Guidance notes on GASP version 6

Stephen A Wells, Bath, May 2017

Introduction – capabilities and what's new.

GASP (Geometric Analysis of Structural Polyhedra) originated as a utility for assessing polyhedral geometry in models of aluminosilicate minerals. It then developed into a modelling engine for flexible polyhedral frameworks, with the goal of producing structures in which polyhedral bonding constraints and steric exclusion are satisfied as far as possible. This led to the discovery of the "flexibility window" phenomenon in zeolites. A flexibility window is a range of cell parameters (i.e. unit cell shapes and sizes) within which the polyhedra of the framework can in principle be made geometrically ideal and regular. Outside the window, distortions are inevitable.

The method at the heart of GASP is the fitting of a geometric "template" to a cluster of atoms. In a tetrahedral group of atoms such as an SiO_4 unit, for example, the template would have ideal tetrahedral geometry and a defined Si-O bond distance. The mismatch between the template and the cluster provides a measure of the structure's distortion from ideal geometry. GASP can also seek to move the atoms and templates together to minimise the mismatch, a process of "geometric relaxation".

In 2015, GASP was substantially rewritten as version 5 in order to include a number of new functions and to improve the user experience by rationalising the input of commands and options. The main functions introduced at this stage were as follows: GASP now has the ability to handle not only regular polyhedra, but also general bonded units including organic moieties. This makes it possible to handle MOFs, ZIFs, zeolites with channel contents and other flexible frameworks. Rigid clusters in organic moieties – representing the effect of delocalised bonding in sp2 systems – are now detected and handled by the bonding system. Rigidity can also be specified through an input bonding file if desired. This is discussed further in Appendix III. The input format has been entirely revised and all commands are now given in a single input file with defined blocks for geometry and bonding information, control parameters, input and output. The geometric relaxation algorithm has been improved in both speed and stability.

In 2016, with the inclusion of new features, GASP has now reached version 6. The main feature introduced in this version is an automated "window search" function. The process of exploring the flexibility window of a structure is now partially automated, substantially reducing the user effort involved in exploring framework flexibility. Smaller changes include the option to scale the steric radii for atoms in a "1-4" bonding relationship, which can be important for flexible molecules; and an "imperfect" option to handle polyhedra with non-ideal geometries. At the code level, the code has now been sensibly divided into a series of modules with headers, with greater use of Object-Oriented Programming approaches, for cleaner code and greater ease of development in future. Finally, the file format for structure input and output has now been standardised on CIF, for greater ease of use.

Citing GASP

The original paper describing the fitting engine underlying GASP is: *Finding best-fit polyhedral rotations with geometric algebra*, Wells, S.A., Dove, M.T., & Tucker, M.G. (2002), JPCM 14(17), 4567-4584. DOI: 10.1088/0953-8984/14/17/327.
The paper defining the "flexibility window" concept is: *The flexibility window in zeolites*, Sartbaeva, A., Wells, S.A., Treacy, M.M.J., & Thorpe, M.F. (2006), Nature Materials **5**(12), 962-965. DOI: 10.1038/nmat1784.
The first publication using version 5 of GASP is: *Flexibility windows in faujasite with explicit water and methanol extra-framework content*, Wells, S.A., Ka Ming Leung, Edwards, P.P & Sartbaeva, A. (2015), Dalton Transactions, DOI: 10.1039/C4DT03150D
A recent review discussing the development and features of GASP is: GASP: software for geometric simulations of flexibility in polyhedral and molecular framework structures, Wells, S.A. and Sartbaeva, A. (2015), Molecular Simulations, DOI: 10.1080/08927022.2015.1032277

Users are encouraged to cite early, often and repeatedly.

Copyright and user agreement

Compilation

GASP is a command-line program which is intended to be run in a terminal window. Throughout this guide it is assumed that the code exists in a Linux-like environment. Under Windows, it is recommended to download and install Cygwin from https://www.cygwin.com/ This will provide a Linux-like terminal environment within which the codes can be compiled and run. It is also possible to compile and run using the native Windows command line environment (`cmd`).

