_2(r-r_0)^2 + \frac{1}{6}k_3(r-r_0)^3 + \frac{1}{12}k_4(r-r_0)^4$	k_2 in eVÅ^{-2}, r_0 in Å k_3 in eVÅ^{-3}, k_4 in eVÅ^{-4}
Morse	$D\{[1-\exp(-a(r-r_0))]^2 - 1\}$	D in eV, a in Å^{-2}, r_0 in Å
Spring (core-shell)	$\frac{1}{2}k_2r^2 + \frac{1}{24}k_4r^4$	k_2 in eVÅ^{-2}, k_4 in eVÅ^{-4}
General	$A\exp(-r/\rho)r^{-m} - Cr^{-n}$	A in eVÅ^m, ρ in Å, C in eVÅ^n
Stillinger-Weber (sw2)	$A\exp(\rho/(r-r_{max}))(Br^{-4} - 1)$	A in eV, ρ in Å, B in Å^4
[†] combination rules permitted		
[⋆] k_3, k_4 are optional		

Cut-offs and molecules

All short-ranged two-, three- and four-bodied potentials have finite cut-offs in real space which must be set by the user in some way. Unless the cut-off chosen is so large that convergence is genuinely achieved then it effectively becomes a parameter of the potential. Hence when publishing new potentials it is good practice to publish the cut-offs. Similarly, if you are trying to reproduce the results of previously published potentials make sure you use the same cut-offs.

The main effect of finite cut-offs is to introduce discontinuities into the energy surface as atoms move across the boundary. Generally speaking, the energy minimisation procedure in GULP is not too sensitive to these because of the use of analytical second derivatives. However, if working with only first derivatives or particularly short cut-offs this can be the reason for a minimisation failing to satisfy the required convergence criteria.

An important difference between GULP and some other programs is that it is perfectly allowable for potentials to overlap, i.e. two or more potentials can act between the same species at the same distance. Hence, there are no resulting restrictions for the cut-offs and complex potential functions can be built by combining several potentials together. Conversely, it is important not to duplicate potentials when not intended.

For some types of potential the cut-offs may correspond to chemical criteria such as bond lengths or they may only need to act between molecules or conversely only within them. In such cases it is best not to use distance cut-offs to achieve the correct effect, but instead

Table 2: Functional forms for three-body potentials incorporated into GULP (where r represents the distance between two atoms i and j, θ_{ijk} represents the angle between the two interatomic vectors i-j and j-k).

Potential Name	Formula	Units for input
Stillinger-Weber (sw3)	$K\exp(\rho/(r_{12}-r_{\max})+\rho/(r_{13}-r_{\max}))(\cos(\theta_{213})-\cos(\theta_0))^2$	K in eV, ρ in Å
Three-body †	$\frac{1}{2}k_2(\theta-\theta_0)^2+\frac{1}{6}k_3(\theta-\theta_0)^3+\frac{1}{12}k_4(\theta-\theta_0)^4$	θ_0 in $^\circ$ k_2 in eVrad^{-2}, k_3 in eVrad^{-3}, k_4 in eVrad^{-4}
Three-body*	$\frac{1}{2}k_2(\theta_{213}-\theta_0)^2\exp(-r_{12}/\rho)\exp(-r_{13}/\rho)$	k_2 in eVrad^{-2}, θ_0 in $^\circ$, ρ in Å
Axilrod-Teller	$K(1+3\cos\theta_{213}\cos\theta_{123}\cos\theta_{132})/(r_{12}r_{13}r_{23})^3$	K in eVÅ^9
Exponential	$A\exp(-r_{12}/\rho)\exp(-r_{13}/\rho)\exp(-r_{23}/rho)$	A in eV, r in Å
Urey-Bradley	$\frac{1}{2}k(r_{23}-r_0)^2$	k in eVÅ^{-2}, r_0 in Å
† harmonic, keyword `three`		
* harmonic + exponential, keyword `three expo`		

Table 3: Functional forms for four-body potentials incorporated into GULP (where ϕ_{ijkl} is the torsional angle between the planes ijk and jkl).

Potential Name	Formula	Units for input
Four-body	$k(1+\cos(n\phi-\phi_0))$	k in eV, ϕ_0 in $^\circ$
Ryckaert-Bellemans	$\sum k_n(\cos\phi)^n$	k_n in eV
Out of plane	$kd**2$	k in eVÅ^{-2}

to use the molecule handling facilities within GULP. There are three keywords which when specified activate the molecule facility within the program - `molecule, molq` and `molmec`. If any of these words are present then a search will be performed to locate any molecules within the structures input. This is done by searching for bonds based on the sum of the covalent radii plus a percentage tolerance factor. For most common compounds the default covalent radii will be sufficient to locate all the bonds - if this is not the case then it is possible for the user to either increase the tolerance factor or to adjust the covalent radii using the `covalent` option from the `element` group of commands.

An alternative scenario is that atoms become bonded which shouldn't be. For example, metal atoms often can become bonded in ionic compounds because the covalent radii is no longer relevant for a positively charged ion. These bonds can be removed either by manually setting the radii of the element to zero or by using the `nobond` option to exclude the formation of certain bond types. Whether the correct molecules have been located or not can be seen from the molecule print out in the output file. The three molecule-based keywords mentioned above differ in what they imply for the treatment of intramolecular electrostatics:

`molecule`	$\Rightarrow$	exclude all Coulomb interactions within the molecule
`molq`	$\Rightarrow$	retain all Coulomb interactions within the molecule
`molmec`	$\Rightarrow$	exclude all Coulomb interactions between atom which are bonded (1-2) or two bonds away (1-3)

The specification of `molmec` does not automatically imply that all potentials will be treated in a molecular mechanics fashion, only the electrostatic terms. Providing one of the above there terms is present then optional words may be added to a potential specification line which control aspects of the potential cut-offs. Below is a list of the words that are available and whether it is necessary to still give any cut-offs on the potential parameter line:

Option	Effect	Cut-offs?
`intra`	only act within a molecule	yes
`inter`	only act between molecules	yes
`bond`	only act between bonded atoms	no
`x12`	do not act between bonded atoms	yes
`x13`	do not act between 1-2 and 1-3 atoms	yes

Although with some options it is necessary to still specify a cut-off for generality, the value may not be important any more. For example, if an O-H potential for water is specified as being intramolecular then as long as the maximum distance cut-off is greater than about 1.0 Å then it doesn't matter particularly what it is. Similarly for a potential which is given as being `x12` then it doesn't matter if the minimum cut-off distance is zero - the potential won't act between bonded atoms.

By default, GULP dynamically calculates the molecular connectivity during a calculation. The reason for this is that it ensures that the restart file will yield the same answer as the point in the calculation where it left off. However, sometimes difficulties occur because a bond becomes too long and the molecule breaks into two. When this happens GULP will stop with an error message as this often indicates that the potential model is not working well for the system under study. If the user wants to proceed regardless then there is a keyword `fix` which tells the program to fix the connectivity as that at the starting geometry and not to update it. This means that the program will never stop with this error, but it does mean that a restart may not give the same answer as the initial run if atoms have moved too far.

In the case of ionic materials where the user would like to try to remove some of the numerical problems associated with cut-offs then there are some other options. The normal way of doing this is with a cut-and-shifted potential. In this approach the potential is forced to go to zero at the cut-off by adding a constant to the energy. This makes the energy continuous, but the gradient still has a discontinuity. Again this can be resolved by adding a second term which shifts the gradient to be zero at the cut-off. In GULP this takes the form of a linear term in the distance which, provided the cut-off isn't very short, will have minimal effect in the region of the potential minimum. These corrections are activated using the potential options `energy` or `gradient` after the potential type, but are only currently applicable to certain two-body potentials where it is appropriate. It should be noted that some potential functions go to zero by construction at the cut-off, for example the Stillinger-Weber two- and three-body potentials.

Combination rules

When using Lennard-Jones potentials it is common to use combination rules to determine the interaction parameters between two species. This means that the parameters for the interaction are determined from one-centre only parameters by some form of averaging. The main advantage of this approach is that it reduces the number of parameters to be determined and aids transferability of potentials. Conversely, the resulting potentials may not be as accurate for any one given system. There are two types of combination rule used, depending on whether the potential is being used in the ε/σ or A/B format (see Table 2 for details). If the potentials are being used in the A/B form then the average is taken using a geometric mean:

$$A_{ij} = \sqrt{A_i A_j}$$

$$B_{ij} = \sqrt{B_i B_j}$$

However, if the ε/σ form is being employed then a more complex relationship is needed:

$$\varepsilon_{ij} = \frac{2(\varepsilon_i \varepsilon_j)^{\frac{1}{2}}(\sigma_i^3 \sigma_j^3)}{(\sigma_i^6 + \sigma_j^6)}$$

$$\sigma_{ij} = \left(\frac{\sigma_i^6 + \sigma_j^6}{2}\right)^{\frac{1}{6}}$$

Within GULP it is possible to specify the parameters by species, rather than by pairs of species, using the `atomab` or `epsilon` options. If the word `combine` is added to the specification of a `lennard`-type potential then the parameters can be omitted from the input and they will be generated using the appropriate combination rules. In turn this makes it possible to fit potentials based on combination rules without having to do this via a series of constraints.

Mean field theory

One of the biggest problems that can face someone attempting to simulate complex materials is the fact that often they can be partly disordered or involve partial occupancies of sites. One approach to treating such systems is to generate a supercell so that lots of permutations can be examined. However, the number of possibilities is usually too large to examine each one individually to locate the most stable configuration. Furthermore, this process may alter the symmetry of crystal. Fitting potentials to such structures also becomes rather difficult.

An alternative approach to handling partial occupancies is to use mean field theory. The effect of this is that each site experiences a potential which is the mean of all possible configurations on the disordered positions. In doing so we are assuming that all possible configurations are equally as likely, i.e. the less stable configurations are equally as likely as the more stable ones. This may apply to materials were there is little energetic difference between configurations or to ones which were formed under kinetic rather than thermodynamic control and haven't had chance to achieve a Boltzmann distribution. It must be decided for any given material whether it is therefore appropriate to use this approach.

The practical upshot of the mean field method is that all interactions just become scaled by the site occupancies of both atoms. This has been implemented in GULP such that the user can specify the site occupancy in addition to the coordinates (see the later section on the input for further details) and the program will automatically handle most aspects of the mean field approach. This includes ensuring the total occupancy on a site does not exceed unity and where two different ions share a site with partial occupancy they are constrained to move as a single ion in optimisations.

One important word of warning - it is important that the user thinks through interactions carefully when using the partial occupancy feature to ensure that everything is handled properly. The biggest danger comes in systems where there are two partially occupied sites very close to each other such that in the real system their occupancy would be mutually exclusive. When this happens it is often necessary to specifically exclude potentials between these atoms to obtain the correct behaviour.

Algorithms for energy and derivative evaluations

GULP actually contains several different algorithms for calculating the energy and its first and second derivatives. By default the program will try to choose the most efficient for any given system, excluding possibilities such as the cell multipole method which would actually lead to slight changes in the answer. Normally the user will need to know nothing about what algorithm is being used, so this section is really for the curious.

Usually real-space interactions are calculated in a lower-half triangular fashion to avoid double counting of interactions which would give rise to loops of the form shown below:

```
do $i$ = 2, $numat$
    do $j$ = 1, $i$-1
        [Calculate interaction between $i$ and $j$]
    enddo
enddo
```

If there is the possibility of self-terms or interactions between periodic replications of the same atom then the $i = j$ term would not be excluded, though it may be more efficient to handle this case in a separate loop. For solids where there is significant space group symmetry then a different algorithm may be more efficient:

```
do $i$ = 1, $nasym$
    do $j$ = 1, $numat$
        [Calculate interaction between $i$ and $j$]
    enddo
enddo
```

where *nasym* is the number of species in the asymmetric unit and *numat* is the number of atoms in the full unit cell. Both the symmetry adapted and standard algorithms are present in GULP with selection being made based on the amount of symmetry in the crystal. The use of symmetry can result in up to an order of magnitude speed-up in favourable cases and therefore is well worth using. More details concerning the use of symmetry, in particular with respect to the calculation of derivatives, can be found elsewhere [11].

The second algorithmic aspect to mention applies to the situation when a constant volume optimisation is being performed and some atoms are held fixed. Typical cases where this occurs are in an optical calculation, in which only shells are relaxed, or where a molecule is docked within a rigid microporous material. In this case the energy of interaction between certain atoms is a constant term and the forces on them are ignored. When this happens these atoms are excluded or frozen out of the energy calculation after the first point to save computational expense.