GASP and its associated utility programs are written in C++ as a series of independent files. The source codes are as follows:

```
basic_utils.cpp  *
Vectors.cpp  *
Rotors.cpp  *
Geometry.cpp  *
Structure.cpp  *
GASPcontrol.cpp  *
GASPmain.cpp
cif2xtl.cpp, coordinates2xtl.cpp, xtl2cif.cpp
super.cpp, xtlsuper.cpp
duper.cpp, xtlduper.cpp
shifter.cpp, xtlshifter.cpp
netdr2.cpp
polycomp.cpp polyana.cpp
rotordotcom.cpp
```

The codes marked with an asterisk* have matching header (`.h`) files. Each of the others will generate a matching executable (`.exe`) file on compilation. Compilation is managed by a shell script (`makegasp.sh`). From a Linux or Cygwin terminal window, running "`bash makegasp.sh`" will carry out the following commands. If any user wishes to construct a full Makefile for the `make` utility, they are very welcome to do so! The text of the `makegasp.sh` file is as follows. If your system has a different C++ compiler its name should be given in place of "`g++`" at the start of the file.

```
#!/bin/bash
COMPILER="g++"
echo "Compiling binaries from cpp files"
$COMPILER -c basic_utils.cpp Vectors.cpp Rotors.cpp Geometry.cpp Structure.cpp
GASPcontrol.cpp GASPmain.cpp
$COMPILER -c duper.cpp  super.cpp coordinates2xtl.cpp xtl2cif.cpp polycomp.cpp
shifter.cpp rotordotcom.cpp
$COMPILER -c xtlduper.cpp xtlsuper.cpp cif2xtl.cpp xtlshifter.cpp netdr2.cpp
polyana.cpp
echo "Compiling gasp executable"
$COMPILER -o gasp basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
GASPcontrol.o GASPmain.o
echo "Compiling duper executable"
$COMPILER -o duper basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o duper.o
$COMPILER -o xtlduper basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
xtlduper.o
echo "Compiling super executable"
$COMPILER -o super basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o super.o
$COMPILER -o xtlsuper basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
xtlsuper.o
echo "Compiling coordinates2xtl executable"
$COMPILER -o coordinates2xtl basic_utils.o Vectors.o Rotors.o Geometry.o
Structure.o coordinates2xtl.o
echo "Compiling xtl2cif and cif2xtl executable"
$COMPILER -o xtl2cif basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
xtl2cif.o
$COMPILER -o cif2xtl basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
cif2xtl.o
echo "Compiling polyhedron comparison polycomp and polyhedron analysis polyana
executables."
$COMPILER -o polycomp basic_utils.o Vectors.o Rotors.o polycomp.o
$COMPILER -o polyana basic_utils.o Vectors.o Rotors.o polyana.o
echo "Compiling shifter utility"
$COMPILER -o shifter basic_utils.o Vectors.o Rotors.o Structure.o shifter.o
$COMPILER -o xtlshifter basic_utils.o Vectors.o Rotors.o Structure.o xtlshifter.o
echo "Compiling rotordotcom comparison utility"
$COMPILER -o rotordotcom basic_utils.o Vectors.o rotordotcom.o
echo "Compiling netdr2 comparison utility"
$COMPILER -o netdr2 basic_utils.o Vectors.o Rotors.o Geometry.o Structure.o
netdr2.o
```

The output when the script is run should be as follows:

```
$ bash makegasp.sh
Compiling binaries from cpp files
Compiling gasp executable
Compiling duper executable
Compiling super executable
Compiling coordinates2xtl executable
Compiling xtl2cif and cif2xtl executable
Compiling polyhedron comparison polycomp and polyhedron analysis polyana
executables
Compiling shifter utility
Compiling rotordotcom comparison utility
Compiling netdr2 comparison utility
```

Execution

GASP is run from a command window/terminal. GASP is invoked by the command `<PATH>/gasp.exe` . If run without an argument, it will seek command input from a file `gasp.inp` and will fail if no such file exists. Otherwise an input file can be specified:

```
<PATH>/gasp.exe myinputfilename
```

GASP will read commands from the file (or fail gracefully if no file is found) and parse the input for validity before carrying out commands.

In mixed DOS/UNIX environments, all text files should be converted to UNIX type line endings (using the `dos2unix` utility) before running GASP.

Structure input

GASP now expects a structure to be given as a cell and fractional coordinates in `.cif` format with P1 symmetry (all atoms explicitly present). This should be easy to generate from conventional structure viewers such as Crystalmaker. The option also exists to use the legacy `.xtl` format, which is discussed in Appendix 5. In general **GASP input structures should be quite large**; if any cell edge is less than about ten Angstroms then it is recommended to build a supercell for input. GASP comes with utilities for supercell generation and duplicate atom removal, `super.exe` and `duper.exe`, which may be useful for structure preparation; these are described in Appendix IV. An example of CIF format for GASP is shown below:

```
data_example-fau.cif

_cell_length_a          23.941
_cell_length_b          23.941
_cell_length_c          23.941
_cell_angle_alpha       90
_cell_angle_beta        90
_cell_angle_gamma       90
_symmetry_space_group_name_H-M 'P 1'
_symmetry_Int_Tables_number 1

loop_
_atom_site_type_symbol
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
o o_1   0.00000000   0.89320000   0.10680000
o o_2   0.00000000   0.39320000   0.60680000
o o_3   0.50000000   0.39320000   0.10680000
…… lines omitted ……
si si_574   0.28515000   0.37320000   0.05250000
si si_575   0.78515000   0.37320000   0.55250000
si si_576   0.78515000   0.87320000   0.05250000
```

Some points to note: GASP does not actually read the `_symmetry` entries as it assumes P1 symmetry. In the `_atom_site` loop, only four entries are actually read, specifically the `_type_symbol` and `_fract_x,y,z` entries. Other `_atom_site` entries such as `_label` or `_charge` can be present but will be ignored on readin. Any other `loop_` not containing `_atom_site` entries is ignored entirely.

We note for completeness that GASP makes use of fully periodic boundary conditions throughout.

If transferring files from Windows systems onto Linux/UNIX or Mac OSX, remember to use `dos2unix` or equivalent, to ensure that line endings are read correctly. If `dos2unix` is not available, the command `tr -d '\r' < startingfile > convertedfile` may be effective. This conversion should also be applied when using Cygwin under Windows if files have been edited using Windows utilities such as Notepad or Wordpad.

Control: the command input file

The input file is divided into a number of blocks, each of which begins with a `keyword` and ends with `/keyword` . Only information within a block is parsed by the program; lines outside a block are ignored. The hash symbol `#` can be used to begin a comment; anything following a `#` is ignored. Comments can appear on their own line or after a valid command (leave a space!). Blank lines are ignored. The order of blocks is irrelevant as all input is parsed before any functions are carried out. This section discusses all blocks.

The input is not case sensitive, in the sense that all commands and options can be given in upper case, lower case or mixed. However, filenames are preserved as given, as filenames in a Linux environment are generally case sensitive.

`TITLE` **block – optional**.

If a title block is given, any line within it will be written as the title line in any structure output. If multiple lines are given only the last line is retained. The title block is completely optional and can be omitted.

`ELEMENT` **block – compulsory**.

GASP will halt if no element block is detected. All elements found in the input structure must be named in this block along with their steric radius in Angstroms.

For example `o 1.35` defines an oxygen atom with a 1.35 Angstrom radius as is typical for zeolite studies.

GASP has no periodic table knowledge and so fictitious element labels are permitted. The only exception is for the elements `c` and `n` for which special rigidity rules may apply – see Appendix III.

`POLY` **block – compulsory if no** `BOND` **block is given**.

A poly block contains the geometry of any regular polyhedra in the structure, in the form *shape central-species vertex-species bond-length*. Thus for example the tetrahedra in a silicate are defined by `tet si o 1.61` . GASP will use the information to identify bonded groups and generate bonding templates with ideal geometry. Available types are `tet`rahedron, `oct`ahedron, `tri`angle, `squ`are planar and `bar` .

Examples would be:

```
tet si o 1.61 # silicates
oct ti o 1.9 # perovskite titanate
tri b o 1.4 # b2o3 borate
bar c n 1.2 # cyanide molecule
squ cu o 1.8 # square planar unit
```

In generating bonding, GASP will seek vertex atoms lying within 1.3 times the given bond length of a centre atom. This multiplier is tunable using the `polypad` option.

The input can contain zero or one `poly` block, but must contain at least one of `poly` or `bond`.

Note that the `bar` type defines a diatomic unit. Either of the atoms in a bar can also have other bonded neighbours; thus bar definitions can be used to fix bond lengths without placing restrictions on bond angles.

`BOND` **block – compulsory if no** `poly` **block is given**.

This block contains bonding information for any clusters which do not have regular polyhedral shape. Each line should list a "centre" element followed by all elements to which it might bond as neighbours: for example:

```
Bond # organic molecule typical bonding
c c h o n
n c h
o c h
/bond
```

Typically one line should be given for each species that bonds to **more than one** neighbour. In the above case no line is needed for terminal hydrogens.

By default GASP seeks neighbours within a radius of 2.0 Angstroms. This can be tuned by giving an optional `within` specification, thus:

```
bond within 1.2 #less than the typical value
o h #suitable for water molecules, if you have any.
/bond
```

The input can contain multiple bond blocks so different finding radii can be specified for different clusters. If the input contains no poly blocks then it must contain at least one bond block. **The geometry of bonded clusters is defined by the input structure**, so it is up to the user to provide the desired input geometry. GASP then maintains bonding templates with this geometry for use in a relaxation.