0.3.2 Energy minimisation

By far the most commonly performed task for GULP will be energy minimisation as this is normally a prerequisite for most other types of calculation, such as lattice properties and phonons, if meaningful results are required. In this section we will actually be concerned with the search for general stationary points at which the gradients are zero. Hence we will technically cover both energy minimisation and maximisation!

All stationary points by definition must have zero gradients for all atoms, or as close as possible within the numerical limits of the method being used to calculate them. Of equal importance is the second derivative or hessian matrix at the stationary point. The hessian matrix may be diagonalised to obtain eigenvalues and eigenvectors which physically represents a mapping to a different set of geometrical variables which involve combinations of individual atomic coordinates.

The nature of a stationary point can be characterised by the number of imaginary eigenvalues for the hessian. A true minimum should possess all real eigenvalues, in which case the hessian is said to be positive definite. A stationary point with N imaginary modes can be described as an Nth order transition state. Of greatest importance is the first order transition state which represents the lowest energy pathway between two minima.

It is important to remember that an energy minimum could be just a local minimum and is not necessary the global energy minimum. Often we are deliberately seeking a local minimum. If every time we energy minimised the structure of a zeolite we obtained α-quartz this would be very clever, but very annoying! To search for the global minimum is in general very difficult for a system of any complexity. Genetic algorithms offer one possible method within GULP.

First let us consider how to locate the local energy minimum nearest to the initial structure input. The energy about any given point can be expanded as a Taylor series:

$$E(x + dx) = E(x) + E'(x)dx + \frac{1}{2}E''(x)dx^2 + ...$$

where $E'(x)$ is the vector of first derivatives at x and $E''(x)$ is the matrix of second derivatives. Subsequently we shall terminate the Taylor expansion at the second order term, neglecting higher order terms. This is exact for an energy surface which is harmonic, but more normally is only a first approximation.

By differentiating this we can estimate the vector dx from the current point to the energy minimum:

$$dx = -H^{-1}g$$

where $H = E''(x)$ and $g = E'(x)$. For an harmonic energy surface the displacement vector dx would take us to the local energy minimum in one step. In the general case we can use the above procedure iteratively until the minimum is reached as once we are close to the minimum the energy is often fairly close to harmonic. This procedure is referred to as the Newton- Raphson method.

There are, however, two complications. Firstly, the second derivative matrix is much more computationally expensive to calculate exactly than are the gradients and energy. Hence repeated calculation of the second derivatives and inversion of the matrix is undesirable. Secondly, if the hessian is not positive definite then the Newton-Raphson procedure will converge towards a maximum along any imaginary modes instead of the minimum. We shall now address these two aspects.

A large number of methods have evolved in which the inverse hessian is updated between cycles of minimisation based upon the gradient, g, and position, x, vectors from the current and previous cycles. One of the first and most famous methods is that due to Davidon, Fletcher and Powell (DFP) [12]:

$$H_{i+1}^{-1} = H_i^{-1} + \frac{(x_{i+1} - x_i)(x_{i+1} - x_i)}{(x_{i+1} - x_i)(g_{i+1} - g_i)} - \frac{(H_i^{-1}(g_{i+1} - g_i))(H_i^{-1}(g_{i+1} - g_i))}{(g_{i+1} - g_i)H_i^{-1}(g_{i+1} - g_i)}$$

A later improved alternative is that due to Broyden, Fletcher, Goldfarb and Shanno (BFGS) [13] which is the same as above except for an additional term:

$$.... + ((g_{i+1} - g_i)H_i^{-1}(g_{i+1} - g_i))u.u$$

where the vector is defined according to the equation;

$$u = \frac{(x_{i+1} - x_i)}{(x_{i+1} - x_i)(g_{i+1} - g_i)} - \frac{H_i^{-1}(g_{i+1} - g_i)}{(g_{i+1} - g_i)H_i^{-1}(g_{i+1} - g_i)}$$

The methods largely differ in the degree to which retention of positive definiteness of the inverse hessian is guaranteed during updating. GULP offers both schemes, however by default BFGS is chosen.

In theory any positive definite matrix is sufficient to act as a starting point for the above updating schemes. As the minimisation progresses the matrix should tend to the exact inverse second derivative matrix. By default GULP takes the exact inverse second derivative matrix as the starting point. Should the second derivative matrix have a determinant of zero (causing the inversion to fail) then the absolute magnitude of the diagonal elements will be inverted to ensure a reasonable positive definite matrix results. Alternatively, a unit matrix may be specified. This is obviously faster initially as no second derivatives have to be calculated. However, the convergence later is very slow.

As the minimisation proceeds the hessian can be reset either after a fixed number of cycles or when the minimiser decides the approximate hessian is no longer appropriate. Because the formula for the minimisation step is only an approximation it is desirable to perform a line search during each cycle;

$$dx = -\alpha H^{-1}G$$

where α is the parameter which gives the lowest energy along the direction of the search vector. This procedure also guards against an ill-conditioned hessian causing the minimisation to go uphill towards a transition state as would happen in pure Newton-Raphson.

In cases where the second derivatives are very expensive to calculate there are two approaches that can be taken. Firstly, as described above the BFGS method can be used in conjunction with an initial unit matrix. Secondly, the conjugate gradients method is also available which again only uses gradients and is based on an updating scheme. The difference is that the latter does not require the use of the hessian at all and thus formally requires less array space.

After all minimisations it is important to check that a minimum has indeed been reached by characterising the hessian. There is a method of minimisation, called Rational Function

Optimisation (RFO) or eigenvector following [14], which in theory guarantees that a true minimum is obtained within the parameter space specified subject to the condition that there is a gradient component in any imaginary directions. In this approach the hessian is diagonalised at each step to obtain the eigenvalues and eigenvectors of the hessian. If the hessian has the wrong number of imaginary eigenvalues then a 'level shift' is effectively added to obtain the correct number. By repeating this procedure the system will evolve either up or down hill until a stationary point of the correct nature is located. In this way it is possible to locate transition states as well as minima.

When searching for saddle points it is not necessary to follow the softest eigenvalue uphill. A particular mode can be selected for eigenvector following at the start and the procedure will select the mode at each step which maximises the overlap with the mode from the previous step. It is important to restrict the maximum step size during an RFO optimisation as too large a step can lead to a region with a different curvature.

If the RFO approach guarantees to find a minimum of the correct curvature it may be wondered why this is not used as the default minimiser all the time. The reason is primarily because it is more expensive than the BFGS approach because of determining the eigenvectors of the hessian and due to more frequent exact calculation of the hessian. The best approach to minimisation is to use BFGS at the start when the gradients are high (or even conjugate gradients in some cases) and then to switch to the RFO minimiser when the gradient norm falls below a certain tolerance (this can be achieved using the `switch` option).

In many cases, the exact second derivative matrix is not needed at the start of an optimisation as the system may be in a non-quadratic region of the potential energy surface. The hessian can then be started as a unit matrix and updated subsequently using the BFGS procedure, with a switch to the exact hessian occurring once the gradient has dropped below some threshold value. When running very large systems it is necessary to use conjugate gradients [13] instead of a hessian based technique as the memory requirements for storing even a lower half triangular second derivative matrix become prohibitive and matrix operations start to dominate the computational expense of the calculations.

There are several possible choices of minimisation variable for optimisation. In GULP the minimisation uses fractional coordinates for the atomic positions and strain for the unit cell vectors. Strain is applied by multiplying the cartesian cell vectors by the following matrix:

$$\begin{bmatrix} \varepsilon_1 & \frac{1}{2}\varepsilon_6 & \frac{1}{2}\varepsilon_5 \\ \frac{1}{2}\varepsilon_6 & \varepsilon_2 & \frac{1}{2}\varepsilon_4 \\ \frac{1}{2}\varepsilon_5 & \frac{1}{2}\varepsilon_4 & \varepsilon_3 \end{bmatrix}$$

where the strains ε_1-ε_6 correspond to xx,yy,zz,yz,xz,xy components, respectively. By working with the strains we automatically eliminate the rotational degrees of freedom of the unit cell, as well as having advantages for the calculation of properties as we shall in the next section.

0.3.3 Lattice properties

For an optimised bulk structure it is possible to calculate the second derivatives with respect to both internal and external strains. For this case it is possible to derive a number of properties which are a function of the second derivatives and play an important role in describing the response of the lattice to different types of perturbation. We shall now consider each of these in turn:

Elastic constants

The elastic constant matrix is a 6 $\times$ 6 matrix which contains the second derivatives of the energy density with respect to external strain:

$$c_{ijkl} = \frac{1}{V}(W_{ss} - W_{sc}W_{cc}^{-1}W_{cs})$$

where W_{ss} is the strain-strain second derivative matrix, W_{cc} is the Cartesian space coordinate second derivative matrix, W_{cs} is the mixed Cartesian-strain second derivative matrix, and V is the volume of the unit cell. It is important to note that the elastic constant matrix, in general, depends on the orientation of the unit cell relative to the Cartesian axes. Note that GULP aligns the a cell vector along the x axis, b in the xy plane and the elastic constants are calculated accordingly. When a centred unit cell is converted to its primitive equivalent the orientation of the crystal relative to the axes is preserved.

Dielectric constants

The dielectric constants are calculated both in the high frequency and low frequency, or static, limits. The elements of the 3 x 3 matrices are given by:

$$\epsilon_{\alpha\beta} = \delta_{\alpha\beta} + \frac{4\pi}{V}q^T W_{cc}^{-1}q$$

where q is a vector containing the charges of each species and α, β are the Cartesian directions. For the static dielectric constant matrix the summation runs across all species, including cores and shells, whereas for the high frequency case the sum is only over shells. If there are no shells present in the model then the high frequency dielectric constant matrix is a unit matrix and thus is not printed out.

Piezoelectric constants

There are two variants of piezoelectric constant matrices, piezoelectric stress and piezoelectric strain. The second of these can be obtained from the former by multiplying by the inverse elastic constant matrix. For many materials the piezoelectric constants are zero by symmetry if there is a centre of inversion. The piezoelectric stress constants are derived from the second derivative matrices according to the relationship:

$$P_{\alpha i} = -\frac{4\pi}{V}q^T(W_{cc}^{-1}W_{cs})^{\alpha i}$$

0.3.4 Phonons

Calculation of phonon modes

One of the main properties that can be calculated from the Cartesian second derivative matrix are the vibrational frequencies. These are obtained by diagonalising the so-called dynamic matrix which consists of the mass-weighted Cartesian second derivatives for an isolated cluster or for a solid at the Γ-point:

$$D = m^{-\frac{1}{2}}W_{cc}m^{-\frac{1}{2}}$$

The vibrational frequencies are the square roots of the eigenvalues of the dynamical matrix. Hence, if there are any negative eigenvalues the corresponding vibrational frequencies will be imaginary, thus implying that the system is unstable with respect to a distortion given by the eigenvector of the imaginary mode. In particular, at the Γ-point the first three vibrational frequencies should be equal to zero as they correspond to the translation of the lattice.

The above equation for the dynamical matrix is modified in the case where a shell model is being used as these particles have no mass, yet they must be involved in the second derivatives:

$$D = m^{-\frac{1}{2}}(W_{\mathrm{core-core}} - W_{\mathrm{core-shell}}W^{-1}_{\mathrm{shell-shell}}W_{\mathrm{shell-core}})m^{-\frac{1}{2}}$$

In the case of a periodic solid the vibrational modes become phonons and the dynamical matrix becomes a function of a reciprocal lattice vector k chosen from the Brillouin zone. This means that in constructing $D(k)$ all interactions are multiplied by the phase factor $\exp(ik.r_{ji})$, where r_{ji} is the interatomic vector. A more detailed discussion of the theory of phonons can be found elsewhere [15].

Phonon dispersion curves

If we calculate how the frequencies vary between two points in the Brillouin zone the results are a series of phonon dispersion curves. This procedure is automated within GULP in that the `dispersion` option can be used to calculate the phonons at a number of points between two or more extremes. The resolution of the curves obviously depends on how many points are used along the pathway.

Phonon density of states

We may also be interested in the phonon density of states for a solid as the number of frequencies versus frequency value becomes a continuous function when integrated across the Brillouin zone. While full analytical integration across the Brillouin zone is not readily carried out, this integral can be approximated by a numerical integration. We can imagine calculating the phonons at a grid of points across the Brillouin zone and summing the values at each point multiplied by the appropriate weight (which for a simple regular grid is just the inverse of the number of grid points). As the grid spacing goes to zero the result of this summation tends to towards the true result.