Note that, if desired, the bonding in polyhedral units in the framework can be defined using a bond block instead of a poly block; in this case they are treated as a bonded cluster and GASP will not attempt to regularise their geometry.

`OPTION` **block – optional**

This block contains control and parameter options. The most important of these are `relax`, `smallmove` and `smallmis`.

`relax` is a switch which activates the geometric relaxation function. Relaxation stops by default when the largest move made by any atom is less than `1e-6`. This can be tuned using an option such as `smallmove 1e-7 # really really small move please`.

It is also possible to add a mismatch criterion such as `smallmis 1e-3 #stop when polyhedra are good`. If this criterion is given then relaxation will stop when ***either*** `smallmove` or `smallmis` is satisfied. This criterion is vital to identify a “flexibility window” (see below).

The `jiggle` option followed by a distance in Angstroms will cause all atomic positions to be perturbed by this distance in a random direction before geometric relaxation takes place.
So for example `jiggle 0.1` will cause random displacements by 0.1 Angstroms. This can occasionally be useful as a symmetry-breaking step before relaxation proceeds.

The `search` option can be used to trigger an automated series of relaxations, during which GASP will vary the cell parameters to find the limits of the “flexibility window” for a structure; this concept is discussed further below. The search option is followed by a keyword which controls which cell parameters are varied.

The `gradual` option can be used to trigger an automated series of relaxations, during which the cell parameters change linearly from those specified by the input structure to those given in a `new cell` line (see `INPUT` below). The format of this option is `gradual NN` where `NN` is the number of steps. An output structure will be produced for each step, numbered from `00` (the relaxed input structure) to `NN`. The name used for these output structure can be chosen in the `OUTPUT` block.

Further options are described in Appendix II.

`INPUT` **block – compulsory**.

This block specifies input files. A line `structure filename` must be present indicating where to read an input structure from – without this, execution will stop.

If desired a line `bonding bondfilename` can also be given. In this case the bonding topology will be read from the specified file rather than being calculated from the bond and poly information. This is most useful if a bonding file has previously been generated from a reference structure. If bonding input is given, execution will stop if the bonding file references an atom number not present in the structure. The bonding file does contain element information to aid legibility but it is permissible for the elements in the structure not to match those in the bonding file – this allows one file to be used for a silicate and an aluminosilicate, for example. Bonding input is most useful when (i) the distance-based bonding routine might fail for the input structure, or (ii) when rigidity within an organic moiety needs to be edited. Note that a `bond` or `poly` block is still required even if a bonding input file is specified! The information in bond and poly is used in cluster formation and cannot be omitted.

The cell parameters of the input structure can be modified within GASP before a relaxation is carried out. This is specified by a line `new cell a b c alpha beta gamma`. Thus the unit cell can be sheared or compressed and the geometric relaxation will explore the structure's adaptation as a result. This allows a single reference input structure to be used to explore a range of strains. GASP will halt if a new cell is given without a relaxation being called for.

It is also possible to specify a second structure input file using the line `new structure newfilename`. In this case, the cell parameters and the fractional coordinates of all atoms are updated to the new values before geometric relaxation commences. This provides a "restart" facility for systems containing non-polyhedral bonded clusters; the geometry of the clusters will be defined by the first structure input. See Appendix VI for more details.

The "new structure" and "new cell" options are compatible; if both are given, "new structure" is applied first, then "new cell", before relaxation proceeds.

`OUTPUT` **block – optional**.

GASP can output information on structure, polyhedra and clusters, bond lengths and angles, and report the bonding network identified in the input. It can also report on the size of geometry mismatches and steric clashes. Thus a complete block might look like:

```
output
structure structureoutputfile #outputs structure
bonding bondingfile # topology
pol polfile  #polyhedral geometry
bonds bondlengthfile  # bond lengths
angles anglesfile  # all three-atom angles
mismatch mismatchfile # you can also use "clash mismatchfile"
/output
```

None of these are compulsory and indeed no output block need be given at all if only screen output is desired. The structural information in the output will be that of the relaxed structure if a relaxation has been carried out, or that of the input structure if not.

If a window search (see below) is being carried out, a "base name" for the output files should be specified using the `window` keyword, as follows:

```
window mybasename
```

Structures at the edge of the flexibility window will be exported as files named with a combination of the base name and a serial number. A logfile will also be generated with the name `mybasename.windowlog.txt`.

If the `gradual` option is selected, a "base name" for the output files can be specified using the `gradualname` keyword as follows:

```
gradualname mystructure.Nsteps
```

The output structure will be named using the basename and a numeric suffix.

Examples of input

1) An input file which identifies tetrahedra in a silicate and reports their geometry:

```
title
silicate geometry check
/title

element
si 0.26
o 1.35
/element

poly
tet si o 1.61
/poly

input
structure example-fau.cif
/input

output
pol fau.pol
bonding fau.bonding
bonds fau.bondlength
angles fau.angles
mismatch fau.mis
/output
```

2) A file to geometrically relax a zeolite with a given cell parameter and to report the relaxed structure

```
title
silicate relaxer
/title

element
si 0.26
o 1.35
/element

poly
tet si o 1.61
/poly

option
relax
smallmove 1e-6
smallmis 1e-3 # stop when polyhedra are close to ideal
/option

input
structure example-fau.cif
new cell 23.00 23.00 23.00 90 90 90 # compressing unit cell
/input

output
structure fau-23.00.xtl
/output
```

3) A file to find the bonding in a ZIF.

```
TITLE
testing GASP on ZIF4
/TITLE

ELEMENT
C 1.8
N 1.4
H 1.0
Zn 0.6
/ELEMENT

POLY #for the coordination of the metal sites
tet zn n 2.0
/POLY

BOND within 1.8 # bonding in imidazolate units
C C H N #bond carbon to any of c,h,n
N C #bond nitrogen only to c
#h is only a terminal atom and so doesn't need its own entry
/BOND

INPUT
Structure zif4.cif
/INPUT

OUTPUT
bonding zif4.bonding # report the bonding among the atoms
pol zif4.pol #cluster geometry
/OUTPUT
```

4) A file to relax the ZIF with a slightly altered cell parameter using the bonding file already generated (This allows for e.g. editing of the bonding or rigidity if needed).

```
TITLE
Testing relax on ZIF4
/TITLE

ELEMENT
C 1.8
N 1.4
H 1.0
Zn 0.6
/ELEMENT

POLY
tet zn n 2.0
/POLY

BOND within 1.8
C C H N
N C
/BOND

OPTION
relax
smallmove 1e-6
smallmis 1e-3
/OPTION

INPUT
Structure zif4.cif
bonding zif4.bonding #using the bonding topology we generated previously
new cell 15 15 18 90 90 90 #experimenting with strain
/INPUT

OUTPUT
structure rel-zif4.cif
pol rel-zif4.pol #cluster geometry to compare to original
mismatch rel-zif4.mis #mismatch and clash information
/OUTPUT
```

5) A file to carry out a “window search” on a zeolite and find the extent of its “flexibility window”; that is, the program will vary the cell parameters until the structure cannot be relaxed to within the “smallmis” mismatch criterion.

```
TITLE
Testing gasp6 on faujasite
/TITLE

ELEMENT #compulsory element block
si 0.26
o 1.35
/ELEMENT

POLY
tet si o 1.61
/POLY

OPTION
smallmove 1e-7
smallmis 1e-3 #window criterion: mismatch 0.001 Angstroms
relax #run GASP relaxation
search cub #cubic search: vary (abc) together
/OPTION

INPUT
Structure example-fau.cif
/INPUT

OUTPUT
window fau-cub-win  #basename for window-search output structures
/OUTPUT
```

GASP work case 1: analysing a structure for geometry.

The first application of GASP is typically to analyse the geometry of an input structure. We therefore discuss the output files produced by an analysis. During operation, GASP also produces screen output (using the “cerr” channel) which is generally self-explanatory. Warnings will appear on screen if an atom defined as a polyhedral centre has an inappropriate number of bonded neighbours.

Output files

“Structure” output is formatted like the input, with P1 symmetry.

"Bonding" output is a series of lines with the following format:

```
1 389   5  o si.
```

The first two columns are the IDs of two atoms to bond – in this case atoms 1 and 389 of the input structure file. The third column is a "constraint count" which will be 5 by default, but is set to 6 if the bond between the two atoms is "locked" in the sense that variation of the dihedral angle is forbidden. The final two columns are the species of the two atoms; as noted, these need not in fact match the species of the atoms in the input structure. A bonding output file can be given as bonding input. This allows the user to edit the rigidity of selected bonds if desired. Bonding rigidity and editing is discussed further in Appendix III.