For performing these integrations GULP uses a standard scheme developed by Monkhorst and Pack [16] for choosing the grid points. This is based around three so-called shrinking factors, n_1, n_2 and n_3 - one for each reciprocal lattice vector. These specify the number of uniformly spaced grid points along each direction. The only remaining choice is the offset of the grid relative to the origin. This is chosen so as to maximise the distance of the grid from any special points, such as the gamma point as this gives more rapid convergence.

In many cases it is not necessary to utilise large numbers of points to achieve reasonable accuracy in the integration of properties, such as phonons, across the Brillouin zone. For high symmetry systems several schemes have been devised to reduce the number of points to a minimum by utilising special points in k space. However, because GULP is designed to be general the Monkhorst-Pack scheme is used. The user can input special points instead, if known for the system of interest.

Often it is not necessary to integrate across the full Brillouin zone due to the presence of symmetry. By using the Patterson group (the space group of the reciprocal lattice) GULP reduces the integration region to that of the asymmetric wedge which may only be 1/48-th of the size of the full volume [17].

When producing plots of the phonon density of states the critical factor, apart from the resolution of the integration grid, is the 'box' size. The continuous density of states curve has to be approximated by a series of finite regions of frequency or boxes. Each phonon mode at each point in k space is assigned to the box whose frequency region it falls into.

The smaller the box size the better the resolution of the plot. However, more points will be needed to maintain a smooth variation of number density.

Infra-red phonon intensities

In order to make comparison between theoretically calculated phonon spectra and experiment it is important to know something about the intensity of the vibrational modes. Of course the intensity depends on the technique being used to determine the frequency as different methods have different selection rules. While Raman intensities are not readily calculable from most potential models, due normally to the absence of polarisabilities higher than dipolar ones, approximate values for infra-red spectra can be determined [18]:

$$I_{IR} \propto (\sum_{\text{all species}} qd)^2$$

where q is the charge on each species and d is the Cartesian displacement associated with the normalised eigenvector.

Thermodynamic quantities from phonons

There are a range of quantities that can be readily calculated from the phonon density of states. The accuracy with which they are determined though clearly depends on the k points or shrinking factors selected for the Brillouin zone integration. For systems with large unit cells a small number of k points, perhaps even the Γ-point alone, will be sufficient. However, for those systems with small to medium unit cells it is important to examine how converged the properties calculated are with respect to the grid size.

If a phonon calculation is performed then GULP will automatically print out the relevant thermodynamical quantities. This output depends partly on whether a temperature has been specified for the given structure. If the calculation is set for zero Kelvin then only the zero point energy is output:

$$ZPE = \sum_{k-\text{points}} w_k \sum_{\text{all modes}} \frac{1}{2} h\nu$$

where w_k is the weight associated with the given k point. In principal, the zero point energy should be added to the lattice energy when determining the relative stability of two different structures. However, because the derivatives of the zero point energy are non-trivial it is normally neglected in an energy minimisation.

For temperatures above absolute zero we can calculate the vibrational partition function, which in turn can be readily used to calculate three further properties:
Vibrational partition function:

$$Z_{\text{vib}} = \sum_{k-\text{points}} w_k \sum_{\text{all modes}} \left(1 - \exp\left(-\frac{h\nu}{kT}\right)\right)^{-1}$$

Vibrational entropy:

$$S_{\text{vib}} = R \ln Z_{\text{vib}} + RT \left(\frac{\partial \ln Z_{\text{vib}}}{\partial T}\right)$$

Helmholtz free energy:

$$A = U - TS_{\text{vib}}$$

where

$$U = U_{\text{lattice energy}} + U_{\text{vibrational energy}}$$

Heat capacity at constant volume:

$$C_{\mathrm{v}} = RT\left(2\left(\frac{\partial \ln Z_{\mathrm{vib}}}{\partial T}\right) + T\left(\frac{\partial^2 \ln Z_{\mathrm{vib}}}{\partial T^2}\right)\right)$$

0.3.5 Free energies

Although the most common methods for studying the properties of materials as a function of temperature are molecular dynamics and Monte Carlo simulations, there is an alternative based on static methods within the quasi-harmonic approximation. This is to directly minimise the free energy of the system at a given temperature, where the free energy is calculated from the lattice energy combined with contributions from the phonons including the entropy and zero point energy.

The advantages of working with free energy minimisation are that MD simulations are quite expensive due to the need to reduce the uncertainty by sampling large amounts of phase space. Molecular dynamics and free energy minimisation are in fact complementary techniques. The later approach breaks down at high temperatures as anharmonic effects become important - typically it works at temperatures up to half the melting point as a rough guide. Conversely, molecular dynamics is not strictly valid at low temperatures because the zero point motions and quantum nature of the vibrational levels is ignored.

Although in principle it is possible to analytically fully minimise the free energy of a solid, in practice this is extremely difficult as it requires the fourth derivatives of the energy with respect to the Cartesian coordinates. Hence, a number of approximations are normally made - the main one being that the principal effect of temperature is to expand or contract the unit cell and the effect on internal degrees of freedom is less important.

When changing unit cell parameters we are concerned with the Gibbs free energy as this is appropriate to a constant pressure calculation. This quantity is related to the Helmholtz free energy, whose relationship to the vibrational entropy has already been given previously, by the expression;

$$G = A + PV$$

with

$$P = P_{\mathrm{ext}} - P_{\mathrm{int}}$$

where P is the pressure. The pressure has two components - any external applied pressure plus the internal phonon pressure coming from the vibrations. The phonon pressure is given by:

$$P_{\mathrm{int}} = -\frac{\partial A}{\partial V}$$

In order to calculate the Gibbs free energy it is therefore necessary to calculate the derivative of the Helmholtz free energy with respect to the unit cell volume. This can be done numerically by finite differences. Central differencing is more expensive than using forward differences. However, it is generally necessary to determine the phonon pressure with sufficient accuracy. In turn each calculation of the Helmholtz free energy requires a constant volume minimisation for the given set of unit cell parameters, followed by a phonon calculation.

Once the Gibbs free energy has been calculated then the next stage of a free energy minimisation is to isotropically expand or contract the unit cell until the external pressure balances the internal pressure. Having done this then the derivatives of the Gibbs free energy can be evaluated numerically by finite differences and the unit cell optimised with respect to this quantity.

Because of the three levels of optimisation plus phonon calculations involved, free energy minimisations are rather expensive and shouldn't be undertaken lightly! Due to the numerical nature of several of the derivatives it may be necessary for the user to adjust the finite differencing interval for a calculation to work optimally. Also the calculations are very sensitive to the quality of the underlying energy surface. Potentials with short cutoffs, leading to discontinuities, and soft modes can cause difficulties for the method, so always check your model well before starting.

Free energy minimisation can be used in conjunction with fitting to allow a series of structures at different temperatures to be fitted with inclusion of the thermal effects, though again this is an expensive procedure. It is important to note that a free energy minimisation at 0 K is not the same as an ordinary static calculation. This is due to the presence of the zero point energy in the former method.

0.3.6 Defects

The Mott-Littleton method

The calculation of defect energies is more difficult and approximate than the calculation of bulk properties. In theory, a defect can cause very long range perturbations, particularly if it is not charge-neutral. Consequently the user must always check the convergence of the approximations made.

The simplification in the modelling of defects is to divide the crystal that surrounds the defect into three spherical regions known as regions 1, 2a and 2b [19-21]. In region 1 all interactions are treated exactly at an atomistic level and the ions are explicitly allowed to relax in response to the defect. Except in the case of very short-ranged defects it is not generally possible to achieve the desired degree of convergence by increasing region 1 before running out of computer resources. Consequently, in region 2a some allowance is made for the relaxation of ions but in a way that is more economical.

In region 2a the ions are assumed to be situated in an harmonic well and they subsequently respond to the force of the defect accordingly [22]. This approximation is only thus valid for small perturbations and also requires that the bulk lattice has been optimised prior to the defect calculation. For region 2a individual ion displacements are still considered, whereas for region 2b only the implicit polarisation of sub-lattices, rather than specific ions, is considered.

If the vector x represents the positions of ions in region 1, while ζ represents the displacements of ions in region 2a, then the total energy of the system may be written as:

$$E = E_1(x) + E_{12}(x,\zeta) + E_2(\zeta)$$

where E_1 and E_2 are the energies of regions 1 and 2 respectively, and E_{12} is the energy of interaction between them. We now assume that the energy of region 2 is a quadratic function of the displacements:

$$E_2(\zeta) = \frac{1}{2}\zeta^T W \zeta$$

We also know that we wish to obtain the displacements in region 2 for which the energy is a minimum:

$$\frac{\partial E}{\partial \zeta} = 0 = \frac{\partial E_{12}(x,\zeta)}{\partial \zeta} + W\zeta$$

This expression can be used to eliminate E_2 from the total energy, leaving it purely in terms of E_1 and E_{12}:

$$E = E_1(x) + E_{12}(x,\zeta) - \frac{1}{2}\frac{\partial E_{12}(x,\zeta)}{\partial \zeta}\zeta$$

The displacements in region 2 are formally a function of x for region 1 which makes the minimisation of the total energy with respect to both the positions of region 1 and the displacements of region 2 potentially complicated. This problem can be avoided by using force balance in region 1 as the criteria for convergence (i.e. all forces on ions in region 1 must be zero), rather than purely minimising the energy. The two approaches are equivalent provided that region 2 is at equilibrium also. This will be achieved provided that the displacements in region 2 are small enough that they are genuinely quadratic.

In terms of the minimisation procedure employed for defect calculations the force balance process leads to a slightly different approach to the bulk optimisation. Initially the same Newton-Raphson procedure with BFGS hessian updating and line searches is employed to avoid convergence to stationary points which are not minima. After at least one cycle of the above and when the gradient norm falls below a certain threshold the minimiser abandons the line search procedure and aims purely to reduce the gradients to zero, regardless of the energy. In practice positive changes in the energy near convergence are only ever small.

The defect energy is now the difference in the total energies for the defective and perfect lattice, E_d and E_p respectively, with corrections due to the energy of any interstitial or vacancy species at infinite separation from the lattice, E_∞:

$$E_{\text{defect}} = E_d - E_p + E_\infty$$

Two final aspects must be dealt with in order to obtain the final working equations for the defect energy. Firstly, due to the slow convergence of electrostatic terms in real space alone we cannot evaluate the region 1 - region 2 energy directly. Instead we must calculate the energy of region 1 interacting with the perfect lattice to infinity and then explicitly subtract and add back the terms due to ions which are no longer on their perfect lattice sites. Secondly, because the displacements in region 2 depend on the force acting on a given ion, which in turn is a function of other region 2 ions, there is in fact a linear dependency of the energy on ζ. By suitable manipulation of the energy terms this may be removed to leave the following expression for the defect energy:

$$\begin{aligned} E_{\text{defect}} &= E_{11}(dd) - E_{11}(dp) + E_{1\infty}(dp) - E_{1\infty}(pp) \\ &\quad + E_{12a}(dd) - E_{12a}(dp) + E_{12a}(pp) - E_{12a}(pd) \\ &\quad - \sum \left(\frac{\partial E_{12a}(dd)}{\partial r} - \frac{\partial E_{12a}(pd)}{\partial r} \right) \end{aligned}$$

where the general symbol $E_{ij}(kl)$ denotes the energy of interaction summed over all ions in region i interacting with ions in region j where i and j can be 1, 2a or ∞ (signifying a sum over 1, 2a and 2b out to infinity). The letters k and l indicate whether the energy is for the perfect or defective coordinates in regions i and j respectively, depending on whether they are p or d.

Displacements in region 2a

Expanding the energy as a Taylor series and truncating at second order gives the Newton-Raphson estimate of the vector from the current ion position to the energy minimum position in terms of the force, g, acting on the ion:

$$\zeta = -W^{-1} g$$

Hence if we know the local second derivative matrix and the force acting on the ion we can calculate its displacement. There are a number of possible ways of calculating the force acting on the ions in region 2a. The most common approach is to use the electrostatic force due to only the defect species - i.e. the force due to any interstitial species based on their current positions, less the force due to any vacancies at the position of the original vacant site. In this way region 2a responds to the change in the multipole moments of the defect species in region 1, but not the influence of other forces. Hence for this approximation to strictly hold the distance between any defects and the boundary of region 2a should be greater than the short-range cutoff.

Region 2b energy

Region 2b is assumed to be sufficiently far from the defects that the ions only respond by polarising according to the electrostatic field resulting from the total defect charge placed at the centre of region 1. This can be written for cubic systems as follows:

$$E_{2b} = -\frac{1}{2}Q^2 \sum_{i \neq 1,2a} \frac{q_i m_i}{R_i^4}$$

Because this expression is just dependant on the distance and a couple of lattice site related parameters the region 2b energy can be evaluated using a method analogous to the Ewald sum and then subtracting off the contribution from ions in regions 1 and 2a. An alternative more general, but still not completely general, expression is the following where the lattice site dependant property is now an anisotropic tensor, rather than a scalar [23]:

$$E_{2b} = -\frac{1}{2}Q^2 \sum_{i \neq 1,2a} \sum_{\alpha\beta} \frac{q_i M_i^{\alpha\beta} R_i^{\alpha} R_i^{\beta}}{R_i^6}$$

This can again be calculated by partial reciprocal space transformation based on the second derivatives of the R^{-4} lattice sum.

0.3.7 Fitting

Fundamentals of fitting

Before any production runs can be performed with an interatomic potential program it is necessary to obtain the potential parameters. If you are lucky there may be good parameters for your system of interest already published in the literature so you can just type them in and get going straight away. Unfortunately most people are not so lucky! The fitting facility within GULP [24] allows you to derive interatomic potentials in either of two possible ways. Firstly, you can determine the parameters by fitting to data from some higher quality calculation, such as an *ab initio* one, normally by attempting to reproduce an energy hypersurface. Secondly, you could attempt to derive empirical potentials by trying to reproduce experimental data.

Regardless of which method of fitting you are using the key quantity is the 'sum of squares' which measures how good your fit is. Ideally this should be zero at the end of a fit - in practice this will only happen for trivial cases where the potentials can be guaranteed to completely reproduce the data (for example fitting a Morse potential to a bond length, dissociation energy and frequency for a diatomic should always work perfectly). The sum of squares, F, is defined as follows:

$$F = \sum_{\text{all observables}} w(f_{\text{calc}} - f_{\text{obs}})^2$$

where f_{calc} and f_{obs} are the calculated and observed quantities and w is a weighting factor. There is no such thing as a unique fit as there are an infinite number of possible fits depending on the choice of the weighting factors. The choice of weighting factor for each observable depends on several factors such as the relative magnitude of the quantities and the reliability of the data (for instance a crystal structure will generally be more reliable than an elastic constant measurement).

The aim of a fit is to minimise the sum of squares by varying the potential parameters. There are several standard techniques for solving least squares problems. At the moment GULP uses a Newton-Raphson functional minimisation approach to solving the problem, rather than the more conventional methods. This is because it avoids storing the co-variance matrix. The downside is that near-redundant variables are not eliminated. Currently the minimisation of the sum of squares is performed using numerical first derivatives. The reason for using numerical derivatives is because many of the properties, particularly those derived from second derivatives, are rather difficult to implement analytical derivatives for. Consequently the value of the gradient norm output during fitting should only be taken as a rough guide to convergence.

The choice of which potential parameters to fit belongs to the user and is controlled by a series of flags on the potential input line ($0 \Rightarrow$ fix, $1 \Rightarrow$ vary). There are also options contained within the variables sub-section for allowing more general parameters to fit, such as charge distributions. Note that when fitting charges at least two charges must be varied to have any effect as the program eliminates one variable due the charge neutrality constraint. There is also the option to vary the charge `split` between a core and shell while maintaining a constant overall charge on the ion. The user may also impose their own constraints on fitting variables through the `constrain fit` option.

It is generally recommended that a small number of parameters are fitted initially and the number gradually increased in subsequent restarts. Often if all parameters are allowed to vary from the start unphysical parameters may result. Dispersion terms of Buckingham or Lennard-Jones potentials are particularly prone to poor behaviour during fitting, as they tend to go to zero or become exceedingly large. It is generally recommended that such terms are set equal to a physically sensible value (based on quantum mechanical estimates or polarisability-based formulae) and held fixed until everything else is refined.

A final check that the program looks for is that the total number of variables being fitted is less than the total number of observables!

Fitting energy surfaces

To fit an energy surface it is basically necessary to input all the structures and the energies that correspond to them. To do this it is just a matter of putting one structure after another in the input file (within the limit of the maximum number of structures for which the program is dimensioned). It is possible to fit the gradients acting on the atoms as well the energy of each structure, though often just the energies are fitted. If the latter is the case, then the easiest way to turn off the fitting of the gradients is to specify `noflag` as a keyword to prevent the program for looking for gradient flags in the absence of a keyword to specify them.

Perhaps the only unique feature of fitting an energy surface is the need to include an energy shift in some cases. This is a single additive energy term which is the same for all structures and just moves the energy scale up and down. The justification for this is that often it is impossible to calculate the energy that corresponds exactly to the interatomic potential one from a quantum mechanical calculation [25]. Most commonly this arises where the potential model has partial charges in which case there is an unknown term in the lattice energy due to ionisation potentials and electron affinities for fractions of an electron.

To simplify the specification of this shift value in the input, if you give the `shift` option after the first structure then this value will apply to all subsequent structures until a different value is input. Similarly, its magnitude can be altered by using the `variables` subsection to specify the shift as a variable and this will apply to all structures. It is generally recommended that the shift is fitted first and allowed to fit through out the procedure.

Empirical fitting

An alternative to fitting quantum mechanical data to derive an interatomic potential is to actually fit experimental data. In this case the procedure serves two purposes. Firstly, the degree to which all the data can be reproduced may serve as some guide as to the physical correctness of the model used. Secondly, it provides a means of extrapolation of experimental data for one system to a different one where the data may not be known, or alternatively to unknown properties of the same material.

Any of the properties that can be calculated for the bulk solid or gas phase molecule can also be used in reverse to fit a potential to. Obviously the essential ingredient in the fit is the experimental structure, without which you won't get very far! The conventional way to fit the structure is by requiring that the forces on the atoms are zero. This is clearly not a perfect strategy as it could be satisfied by a transition state rather than a minimum, though in practice it is rare, except when symmetry constraints are imposed.

Normally a good fit requires some second derivative information as well as the structure. For very high symmetry systems, such as rock salt, the structural data alone is completely inadequate. If we imagine a potential as being a binomial expansion about the experimental geometry, then unless the first and second derivatives are reasonably well reproduced by our model then the range of applicability will be almost zero. Typical sources of second derivative information are elastic, dielectric and piezoelectric (where applicable) constants. Also vibrational frequencies contain far more information than any of the above. However, the fitting of frequencies is not straightforward. To fit the frequency magnitudes is certainly possible, however, you have no guarantee that the correct mode has been fitted to the correct eigenvalue. Hence, frequency fitting only tends to be useful from empirical data for special cases, such as O-H stretching modes which are well separated from other modes and for diatomics where there is no problem in assignment!

One other case where frequency fitting can be useful is at the lower end of the spectrum. For an isolated molecule or a solid at its Γ point the first three modes should have zero frequency as they are just translations. In some cases there may be imaginary modes due the potentials not correctly reproducing the true symmetry. Hence by fitting the first three modes to be zero it is possible to encourage the potentials to yield the correct symmetry.

Simultaneous fitting

There is one main difficulty in the conventional scheme for fitting in which the forces on the atoms are minimised by variation of the potential parameters which arises when using a shell model. Normally we don't know what the shell coordinates are at the outset unless the ions are sited at centres of symmetry. In the past people have tried fitting with the shells placed on top of the cores. However, this means that the potentials are tuned to minimise the polarisation in the system and leads to the shell model having only a small beneficial effect. It also doubles the number of observables connected with gradients, but only introduces a small number of extra variables thus making it harder to get a good fit.

The solution to this problem is allow the shell positions to evolve in some way during the fit. There are two possibilities - either we can minimise the shell positions at every point during the fit or we can added the shell coordinates as fitting parameters. In the case where only structural data is being fitted the two methods are equivalent except in the way that they evolve towards the answer. When other properties are included the second approach is not strictly correct, though the difference is usually small.

After experimenting with several test cases it was found that the second scheme in which the shell coordinates become fitted variables was far more stable in convergence and more efficient. Hence this is the scheme that has been adopted and is referred to as 'simultaneous' fitting due to the concurrent fitting of shell positions. Whenever working with shell models it is recommended that the keyword `simultaneous` is added during conventional fitting - it can improve the sum of squares by several orders of magnitude! Not only does this scheme apply to the coordinates of shells, but also to the radii of breathing shells as well.

Relax fitting

It has been observed that sometimes in conventional fitting getting an improved sum of squares doesn't always get you what is considered to be a better quality fit. This is because people often use different criteria to make their judgement to the ones input into the fitting process. In particular they look at the difference between the optimised structural parameters and those from experiment, rather than looking at the forces. The reason why the forces can be lower, but lead to a worse structure is because in a harmonic approximation the displacements in the structure are given by the gradient vector multiplied by the inverse hessian. Hence, if the gradients get smaller but the inverse hessian gets much larger then the situation may get worse.

The solution to this problem is to fit according to the criteria by which the structures are judged - this is what relax fitting does. This means that at every point in the fit the structure is optimised and the displacements of the structural parameters calculated instead of the gradients. In this approach the shell model is naturally handled correctly and so there is no need for simultaneous fitting. The downside is that it is much more expensive in computer time than conventional fitting. Also you can only start a relax fit once you have a reasonable set of potential parameters - i.e. one which will give you a valid minimisation. Hence a conventional fit is often a prerequisite for a relax fit.

There is a further benefit to using relax fitting. In a conventional fit the properties are calculated at the experimental structure normally with non-zero gradients which is not strictly correct. In a relax fit the properties are calculated for the optimised structure where they are valid.

0.3.8 Genetic algorithms

Conventional minimisation techniques based upon methods such as Newton-Raphson are excellent ways of locating local minima. However, they are of limited use in finding global minima. For example, if we know the chemical composition of a compound and its unit cell, but don't know the structure then we would want to locate the most stable arrangement for placing the atoms within the unit cell. To search systematically for a reasonable set of atomic coordinates may take a very long time by hand. Genetic algorithms [26] are a method by which we can search for global minima rather than local minima, though there can never be an guarantee of finding a global minimum. In many respects it resembles Monte Carlo methods for minima searching, though is regarded by some as being more efficient.

The concept behind the method, as the name might suggest, is to carry out a 'natural selection' procedure within the program in the same way that nature does this in real life. We start off with an even numbered sample of randomly chosen configurations. This is our trial set which is allowed to evolve according to a number of principles described below. Before we can do this we need to consider how to represent our data for each configuration. To do this we encode each number as a binary string by dividing the range between the maximum and minimum possible values (for example 1 and 0 for fractional coordinates) into a series of intervals where the number of such intervals is an integer power of 2. Given this data representation the system now evolves according to the following steps:

(a) Reproduction (tournament) - pairs of configurations are chosen at random and the parameters which measure the relative quality of the two are compared (this is the energy for genetic optimisation or the sum of squares for genetic fitting). The best configuration goes forward to the next iteration, except that there is a small probability, which can be set, for the weaker configuration to win the tournament. This process is repeated as many times as there are configurations so that the total number remains constant.

(b) Crossover - a random point is chosen at which to split two binary strings, after which the two segments are swapped over.

(c) Mutation - a random binary digit is switched to simulate genetic mutations. This can help to search for alternative local minima.

The default output from a genetic algorithm run is a given number of the final configurations, where the ones with the best fitness criteria are selected. However, unless the run smoothly progresses to the region of a single minimum it may be more interesting to look at a sample of the best configurations from the entire run. This can be done with GULP using the `best` option.

Genetic algorithms can only locate minima to within the resolution allowed by the discretisation used in the binary representation. Also they are very slow to converge within the region of a minimum. Hence, the genetic algorithm should be used to coarsely locate the regions associated with minima on the global surface, after which conventional Newton-Raphson methods will most efficiently pin-point the precise minimum in each case.

0.3.9 Electronegativity equalisation

Electronegativity equalisation is a rapid method for the calculation of approximate charge distributions. The basic concept behind the approach is that each element has an intrinsic electronegativity plus a term which varies as a linear function of the charge on the site, so that the more positively charged a site becomes, the more its electronegativity increases. In a system at equilibrium the electronegativity of all atoms must be equal otherwise charge will flow to remove the inequality. Hence the charge distribution can be obtained by solving a series of coupled equations involving the electronegativity and the Coulomb interactions between sites. The full details of the method can be found in a number of references [9,27].