"Bonds" output reports bond lengths for every interatomic bond in the structure. An output line looks like this: `1 o   389 si     1.60952`, that is, “id1 species1 id2 species2 bondlength”. All bonds around each atom are reported, so each bond will in fact appear twice in the list, once for each of the atoms involved.

"Angles" output reports all three-atom angles in the structure, in degrees. The first atom on each line is the central atom at which the angle is measured. The list thus includes both internal angles within polyhedra and external (bridging) angles between them. An output line looks like this: `1 o   389 si   537 si      158.541`. That is, three sets of “id_species”, followed by the angle in degrees.

"Pol" output: polyhedral and cluster information is intended for detailed analysis of the distortions of polyhedra from their ideal geometry. Each cluster is reported in an entry like this:

```
Poly:       1   Atoms:    385       9    105    234    293
   385  22.764105   3.059922   0.915001    22.764125   3.059928   0.915014
     9  21.640212   2.359788  -0.000000    21.639969   2.359532  -0.000310
   105  23.986603   3.544655  -0.013397    23.986787   3.544661  -0.013549
   234  22.126358   4.330903   1.669097    22.126377   4.331236   1.669438
   293  23.303346   2.004371   2.004371    23.303366   2.004282   2.004478
```

The first line reports the cluster's serial number followed by the IDs of all atoms involved. Each subsequent line reports an atom ID, the atom's Cartesian position, and the Cartesian position of the corresponding vertex in the bonding template ("ghost") for this cluster. If the cluster is a regular polyhedron then the first atom will lie at its centre. Regular polyhedra are reported before molecular clusters.

"Mismatch" and “clash” output: this will begin with a few lines of summary information on mismatch and clash sizes, followed by a list of all clashes if any are present.
The summary information looks like this:

```
Measure: |mismatch| mismatch^2 |clash| clash^2
Worst 0.000896248 8.03261e-007 0 0
Total 0.488729 0.000345185 0 0
N 962 962 0 0
Clash list:
```

The "worst" line reports the largest atom-to-vertex geometric mismatch, in Angstroms, followed by its square, then the largest steric clash, followed by its square. The "total" line reports the same measures summed over the whole structure. Generally the size of the largest mismatch, and the sum of squares of all mismatches, are the useful measures. The "N" line simply reports the number of atom-to-vertex vectors in the structure, and the number of clashes.

If any steric clashes occur, each is reported in a line as follows:

```
12 o 339 o 0.00229519 2.6977 2.7
```

which states that atom 12, an oxygen, and atom 339, an oxygen, have a steric overlap of 0.0023 Angstroms, as they lie 2.6977 Angstroms apart but the sum of their steric radii is 2.7 Angstroms.

Note that, in the `OUTPUT` block, the keywords `mismatch` and `clash` are synonyms.

GASP work case 2: relaxing polyhedral geometry

The second major application of GASP, often carried out after an initial analysis of the input structure's geometry, is a "geometric relaxation". During this process, GASP seeks to minimise the mismatch between atom positions and template vertex positions, along with steric overlap (clashes). For polyhedral frameworks, this allows us to determine how closely the atomic geometry can be brought to perfectly regular polyhedra.

Screen output

During execution GASP reports running information to the screen using the "standard error" output. This can be redirected into a file for examination if desired, thus: `gasp input.inp 2> outfile`. Most outputs are simply reports of the stage that execution has reached. If a relaxation is called for, then for every cycle of the relaxation GASP will report the largest mismatch between an atom and a cluster vertex, the worst steric overlap if any, and the largest step by which any atom is moved. For example, from a relaxation of a zeolite within its flexibility window, using the `smallmis` criterion:

```
Fitting cycle      276; Mismatch;    0.0010424 ; Clash    0.0000000 ; Move:
0.0001518056
Fitting cycle      277; Mismatch;    0.0010152 ; Clash    0.0000000 ; Move:
0.0001478864
Fitting cycle      278; Mismatch;    0.0009887 ; Clash    0.0000000 ; Move:
0.0001440643
Fitting complete:
Mismatch less than criterion 0.0010000000
GASP relaxation completed.
Relaxation IN WINDOW.
```

Note that at the end of the relaxation GASP reports the reason for stopping – either "Move less than criterion" or "Mismatch less than criterion". If the reason is that the mismatch has been made small, the line "Relaxation IN WINDOW" is also produced.

If runaway results are shown, e.g. very large moves and mismatches, it is advisable to check whether the bonding is correct. Very large clashes may indicate a need for additional damping, see the `dampclash` option in Appendix II; they may also be indicative of a bonding problem, i.e. atoms that should be bonded neighbours are being detected as a steric clash.