The method has been implemented in GULP using two algorithms depending on whether symmetry is used or not. It provides an inexpensive way of obtaining charges for systems for which it has been parameterised, which includes primarily organics and zeolites. Note that many of the parameters have been fitted to reproduce charges from HF/STO-3G calculations which means they may tend to be underestimates.

0.4 Getting started

0.4.1 Running GULP

Under UNIX:
To run GULP on a machine with the UNIX operating system simply type:

```
<directory>gulp < inputfile
```

where `<directory>` is the path name for the location of gulp on your machine, or if the executable is in your current directory or lies in your path then this may be omitted. In this case the output will come to your terminal. If you wish to save it to an output file then type

```
<directory>gulp < inputfile > outputfile
```

You may like to try using one of the example input files (called exampleN, where N is a number) to see what happens! Input may also be typed directly into the program line by line if no input file is specified. Having finished typing all the required input just type 'start' to commence the run.

Under VMS/DCL:
The easiest approach to running GULP on a VAX is to create a file called GULP.COM containing the following:

```
ASSIGN 'P1'.GLP FOR005
ASSIGN 'P1'.OUT FOR006
RUN GULP
DEASSIGN FOR005
DEASSIGN FOR006
```

To run a job then type:

```
@GULP <Inputfile>
```

which will use `<Inputfile>.GLP` as an input file and write `<Inputfile>.OUT` as an output file.

0.4.2 Getting on-line help

To obtain on-line help on GULP type

```
<directory>gulp <CR>
help <CR>
```

A list of all the possible help topics can then be accessed by typing `topics` or alternatively just type the particular keyword or option that you require help on. Only sufficient characters to specify a unique topic are required. To finish with help type `stop` if you wish to exit the program or `quit` if you want to return to interactive use.

If the help command fails to work it means that the path for the location of the file help.txt (which is an ordinary ASCII text file containing all the help information) has not been set at compile time and that the file is not in the present directory either.

An alternative way of accessing help is to generate an HTML file using the gulp2html utility (courtesy of Dr. Jörg-R. Hill) which produces a file help.html which can then be inspected with a suitable browser, such as netscape.

0.4.3 Example input files

With the program you should have received a number of sample input files which illustrate how GULP works for a number of particular run types. They also serve as a test to ensure that the program works correctly on your machine type. Please note that the interatomic potentials should not be taken as correct for general use - some are made up for the purposes of demonstration only! Below is a brief description of what each example file is doing.

Table 4: List of examples provided

example1	optimises the structure of alumina to constant pressure and then calculates the properties at the final point
example2	simultaneous fit of a shell model potential to the structure of α-quartz, followed by an optimisation with the fitted potentials - the general potential is used with energy and gradient shifts for the Si-O instead of the usual Buckingham potential
example3	an electronegativity equalisation calculation is used to derive partial charges for quartz and are then used to calculate the electrostatic potential and electric field gradients at each site - bond lengths are also calculated
example4	simultaneous fit of a shell model potential to La2O3 using an Ewald-style sum to evaluate the C6 terms, followed by an optimisation with the production of a table comparing the initial and final structures at the end
example5	calculation of a phonon dispersion curve for MgO from 0,0,0 to 1/2,1/2,1/2 - note that normally the structure should be optimised first and that although a phonon density of states curve is produced this may not be accurate due to restricted sampling of k space.
example6	calculation of the defect energy for replacing a Mg2+ ion in MgO by a Li+ ion to create a negatively charged defect
example7a	location of the transition state for a magnesium cation migrating to a vacant cation site in MgO in a defect calculation
example7b	this shows an alternative way of obtaining the same result as in 7a by starting the magnesium in a special position and using the resulting symmetry constraints to allow a ordinary minimisation to the saddle point
example8	a molecular defect calculation in which a sulphate anion is removed from BaSO4 - note that the use of the `mole` keyword to Coulomb subtract within the sulphate anion.
example9	an example of how to use a breathing shell model for MgO - including fitting the model, optimising the structure and calculating the properties
example10	optimisation of urea showing how to handle intermolecular potentials
example11	an example of how to map out the potential energy surface for the migration of a sodium cation parallel to the c axis through a crystal of quartz with an aluminium defect using the translate option
example12	optimisation of two structures within the same input file - also illustrates the use of the name option
example13	shows how to use a library to access potentials for an optimisation of corundum
example14	relaxed fit to structure and properties
example15	simple NVE molecular dynamics
example16	example of constant pressure shell model MD
example17	Sutton-Chen calculation for bulk Ni
example18	example of shell model MD in NVT ensemble

example19	shell model MD run for a zeolite with finite mass
example20	shell model MD run for a zeolite with adiabatic algorithm
example21	charged defect optimisation in a supercell
example22	energy surface fit for a molecular crystal
example23	evaluation of the cost function for a particular structure
example24	example of structure prediction for polymorphs of TiO2
example25	free energy minimisation of quartz within the ZSISA approximation

0.5 Guide to input

0.5.1 Format of input files

On the whole it is only necessary to use up to the first four letters of any word, unless this fails to specify a unique word, and the input is not case sensitive as all characters are converted to lower case on being read in.

The first line of the input is the only special line and is referred to as the keyword line. Keywords should all be given on this line. These consist of control words which require no further parameters and generally specify the tasks to be performed by the program. For example a typical keyword line would look like:

```
optimise conp properties phonon
```

or in abbreviated form:

```
opti conp prop phon
```

This combination of words tells GULP to do a constant pressure optimisation and then to calculate the lattice properties and phonons at the optimised geometry. The order of words within the keyword line is not significant.

All subsequent lines can be given in any order unless that line relates to a previous piece of input. Such lines contain 'options' which generally also require the specification of further information. This information can normally follow on the same line or on the subsequent line. For example the pressure to be applied to a structure could be input as either

```
pressure 10.0             or              pressure
                                          10.0
```

In many cases the units may also be specified if you don't wish to use the default:

```
pressure 1000 kbar
```

Any lines beginning with a '#' and anything that follows a '#' part way through a line is treated as a comment and as such is ignored by the program.

When performing runs with multiple structures any structure dependent options are assumed to apply to the last structure given, or the first structure if no structure has yet been specified. Some options should be specified as sub sections of a particular option. For example, `elastic, sdlc, hfdlc, piezo, energy` and `gradients` all are sub sections of the `observables` command and should appear as follows:

```
observables
elastic 2
1 1 54.2
3 3 49.8
hfdlc
1 1 2.9
end
```

Provided there is no ambiguity, GULP will accept these options even if `observables` is omitted, however, it makes the input more readable if the section heading is included.

GULP reads only the first 80 characters on a line in an input file. Should an input line be two long to fit within this limit then the line can be continued on a second or further lines by adding the continuation character '&' to the end of the line.

0.5.2 Atom names

Many parts of the input to GULP require the specification of atom names, be it when giving their coordinates or when specifying potential parameters. The convention adopted in GULP is that an atom should be referred to by its element symbol, optionally followed by a number to distinguish different occurrences of the same element. Numbers between 1 and 999 are valid numbers. Hence examples of valid atom specifiers are `Si`, `Si12, O3` and `H387`. Something like `Si4+` would not be a valid symbol. The reason for using the element symbol is because several calculations use elemental properties such as the mass or covalent radii in dynamical or molecular runs, respectively.

Sometimes it is desirable to label all the atoms in the structure with numbers to identify them, but with the same interatomic potential acting on them. To avoid having to input the potential multiple times for each symbol there is a convention within GULP which it is important to know. Any reference to just an atomic symbol applies to all occurrences of that element, whereas any reference to an atom type with a number only applies to that specific species. For example a Buckingham potential specified as follows:

```
buck
Si core O core 1283.0 0.299 10.66 0.0 12.0
```

would apply to all Si atoms, regardless of whether they are called `Si` or `Si1` etc. However, the following potential:

```
buck
Si1 core O core 1283.0 0.299 10.66 0.0 12.0
```

would only act on `Si1`. It is important to remember this as people have labelled one atom `Si` and the `Si1` in the past and put potentials for both which resulted in twice the potential acting on `Si1` as there should have been. If the potential had been specified as just acting on `Si` then the correct answer would be obtained as it would act on both atoms once.

In addition to the atom label there is optionally a species type specifier which should be one of the following:

`core`	- represents the main part of an atom including all its mass
`shel`	- represents the mass-less component in a shell model
`bcor`	- a core, but with a spherical breathing radius
`bshe`	- a shell, but with a spherical breathing radius

If not given, then `core` is the default type. Note that the `bcor` and `bshe` types only need to be used in the structure specification. There after they can be treated as an ordinary core or shell in the potential specification and the program will select whether the potential should act on the radius or the centre of the species.

0.5.3 Input of structures

The structure for a three-dimensional solid requires the input of three main sets of information - the unit cell, the fractional coordinates and types of the atoms, and finally the space group symmetry. Taking these in order, the unit cell can be input either as the cell parameters:

```
cell
4.212 4.212 4.212 90.0 90.0 90.0
```

or as the cell vectors:

```
vectors
4.212 0.000 0.000
0.000 4.212 0.000
0.000 0.000 4.212
```

Normally it is easiest to use the cell parameter form and this is recommended. The main reason why you might chose to use the cell vectors is because you want to calculate the properties in a non-standard reference frame (given that quantities such as the elastic constants depend on the unit cell orientation relative to the Cartesian frame). If the cell parameters are input, then the a cell vector is aligned along the x axis, the b cell vector in the xy plane and the c cell vector in the general xyz direction.

When GULP transposes a system between the primitive and centred unit cells the orientation of the atoms is preserved so that any properties calculated will be the same regardless of the cell used. It is recommend that the cell parameters be used for input where possible as this ensures that symmetry can be used to accelerate optimisations. Turning now to the internal coordinates of the atoms, these can again be given either in fractional or Cartesian form, though the former is the more natural for a periodic system. Each line of input must contain at least the atom label followed by the coordinates, in which ever units. For example for the case of MgO:

```
fractional
Mg core 0.0 0.0 0.0
O core 0.5 0.5 0.5
```

Note that if the space group symmetry is to be given then it is only necessary to specify the atoms of the asymmetric unit. Furthermore in any cases where a fractional coordinate is a recurring decimal, such as 1/3, then it is necessary to specify this value to six decimal places to be sure of it being recognised correctly as a special position. If we were to include a shell model for oxygen then the input of coordinates would now look like the following:

```
fractional
Mg core 0.0 0.0 0.0
```

```
O core 0.5 0.5 0.5
O shel 0.5 0.5 0.5
```

There is no need to specify the number of atoms to be input or to terminate the section as this is automatically done when the program finds something which is not an element symbol or a special character at the start of a line.

In addition to the coordinates, there are a number of optional parameters which can follow the z coordinate on the line. These are, in order, the charge, the site occupancy (which defaults to 1.0), the ion radius for a breathing shell model (which defaults to 0.0) and 3 flags to identify geometric variables ($1 \Rightarrow$ vary, $0 \Rightarrow$ fix). Note that the flags will only be read if there is no keyword to specify the geometric variables (e.g. `conp` or `conv`). Hence in full the input for MgO could look as follows:

```
fractional
Mg core 0.0 0.0 0.0 2.00000 1.0 0.0 0 0 0
O core 0.5 0.5 0.5 0.86902 1.0 0.0 0 0 0
O shel 0.5 0.5 0.5 -2.86902 1.0 0.0 0 0 0
```

In the case of MgO all the flags can be set to 0 as there are no geometric variables within the unit cell by symmetry.

If we wanted to run a breathing shell calculation for MgO then the input might look like the following for a constant pressure run:

```
fractional
Mg core 0.0 0.0 0.0 2.00000 1.0 0.0
O core 0.5 0.5 0.5 0.86902 1.0 0.0
O bshe 0.5 0.5 0.5 -2.86902 1.0 1.2
```

or for a mean field calculation of the energy of a 40/60 MgO/CaO material:

```
fractional
Mg core 0.0 0.0 0.0 2.00000 0.4 0.0
Ca core 0.0 0.0 0.0 2.00000 0.6 0.0
O core 0.5 0.5 0.5 0.86902 1.0 0.0
O shel 0.5 0.5 0.5 -2.86902 1.0 0.0
```

The space group symmetry can be specified either through the space group number or through the standard Hermann-Mauguin symbol. Again for MgO, either of the following would be valid:

```
space                  or             space
225                                   F M 3 M
```

In general it is better to use the symbol rather than the number as the structure may be in a non-standard setting. The help file contains a full list of the standard symbols for each space group to illustrate how the symbol should be written in the input, though further non-standard settings will be accepted. The `space` option is not compulsory in the input of a structure and if it is absent then GULP will assume that the structure is in P 1 (i.e. no symmetry).