GASP work case 3: Polyhedral comparison and structure comparison

The polyhedral geometry information produced by GASP is convenient when comparing two analogous framework structures – structures with the same topology but whose exact polyhedral geometry can differ. GASP comes with some ancillary programs to carry out a comparison of two sets of polyhedra and decompose their differences into rotational and distortion components. The residual distortion can further by analysed into a component due to bond stretching (calculated as the sum of squares of changes in bond length for each polyhedron) and bond bending (that is, the residual distortion less the bond stretching component). This has the useful effect of placing bending and stretching distortions on a common scale for comparison.

Comparing polyhedra and the `polycomp` ***utility***.

Two `pol` files obtained from structures with the same atom numbering and polyhedral geometry – e.g. the polyhedra from a mineral structure under different degrees of compression – can be compared using the accompanying `polycomp.exe` utility. This program expects three arguments, the first two being the filenames of the two `pol` files, and the third being a filename in which to report output, thus:

```
polycomp fau0.pol fau1.pol faucompare
```

The comparison process will overlay the matching polyhedra from the two input files, quantify the total mismatch-squared between the bond vectors as given, and then identify the rotation which fits the second polyhedron to the first by minimising this mismatch. The file output has a header line:

```
ID M2-before M2-after M2-bend M2-stretch Rx Ry Rz ModR InDegrees
```

The entries are the ID of the polyhedron; the squared mismatch before fitting, and after; the components of the residual mismatch attributable to bond bending and to bond stretching; the bivector components of the rotation – these can be visualised as x,y,z components of a vector describing the rotation axis and magnitude; the rotor magnitude, and the size of the rotation in degrees. For technical reasons the magnitude of the bivector is twice the sine of half the rotation angle – this is very close to the rotation size in radians. Thus for example:

```
3 0.44806673 0.07965503 0.05398654 0.02566849  0.00954206  0.07841616 -0.21410389
0.22821179 13.10411420
```

This line indicates that polyhedron 3 from the second input file can be fitted over its match in the first input file, by a rotation whose axis lies mostly along the *z* axis, with a magnitude of just over 13 degrees. The mismatch between the polyhedra after rotational fitting is less than 20% of the mismatch before fitting (i.e. the difference is mostly rotational); and the mismatch attributable to bond bending is roughly twice that due to bond stretching.

The user should note **that the bond stretching component of the mismatch is calculated from the geometric centre of the polyhedron**. As such the measure is only meaningful for actual polyhedra; for molecular clusters the bend and stretch decomposition is not generally meaningful, though the total mismatch before and after fitting remains a valid measure.

A note on incomplete polyhedra: if GASP identifies a polyhedral-centre atom which is not bonded to enough vertex atoms to form a complete polyhedron, a warning will be written to the screen, and the incomplete polyhedron will simply be omitted from the list of polyhedra within GASP, and hence from the `pol` file output. This feature is intended to facilitate the comparison of `pol` files from defective structures generated by other modelling methods such as Reverse Monte Carlo.

Comparing the net difference of two structures.

The earliest published application of GASP concerned the comparison of two Reverse Monte Carlo models of the same structure, to decompose their differences into rigid-unit and distortion components. For this kind of application, it is convenient to have a measure of the net difference in atom positions between the two structures, before the decomposition into polyhedra. The `netdr2` program carries out this comparison. Its usage is as follows:

```
$ ./netdr2

Welcome to netdr2, computes the net difference between two structures.
Written by Stephen Wells.

Usage:
./netdr2 Name1 Name2
Reads structures from CIF files Name1 and Name2.
The user should ensure they have matching atoms (number and order),
and that the cells are at least very similar if not identical.
Will report the mean squared change in fractional coordinates,
with and without a net global motion correction,
and the equivalent mean squared motion in Angstroms-squared,
computed using the cell parameters of file Name1.
Output will go to screen through the cout channel.
```

For example we can apply this program to compare a faujasite zeolite structure before and after carrying out a geometric relaxation – that is, comparing the initial crystal structure input to the structure with idealised tetrahedra. The output of such a comparison is as follows:

```
Mean-delta-fractional-coordinates:                        0.00000000   0.00000000
0.00000000
Total-squared-delta-fractional-coordinates-uncorrected: 0.09304752
Total-squared-delta-fractional-coordinates-corrected:   0.09304752
Total-squared-delta-Cartesian-coordinates-uncorrected:  53.33218309
Total-squared-delta-Cartesian-coordinates-corrected:    53.33218309
```

The “uncorrected” and “corrected” tags refer to the differences with and without the removal of any net shift of the centroid of all atom positions between the two structures – any such shift is generally small. The calculation is carried out on the fractional atomic coordinates and takes account of cell wrapping.