Related to the `space` option is the `origin` option which allows non-standard origins to be handled. The input for this option can take the form of a single integer (1 or 2) if you want to select one of the standard alternative origin settings. Alternatively if three floating point numbers are input then they are taken to be an origin shift in fractional coordinates, or if three integer numbers are input then they are divided by 24 to obtain the shift.

The structural input for a molecular system is just the Cartesian coordinates. Currently the use of point group symmetry is unavailable for isolated systems so there is no equivalent command to `space` for molecules. There is unlikely to be much benefit from the addition of point group symmetry as most molecular calculations are much faster than their solid state analogues.

Multiple structures can be included in the same file by placing one after another, including mixtures of solid and molecular compounds. A useful option for keeping track of different structures is the `name` option. This must precede the structure and allows the user to give a one word name to the compound which will then be used as a label in the output file. Using this the structural input for a file containing both corundum and quartz might look as follows:

```
name corundum
cell
4.7602 4.7602 12.9933 90.0 90.0 120.0
frac
Al core 0.00000 0.0 0.35216
O core 0.30624 0.0 0.00000
space
167
name quartz
cell
4.91485 4.91485 5.40629 90.0 90.0 120.0
frac
Si core 0.4682 0.0000 0.333333
O core 0.4131 0.2661 0.213100
space
152
```

0.5.4 Species / libraries

In the input for the coordinates there was the option to input the species charge for each individual atom in the asymmetric unit or even the full cell. Normally this is unnecessary as all atoms of the same type have the same charge. In this latter case the charges can be assigned by the species option. So for a zeolite structure, for example, where there may be lots of different Si and O sites we could assign charges as follows:

```
species
Si core 4.00000
O core 0.86902
O shel -2.86902
```

The species command can also serve another purpose which is to assign potential library symbols to each atom type. Quite often we may simulate a whole series of materials with a standard set of potentials. Rather than typing them in every time we can call a library. GULP at present comes with two libraries - one for zeolite and aluminophosphate type systems [3,28,29] and one for metal oxides from the work of Bush et al [30]. All we need to do to call these potentials is to assign the potential types to the types in the library files. In the case of bush.lib, there is no need to do anything as the symbols are just the metal element symbols. For the zeolitic materials there is more than one kind of some atom types and so an assignment is needed. Using this our input would look like:

```
species
Si core Si
O core O_O2-
O shel O_O2-
library catlow.lib
```

0.5.5 Input of potentials

The various types of potentials available in GULP have been tabulated earlier and detailed descriptions of the input format for each one can be found in the on-line help. This section will therefore just contain some general pointers as to how to input potentials.

Let us take the example of a Buckingham potential which acts between magnesium cores and oxygen shells with the parameters A=1280.0eV, ρ=0.300Å, C=4.5 eVÅ^6 and acts over the range of 0 to 12 Å. The input for this would look as follows for an optimisation run:

```
buck
Mg core O shel 1280.0 0.3 4.5 0.0 12.0
```

If we want to perform a fitting run then it is also necessary to specify the flags which indicate which parameters are to be variables (1) and which ones are not (0). There is one flag for each potential parameter and the order of the flags matches that of the parameters. Hence a fit in which we want to vary A only would look as follows:

```
buck
Mg core O shel 1280.0 0.3 4.5 0.0 12.0 1 0 0
```

It would not matter if we had put the flags on the end of the line in the input for an optimisation run - they would have just been ignored. For most potential types some parameters are optional and can be omitted, normally when they are zero. There is always a hierarchy to the order of omission of values. For example, for most two-body potentials if one number is missing then this is assumed to be the minimum cut-off radius and this value is zero (as it quite commonly is). For a Buckingham potential, if a second number is omitted then this is assumed to be the C term which again is often zero. If you are going to omit values it is important to remove the flags when not needed from the input as this may confuse matters. If in doubt give all values!

The number of input parameters can also vary according to any options specified after the potential type. For example, if we wanted the above potential to only act between atoms which are bonded then the input would be:

```
buck bond
Mg core O shel 1280.0 0.3 4.5
```

No potential cut-offs are needed as these are set by the fact that the atoms must be bonded. Similarly for a Lennard-Jones potential when given as `lennard combine` then no potential parameters will appear on the input line as these are determined by combination rules.

More than one potential can be specified for each occurrence of the potential type. Hence the following would be perfectly valid:

```
buck
Mg core O shel 1280.0 0.3 4.5 0.0 12.0
Ca core O shel 1420.0 0.3 6.3 0.0 10.0
```

If the input for one potential is too long to fit on one line then it may be continued on to the next line by using the continuation character '&' at the end of the line.

For two-body potentials there is no ambiguity about the order of the atoms as both are equivalent. For some three-body potentials and all four-body potentials it is important to be aware of the convention regarding the order of input. For a three-body potential which has a unique pivot atom, typically at which the angle is measured, then this pivot atom must be given first and then the two terminal atoms in any order. Hence the O-Si-O angle bending term widely used for zeolites is input as:

```
three
Si core O shel O shel 2.09 109.5 1.9 1.9 3.6
```

In the case of four-body terms there is no unique pivot and so the atoms are input in the order which they are connected. A piece of good advice is that three- and four-body terms are often most readily dealt with using connectivity based cut-offs as part of the molecule set of options.

0.5.6 Defects

In this section we shall cover the basic input required to perform a Mott-Littleton calculation for an isolated defect in an otherwise perfect solid, as activated by the presence of the keyword `defect`. The main run type that we will be concerned with for defects is optimisation as we normally wish to obtain the defect energy and structure. We may also wish to locate transition states for the migration of defects - this follows the same approach as an optimisation, but with the keyword `trans` rather than `opti`. There is currently no facility to fit to defect quantities.

The first issue to consider is the bulk calculation that must precede a defect calculation. For a correct calculation the bulk structure must be optimised at least to constant volume otherwise negative defect energies may result from the removal of bulk forces, rather than defect related ones. If you intend to perform several defect calculations and the bulk unit cell is a reasonable size it is sensible to optimise the bulk as a first job and then use the restart file for the defect calculations to avoid wasted effort. There are also other reasons for optimising the bulk separately. Firstly, if you want to perform a transition state run then the `trans` keyword will try to be applied to the bulk as well as the defect with strange results (this can avoid by using the `bulk_noopt` keyword). Secondly, when creating defects it is important to know where the bulk atoms are so that they can be placed correctly.

At the end of the bulk calculation a property evaluation must be performed as the second derivatives and dielectric constants are needed for the response tensors of region 2. This is automatically invoked and there is no need to add the keyword `property`. Again for some materials the calculation of the properties can be expensive. Hence, if multiple defect calculations are to be performed then this step can be minimised by adding the keyword `save` to the first run (which will write out a temporary file fort.44 which contains the quantities needed for future runs in a binary form to save space) and the keyword `restore` to subsequent runs (which will cause them to read in the information from fort.44 rather than re-calculating it).

Having dealt with the preliminaries, we are now ready to consider how to input the details of the defect calculation. Remember, the following commands should appear after the structure to which they refer. Firstly, we need to determine the defect centre around which the regions are based. This is given using the option `centre` (`center` will also work for the benefit of those of you who are American-minded!). The defect centre is normally placed at the same position as the defect, in the case of a single defect site, or at the middle of a series of defects so as to maximise the distance between any defect and the region 1 boundary. Symmetry is not explicitly input by the user for a defect calculation, however, the program will automatically try to search for any simple symmetry elements. These can then be used to accelerate the calculation. In order to do this it requires that the defect centre is chosen so as to maximise the point group symmetry about itself. This is worth keeping in mind when choosing the location of the defect centre. A general feature of the `centre` option, and those which specify the positions of defect species, is that there are a number of alternative methods for specifying the location:

(a) Atom symbol : this is the label for a species within the unit cell. It is best to use a unique specifier (by changing the type number of one atom if necessary) - if there is ambiguity the program will normally chose the first occurrence of the symbol.

```
centre Mg2 core
```

This will place the defect centre at the final bulk position of the Mg2 core.

(b) Atom number : here the position is given by the number of the atom in the asymmetric unit as input.

```
centre 3
```

This will place the defect centre at the final bulk position of the third atom in the asymmetric unit.

(c) Fractional coordinates : here the position is explicitly given in fractional units - the `frac` option in the following command is optional as it is the default:

```
centre frac 0.25 0.25 0.25
```

(d) Cartesian coordinates: here the position is explicitly given in Cartesian coordinates, the origin of the unit cell being at 0,0,0:

```
centre cart 1.5 2.4 0.8
```

(e) Molecule number: this places the defect centre at the middle of the molecule whose number is given (which corresponds to that in the output):

```
centre mol 2
```

Having located the defect centre the next thing we need to do is to specify the region 1 and region 2a radii. This is done with the `size` command:

```
size 4.0 10.0
```

would result in a region 1 radius of 4.0 Å and a region 2a radius of 10.0 Å. It is important to check how sensitive the defect energy is to these values and to increase them until satisfactory convergence is achieved. One way of doing this while minimising computational expense is by using the restart file. If we wanted to restart a defect calculation run with the above radii, but with region 1 increased to 6.0 Å and region 2a to 12.0 Å then we would just need to edit the restart file to contain the new values, plus the old region 1 size (needed for correct restarting) at the end of the line:

```
size 6.0 12.0 4.0
```

We now have specified the regions - next we need to create some defects. There are three options for this:

`vacancy`	-	removes an ion from the structure to infinity
`interstitial`	-	inserts an ion into the structure from infinity
`impurity`	-	replaces one ion with a different one

The last one, an impurity, is obviously just a short-cut combination of the other two. The actual input for each option for specifying the ion(s) involved follows that for `centre` in that the atom label, atom number, fractional or Cartesian coordinates can be used. The molecule number can also be used with the vacancy option, in which case it removes all the atoms of the molecule from the structure (see example8). Note that when molecules are removed and inserted it is important to correct the defect energy for the molecule at infinity, if this is not zero, as this is not done automatically.

An important part of the interstitial and impurity commands is to specify the type of species to be inserted. For example, the impurity command to replace `O2` with `S` would be:

```
impurity S O2
```

The key thing to note is that the inserting species is always specified first. To make life easy shells are handled automatically in most cases. So if both `O2` and `S` were specified as shell model atoms then the above command would remove both the O2 core and shell, and also insert the S core and shell (initially at the same position). It becomes important when introducing species with a type different to any of the bulk species that all the necessary properties of the inserting ion are given in the `species` option otherwise the atom will be assumed to have no charge and no shell.

There is one further option when introducing an interstitial to make life easier. For example, imagine we want to protonate an oxygen (typically in a zeolite framework) to generate a hydroxyl group at O2. This can be readily achieved using the bond option:

```
interstitial H bond O2
```

this will place the H into the structure at the sum of the covalent radii from O2. To determine the direction for the bond the program maximises the angles to any other atoms to which O2 has bonds.

Putting all these keywords together, we shall illustrate how a lithium impurity could be created in magnesium oxide to generate a negatively charged defect. For the purposes of this example we will work with the full unit cell structure as given by the following:

```
opti conp defect
```

```
cell
4.212 4.212 4.212 90.0 90.0 90.0
frac
Mg core 0.0 0.0 0.0
Mg core 0.0 0.5 0.5
Mg1 core 0.5 0.0 0.5
Mg core 0.5 0.5 0.0
O core 0.5 0.5 0.5
O core 0.5 0.0 0.0
O core 0.0 0.5 0.0
O core 0.0 0.0 0.5
species
Mg core 2.0
O core -2.0
Li core 1.0
```

Based on the above basic input (+ interatomic potentials) all the following would be valid ways of creating the defect:

(a)
```
centre Mg1
size 6.0 12.0
vacancy Mg1
interstitial Li 0.5 0.0 0.5
```

(b)
```
centre Mg1
size 6.0 12.0
impurity Li Mg1
```

(c)
```
centre 0.5 0.0 0.5
size 6 12
impurity Li 0.5 0.0 0.5
```

(d)
```
centre 3
size 6 12
impurity Li cart 2.106 0.0 2.106
```

There are more permutations than this, but hopefully it gives you the general idea.

0.5.7 Restarting jobs

Because on the whole GULP doesn't require very much CPU time per step (the exceptions normally being second derivative calculations on large systems) it doesn't maintain a binary dumpfile with all the details for a restart from exactly where it left off. Instead there is the option to dump out a restart file which is just a copy of the original input (slightly rearranged!) but with any coordinates or potential parameters updated as the run progresses. The frequency with which this is written out can be controlled by the user. If the following is specified:

```
dump every 4 gulp.res
```

then a restart file will be written after every 4 cycles. If the '4' is omitted then it defaults to being one - i.e. a dumpfile is written after every cycle. As the cost of writing out this file is normally small compared to the cost of a cycle this is the usual choice. If the `every` option is omitted then the restart file is written just at the end of the run.

0.5.8 Memory management

In order for a computer to run efficiently it is important to try to keep as much of the job in physical memory as possible and to avoid swapping. This is particularly the case when running several jobs on the same machine. Hence we would ideally like to tailor the memory a job uses to be the amount necessary and no more. Unfortunately in Fortran77 there is no mechanism for the dynamic allocation memory which causes problems in this area.

Although there is a fully dynamic version of GULP written in Fortran90 many machines do not yet have fully functioning compilers for this language (some claim they do, but don't in reality!). Also the performance can be significantly lower than for the f77 compiled version.

As a compromise solution, in GULP versions 1.1 and later there is the ability to use partial dynamic memory allocation within the f77 version on some machines (primarily UNIX ones). It turns out that nearly all the memory in GULP is used by only 3 arrays connected to the second derivatives and the hessian. Hence these are dimensioned at run time using a call to malloc where possible. Normally the program will estimate the maximum size that these arrays might need to be for the input file given and use this to work out how much memory will be needed.

The user can intervene and forcibly restrict the size of these large arrays using the `maxone` option (derived from the name of the parameter in the sizes file which normally controls the dimensions). The input for the `maxone` option is an integer number which is the maximum dimension along one side of the array. For a structure containing N species (cores+shells) in the unit cell this quantity normally needs to be $3N + 6$ to accommodate all second derivatives, or 4N+6 in the case of a breathing shell model. If not all species have shells or radii then it may be possible to trim this back a bit. For large systems the use of second derivatives may become impractical due to the memory involved. In such cases optimisations can be performed using either the `unit` (if the hessian is not too large) or `conj` (to invoke conjugate gradients) keywords. When this is the case the user could specify `maxone 1` and effectively remove all the second derivative matrices to save memory.

As a rough guide, on an Silicon Graphics computer an executable for 1000 species and several tens of structures requires less than 6Mb of memory without second derivatives. The same executable statically compiled with second derivatives would need approaching 200Mb.

0.5.9 Summary of keywords

The following is a concise summary of all the valid keywords available in GULP - for more detail consult the on-line help.

Table 5: Valid keywords in GULP

`angle`	calculate valid three body angles
`anneal`	perform simulated annealing
`average`	output average bond lengths
`bond`	calculate valid bond lengths based on covalent radii
`breathe`	only calculate gradients for breathing shell radii
`broaden_dos`	apply Lorenzian broadening to density of states data
`bulk_noopt`	fix bulk structure prior to a defect calculation
`c6`	calculate C6 terms using lattice sum method
`cartesian`	output Cartesian coordinates for initial structure
`cellonly`	only calculate gradients for and optimise cell parameters
`cmm`	calculate cluster electrostatics using cell multipole method
`compare`	produce a table comparing the initial and final geometries
`conjugate`	use conjugate gradients
`conp`	perform constant pressure calculation - cell to vary
`conv`	perform constant volume calculation - hold cell fixed
`cost`	perform cost function calculation
`dcharge`	output the first derivatives of the atomic charges
`defect`	perform a defect calculation after bulk calculation
`dfp`	use Davidon-Fletcher-Powell update rather than BFGS
`dipole`	add the dipole correction energy for the unit cell
`distance`	calculate interatomic distances
`eem`	calculate charges using electronegativity equalisation
`efg`	calculate the electric field gradients
`eigenvectors`	write out eigenvectors for phonons/frequencies
`fit`	perform a fitting run
`fix_molecule`	fix connectivity for molecules at start and do not update
`free_energy`	perform a free energy instead of internal energy calc
`frequency`	calculate defect frequencies
`full`	write out structure as full rather than primitive cell
`gear`	use the Gear fifth order algorithm for molecular dynamics
`genetic`	perform a genetic algorithm run
`global`	after global optimisation, dump out restart file before opt
`gradients`	perform a single point calculation of energy and gradients
`hessian`	output hessian matrix
`hexagonal`	write out structure as hexagonal rather than rhombohedral
`intensity`	calculate IR intensities for phonon/vibrational modes
`isotropic`	allow cell parameters to vary isotropically
`linmin`	output details of line minimisation
`lower_sym`	try to lower the symmetry using imaginary modes
`md`	perform a molecular dynamics run
`minimum_image`	use the minimum image convention during MD
`molecule`	locate molecules and coulomb subtract within them
`molmec`	locate molecules and coulomb subtract 1-2 and 1-3
`molq`	locate molecules for intramolecular potentials only
`noanisotropic`	use isotropic formula for region 2b energy

`nobreathe`	freeze breathing shell radii during optimisation
`nod2sym`	do not use symmetry for second derivatives
`nodpsym`	do not use symmetry for defect potential calculation
`noelectrostatics`	turns off Ewald sum even when charges are present
`noenergy`	do not calculate the energy - useful for debugging datasets
`noexclude`	do not use atom freezing algorithm during optimisation
`nodensity`	do not output density of state plot after phonon calculation
`nodsymmetry`	do not use symmetry during a defect calculation
`nofirst_point`	skip first point in a translate run
`noflags`	no variable flags are present and all variables to be excluded
`nofrequency`	do not print out frequencies at each k point
`nokpoints`	do not print out k point list
`noksymmetry`	do not use Patterson symmetry in Brillouin zone
`nolist_md`	do not use list method for 3- and 4-body terms in MD
`noreal`	exclude all real space two-body interactions
`norecip`	exclude all reciprocal space two-body interactions
`norepulsive`	do not use truncate repulsive terms based on Ewald accuracy
`nosderv`	do not use symmetry for gradient calculations
`nosymmetry`	turn after symmetry once unit cell has been generated
`nozeropt`	exclude zero point energy from free energy calculation
`numdiag`	estimate on-diagonal hessian elements numerically
`operators`	print out listing of symmetry operators used
`optimise`	minimise the energy with respect to geometrical variables
`outcon`	output constraints in restart file
`phonon`	calculate the lattice phonon modes and cluster frequencies
`positive`	force hessian to remain positive on the diagonal
`pot`	calculate the electrostatic potential at the atomic sites
`potgrid`	calculate the electrostatic potential over a grid
`predict`	perform structure prediction calculation
`property`	calculate the bulk lattice properties
`qeq`	calculate charges using the QEq scheme of Rappe and Goddard
`qok`	its OK to run with a non-charge neutral unit cell
`regi_before`	output region 1 coordinates before defect calculation
`relax`	use relax fitting
`restore`	read in defect calculation restart info and skip property calcn
`rfo`	use the rational function optimisation method
`save`	write out defect calculation restart information
`shell`	only calculate gradients for and optimise shell positions
`simultaneous`	simultaneously optimise shells while fitting
`single`	perform a single point calculation of the energy
`static_first`	perform a static optimisation before a free energy one
`torsion`	calculate valid four body torsion angles
`transition_state`	optimise using RFO to first order transition state
`unit`	use a unit matrix as the initial hessian for optimisation
`verlet`	use the Verlet algorithm for molecular dynamics
`zsisa`	use the ZSISA approximation in a free energy minimisation

Groups of keywords by use

(a) Control of calculation type

`angle, bond, defect, distance, eem, efg, fit, free_energy, genetic, gradients,` md, noenergy, optimise, pot, predict, property, phonon, single, torsion, transition_state

(b) Geometric variable specification
breathe, bulk_noopt, cellonly, conp, conv, isotropic, nobreathe, noflags, shell

(c) Energy calculation algorithm
c6, fix_molecule, minimum_image, molecule, molmec, molq, noanisotropic, nod2sym, nodsymmetry, noelectrostatics, noexclude, nofcentral, noksymmetry, nolist_md, noreal, norecip, norepulsive, nosderv, nozeropt

(d) Optimisation method
conjugate, numdiag, positive, rfo, unit

(e) Output control
average, broaden_dos, cartesian, compare, eigenvectors, hessian, intensity, linmin, nodensity, nodpsym, nofirst_point, nofrequency, nokpoints, operators, outcon, regi_before, restore, save

(f) Structure control
full, hexagonal, lower_symmetry, nosymmetry

Summary of options

The following is a concise summary of all the valid options available in GULP - for more detail consult the on-line help.

Table 6: Valid options in GULP

14_scale	specifies the 1-4 scaling factor for molecular mechanics
accuracy	specifies the accuracy of the Ewald summation
atomab	specifies the one-centre A and B terms for combination rules
axilrod-teller	specifies an Axilrod-Teller three-body potential
best	output best N configurations from genetic algorithm run
bcross	specifies a bond-bond cross term three-body potential
both	all subsequent potentials are both inter- and intra-molecular
box	specify box size/number for dispersion and DOS plots
broaden_dos	alters the default DOS broadening factor
bsm	specifies radial parameters for a breathing shell model
buck4	specifies a four range Buckingham potential
buckingham	specifies a Buckingham potential
bulk_modulus	gives the experimental bulk modulus for fitting (cubic only)
cartesian	input crystal structure using Cartesian coordinates
cell	input unit cell as a, b, c, alpha, beta, gamma
centre	specifies location of defect centre
charge	vary specified atomic charges during fitting
cmm	selects cell multipole method for Coulomb terms and order
configurations	controls the number of configurations in genetic algorithm
constrain	input constraints between variables (fitting and optimisation)
contents	specifies the unit cell contents for structure prediction
cost	set the parameters for the cost function

coulomb	specifies a coulomb subtraction potential
covalent	specifies the covalent radii for an element
covexp	specifies the covalent-exponential potential
crossover	specifies the crossover probability in the genetic algorithm
cutd	cutoff for distance calculation
cutp	overall interatomic potential cutoff
cuts	core-shell cutoff distance
cv	specifies the constant volume heat capacity for fitting
damped_dispersion	C6 and C8 potentials with short range damping
deflist	input defect species list for restart
delf	maximum energy change before hessian is recalculated
delta	specifies the differencing interval for numerical gradients
discrete	specifies the discretisation interval for the genetic algorithm
dispersion	produces phonon dispersion curves
dump	write out a dumpfile for restarts
eam_density	specifies the form and parameters for the Embedded Atom density
eam_functional	specifies the functional form of the Embedded Atom Method
elastic	specifies elastic constant values for fitting
electroneg	input new parameters for electronegativity equalisation
element	opens the element parameter options section
energy	specifies the lattice energy for fitting
ensemble	selects either the NVE or NVT ensemble for MD
entropy	specifies the entropy for fitting
epsilon/sigma	specifies the one-centre ε/σ for combination rules
equilibriation	length of equilibriation period in a molecular dynamics run
exponential	specifies an exponential three-body potential
factor	temperature reduction factor for simulated annealing
finite	use finite differences to evaluate numerical derivatives
fractional	input crystal structure using fractional coordinates
ftol	specifies the function tolerance for optimisations
gdcrit	controls change from energy to force balance in defect calculation
general	specifies a general interatomic potential
genetic	general option for genetic algorithm sub-options
gexp	specifies expotential weights for genetic algorithms
gradients	specifies gradients that are to be used in fitting
grid	specifies the grid for genetic algorithms
gtol	specifies the gradient tolerance for optimisations
harmonic	specifies an harmonic potential
hfdlc	specifies high frequency dielectric constants for fitting
hfrefractive	specifies the high frequency refractive index for fitting
ignore	tells the program to ignore input until "erongi" is found
impurity	replace one ion by another for a defect calculation
integrator	specifies MD integrator algorithm to use
intermolecular	all subsequent potentials are intermolecular only
interstitial	insert an interstitial for a defect calculation
intramolecular	all subsequent potentials are intramolecular only
ionic	specifies the ionic radii for an element
iterations	specifies that the number of cycles of shell optimisation in MD
keyword	allows the input of keywords on any line
kpoints	specify explicit k points for phonon calculation
lennard	specifies a Lennard-Jones potential

`library`	specifies a file containing a library of interatomic potentials
`lin3`	specifies parameters for the ESFF linear three-body potential
`line`	maximum number of points in a line minimisation
`lowest_mode`	sets the lowest and highest modes to be included in the free energy
`manybody`	specifies that a manybody potential should act between two atoms
`marvin`	input commands to be passed through to a Marvin run
`mass`	specifies the atomic mass for an element
`maxcyc`	specifies the maximum number of cycles
`maximum`	upper bound for genetic algorithm parameters
`maxone`	limits size of second derivative arrays for dynamic memory
`mdarchive`	specifies the name for an MD archive file
`minimum`	lower bound for genetic algorithm parameters
`mode2a`	allows the user to chose how region 2a is treated
`morse`	specifies a Morse potential
`move_2a_to_1`	after a defect calc region 2a ions are moved to region 1
`murrell-mottram`	species parameters for the Murrell-Mottram 3-body potential
`mutation`	specifies the mutation probability in the genetic algorithm
`name`	give a one-word name to a structure
`nobond`	excludes bond formation between species in molecule run
`observables`	opens the observables option section
`outofplane`	out of plane distance four-body potential
`origin`	gives the origin setting number or explicit origin
`output`	creates dumpfiles for input to other programs
`piezo`	specifies the piezoelectric constants for fitting
`polynomial`	specifies a polynomial potential
`potential`	inputs electrostatic potential at a point for fitting
`potgrid`	specifies a grid of points at which to calculate the potential
`pressure`	specifies the applied external pressure
`production`	controls the length of the production time for MD run
`project_dos`	generate projected densities of states
`qeqiter`	specifies maximum number of iterations for QEq charges
`qeqradius`	sets the radius at which two-centre integrals switch to 1/r
`qeqtol`	tolerance for convergance of QEq charges for H
`qerfc`	specifies that an erfc screened Coulomb terms should be used
`qtaper`	tapers Coulomb term at short range to a constant value
`region_1`	explicit specification of region ions for defect calculation
`reldef`	maps defect region 1 atoms to perfect ones for restarting
`rspeed`	controls the balance between real and reciprocal space
`rtol`	specifies the bond length tolerance for molecule generation
`rydberg`	specifies the parameters for a Rydberg potential
`ryckaert`	specifies a Ryckaert-Bellemans four-body potential
`sample`	controls sampling frequency during an MD run
`scale`	specifies scaling factor for vectors and Cartesian coordinates
`scmaxsearch`	sets maximum search range for many body potentials in FEM
`sdlc`	specifies static dielectric constants for fitting
`seed`	specifies seed for random number generator
`shear_modulus`	specifies the shear modulus for fitting (cubic case only)
`shellmass`	specifies the proportion of atoms mass for the shell in MD
`shift`	adds an offset to the energy for fitting energy hypersurfaces
`shrink`	specify shrinking factors for Brillouin zone integration
`size`	specifies the sizes of regions 1 and 2a for a defect calculation

`spacegroup`	gives either the space group number or symbol
`species`	specifies the charges for all atomic species
`spline`	specifies spline potential and splining data
`split`	vary specified core-shell charge split during fitting
`spring`	specifies core-shell spring constant
`srefractive`	specifies the static refractive index for fitting
`start`	tells the program to ignore the remaining input and to begin
`stepmx`	controls the maximum step during a minimisation
`stop`	tells the program to stop executing immediately
`supercell`	creates a supercell
`sw2`	specifies a Stillinger-Weber two-body potential
`sw3`	specifies a Stillinger-Weber three-body potential
`switch_minim`	changes the minimiser according to a given criteria
`symbol`	changes element symbol from those in eledata
`temperature`	specifies temperature for thermodynamic properties and MD
`three`	specifies a three-body potential
`time`	places a limit on the run time for the job
`timestep`	controls the timestep in a molecular dynamics run
`title`	inputs title lines for a job
`torsion`	specifies a four-body potential
`tournament`	defines the tournament probability for the genetic algorithm
`tpxo`	species two point crossover in genetic algorithms
`translate`	scans a potential energy surface by translating atoms
`tscale`	controls the temperature scaling in a molecular dynamics run
`ttol`	specifies minimum temperature for simulated annealing
`unfreeze`	sets optimisation flags to 1 within a spherical region
`unique`	sets cost function difference for structures to be unique
`update`	sets the maximum number of hessian updates
`urey-bradley`	specifies a Urey-Bradley three-body potential
`vacancy`	creates a vacancy for a defect calculation
`variables`	opens the variables option section
`vectors`	input lattice vectors to define unit cell
`weight`	changes the weights of observables in fitting
`write`	controls the frequency of writing for the MD dumpfile
`xtol`	controls the parameter tolerance in optimisation

0.6 Guide to output

0.6.1 Main output

Hopefully, if you understand what has gone into the input for the calculation the output will be largely self-explanatory! Hence this section will give only a few brief pointers as to the nature of the output. Many pieces of the output are specific to particular options. The following is a guide to the order which things will appear in the output file, which in turn mirrors the order in which runtypes are executed:

Banner	gives the version number of the program
Keywords	the program echoes the keywords it has found in the input with the exception of some debugging keywords which only affect the more verbose pieces of output

Title	if input by the user
Structural output	for each configuration (structure) in turn the program will echo the structural information that was input and any derived quantities in the following order: • Formula for compound (excluding shells) • Number of species in the asymmetric unit • Total number of species in the primitive cell • Dimensionality of system • For 3-D systems only – Symmetry information – Cell vectors for primitive cell – Cell parameters for primitive and full cell – Cell volume • Temperature • Pressure • Coordinates (fractional for 3-D / Cartesian for 0-D), including the site occupancy and charge. Where applicable, coordinates which are free to vary are indicated by an asterisk following them • Molecule listing (if requested) • Geometry analysis output (if requested)
Species output	contains all species/element specific data
Electrostatic accuracy parameter	
Time limit for run	
Interatomic potentials	
Fitting output	fitting involves all structures and precedes all other calculation types so that they can use the optimised parameters
Translate output	as translate performs the runtype for each point along the specified path it precedes the run type output

Runtype output	the output appears for each configuration in the following order subject to the runtype having been requested by the user: • Electronegativity equalisation • Optimisation / energy / gradient calculation - for non-primitive unit cells values are given for the primitive cell unless specified otherwise • Property calculation • Phonon calculations • Electrostatic potential and derivatives • Molecular dynamics • Defect calculation
Timing information	
File output information	

0.6.2 Files for graphical display

Both phonon dispersion and phonon density of states calculations produce information which is suitable for graphic display. Although there is a crude picture generated in the GULP output it is rather limited by the text nature of the output file. The command `output phonon` can be used to dump the data generated by GULP to two files with the extensions '.dens' and '.disp' which can then be exported into a graph plotting program (after suitable modification) to produce proper plots.

0.6.3 Input files for other programs

The `output` option allows the user to generate input files for a number of other programs, both for displaying crystal structures and for other types of run. The file types available are summarised below:

Marvin	(.mvn)	for a surface calculation
THBREL	(.thb)	for a calculation using the program THBREL
xtl	(.xtl)	for input to the Insight graphical interface from Molecular Simulations Inc. Only applicable to solids.
xr	(.xr)	for input to the G-Vis interface from Oxford Materials (modified CSSR file format)
arc	(.arc/.car)	for input to the Insight graphical interface from MSI. Available for bulk, cluster and defect calculations (each type produces a separate file with the type appended to the name). This file can be generated to contain multiple structures for visualisation as a movie.
cssr	(.cssr)	for input into the Cerius^2 interface from MSI and other programs. Available for bulk calculations
fdf	(.fdf)	contains the structural information in a form suitable for SIESTA

Note that when generating input files for other programs there is no guarantee of compatibility due to differences in features or because of changes in format.

0.6.4 Temporary files

Some use is made by GULP of binary scratch files for certain run types. Most are transient files which are removed before the end of a run. The normal reason for their existence is to economise on memory by allowing large arrays to be overlaid. The following is a list of the Fortran channels that may be used and what they are used for (a 'D' in brackets indicates that the file should be automatically deleted before successful completion of a job):

31/32	restart files for a molecular dynamics run
41/42	defect information needed during execution only (D)
44	restart file for a defect calculation to avoid bulk property calcn
48	region 2a displacements if needed for move_2a_to_1 option (D)
51	storage of frequencies for passing between routines (D)
54	storage of transformation matrix when overwriting array (D)
59	projection of phonons needed for project_dos option (D)

0.7 Guide to installation

Prior to compiling the code there is one file which may need to be edited as it contains system specific information though the code will still compile and run if this is not done:

local.F : This file contains two strings which specify where the files eledata and help.txt can be found on your system should they not be present in the execution directory. There is also a default library directory pointer. You need to change the path to point to the directory where these files will reside on your machine. If GULP cannot find the element information file on your machine then the default values will be set.

After this all you need to do on most Unix machines is type "make".

If you wish to run the program in parallel using MPI then you will need to alter the file "getmachine" accordingly. The usual changes would be to add the "-DMPI" option and in some cases change the compiler name (for example to mpif77/mpif90).

0.8 References

[1] P.P. Ewald, Ann. Phys., 64, 253 (1921)
[2] M.P. Tosi, Solid. St. Phys., 16, 1 (1964)
[3] R.A. Jackson and C.R.A. Catlow, Mol. Simul., 1, 207 (1988)
[4] H.G. Petersen, D. Soelvason, J.W. Perram and E.R. Smith, J. Chem. Phys., 101, 8870 (1994)
[5] U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee and L.G. Pedersen, J. Chem. Phys., 103, 8577 (1995)
[6] B.G. Dick and A.W. Overhauser, Phys. Rev., 112, 90 (1958)
[7] U. Schröder, Solid St. Commun., 4, 347 (1966)
[8] B.H. Besler, K.M. Merz Jr. and P.A. Kollman, J. Comp. Chem., 11, 431 (1990)
[9] K.A. van Genechten, W.J. Mortier and P. Geerlings, J. Chem. Phys., 86, 5063 (1987)
[10] D.E. Williams, Cryst. Rev., 2, 3 and 163 (1989)
[11] J.D. Gale, JCS Faraday Trans., 93, (629) (1997)
[12] R. Fletcher, Practical Methods of Optimisation, Wiley, New York (1980)
[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical Recipes. Cambridge University Press, Cambridge, 2nd edn. (1992)
[14] A. Banerjee, N. Adams, J. Simons and R. Shepard, J. Phys. Chem., 89, 52 (1985)
[15] M.T. Dove, Introduction to Lattice Dynamics, Cambridge University Press, Cambridge (1993)
[16] H.J. Monkhorst and J.D. Pack, Phys. Rev. B, 13, 5188 (1976)
[17] R. Ramirez and M.C. Bohm, Int. J. Quant. Chem., 34, 571 (1988)
[18] E. Dowty, Phys. Chem. Miner., 14, 67 (1987)
[19] C.R.A. Catlow, JCS Faraday Trans. 2, 85, 335 (1989)
[20] A.B. Lidiard, JCS Faraday Trans. 2, 85, 341 (1989)
[21] N.F. Mott and M.J. Littleton, Trans. Faraday Soc., 34, 485 (1938)
[22] C.R.A. Catlow and W.C. Mackrodt, Computer Simulation of Solids, Lecture Notes in Physics, (Springer-Verlag, Berlin), 166, Chapter 1 (1982)
[23] C.R.A. Catlow, R. James, W.C. Mackrodt and R.F. Stewart, Phys. Rev. B, 25, 1006 (1982)
[24] J.D. Gale, Phil. Mag. B, 73, 3 (1996)
[25] J.D. Gale, C.R.A. Catlow and W.C. Mackrodt, Model. Simul. Mat. Sci. Eng., 1, 73 (1992)
[26] K. Gallagher, M. Sambridge and G. Drijkoningen, Geophys. Res. Lett., 18, 2177 (1991)
[27] B.G. Baekelandt, W.J. Mortier and R.A. Schoonheydt, in Modelling of Structure and Reactivity in Zeolites, ed. C.R.A. Catlow, Academic Press, London, Chapter 7 (1992)
[28] K.-P. Schrvder, J. Sauer, M. Leslie, C.R.A. Catlow and J.M. Thomas, Chem. Phys. Lett., 188, 320 (1992)
[29] J.D. Gale and N.J. Henson, JCS Faraday Trans., 90, 3175 (1994)
[30] T.S. Bush, J.D. Gale, C.R.A. Catlow and P.D. Battle, J. Mater. Chem., 4, 831 (1994)