Comparing rotations and the `rotordotcom` ***utility.***

The bivector rotation objects obtained by the comparison routine can be compared to each other using a scalar product. Suppose that we have polyhedra from the same system in three different states, say `fau0.pol`, `fau1.pol` and `fau2.pol` where states 1 and 2 are two different conditions of strain compared to the initial `fau0` state. Using the `polycomp` routine we can compare each of states 1 and 2 to state 0 and obtain outputs `compare0-1` and `compare0-2`. By evaluating the scalar products of the bivectors rotating each polyhedron, we can quantify the overall similarity of the two sets of rotations. This is performed by the `rotordotcom` utility, thus: `rotordotcom compare0-1 compare0-2`

The output to screen will evaluate the similarity of the rotations using a pseudo-cosine. The pseudocosine ranges from 1 when all rotations are identical to -1 if corresponding rotations are equal and opposite; a result of zero indicates that overall the rotations are orthogonal, i.e. the two sets of rotations are not related. This comparison is intended for analysis of the folding mechanisms of zeolite frameworks.

Comparing real and ideal clusters with the `polyana` *utility*

The user may also wish to directly compare the positions of atoms in the cluster to their ideal positions, that is, the "ghost" vertex positions which are also listed in the `.pol` file. This is easily carried out using a very similar logic to the `polycomp` utility; however the `polyana` utility requires only one input filename (a `.pol` file) and an output filename to store the results. The `polyana` utility reads the cluster and ghost information from the `.pol` file and reports the sum of squares of mismatches between the cluster and ghost vertices, as well as its decomposition into bond-bending and bond-stretching terms. The head of an output file looks as follows:

```
ID M2-total M2-bend M2-stretch
1 0.108827 0.0831376 0.0256894
2 0.101459 0.0757695 0.0256894
…
```

This utility may therefore be useful in assessing how close the polyhedra of a structure are to their ideal geometry, without carrying out a geometric relaxation in GASP. The user should note that the bending and stretching components are, again, only meaningful for regular polyhedra and not for molecular clusters.

GASP work case 4: window search.

The concept of the “flexibility window” arises in the study of polyhedral frameworks such as zeolites. For a given structure (topology) and a given polyhedral geometry, there is a range of unit-cell sizes and shapes within which the polyhedra can all retain their ideal geometry within a geometric simulation. This range is the flexibility window. Outside this range, distortions are inevitable, due to the stretching and bending of bonds, and clashes between nonbonded atoms.

Note that the flexibility window is a geometric property; we do not expect in practice to find perfectly regular tetrahedra in a crystal structure, due to physical factors not included in the geometric relaxation.

Automated window search: the flexibility window.

In previous versions of GASP, the search for a flexibility window in a given structure was carried out manually, by repeatedly providing an input structure and a new set of cell parameters and determining whether the structure relaxed to within a `smallmis` (mismatch) criterion of 0.001 Angstroms. This process has now been partly automated to make window searches easier to perform and understand.

The behaviour of the search routine is controlled by an option describing which cell parameters should be varied and which should remain constant. GASP generates a series of search directions based on the option selected, and then explores the effect of varying the cell parameters along each direction in turn. In general, the steps taken have a size of 0.01 Angstroms for the edge lengths *a,b,c* and of 0.01 degrees in the angles *alpha,beta,gamma*. Search directions are described by a “direction key”, a series of six symbols each of which is either `+`,`-` or `0`, representing the variation of each of the six cell parameters in order. There is a search option appropriate to each of the seven crystal systems, and additional options for isotropic scaling of the cell and for variation of each cell parameter individually and independently.

When multiple cell parameters are permitted to vary, GASP will explore them in multiple combinations. For example, if the “tetragonal” search type is selected, the *c* parameter and the *a* parameter can vary independently, with *b*=*a*. This leads to eight search directions as follows:

```
Preparing TETRAGONAL search type.
Direction key: [++0000]
Direction key: [--0000]
Direction key: [00+000]
Direction key: [00-000]
Direction key: [+++000]
Direction key: [---000]
Direction key: [++-000]
Direction key: [--+000]
```

For example, [00+000] indicates that GASP will seek to increase the *c* parameter alone will keeping all other parameters constant; [++-000] indicates that GASP will increase *a*,*b* while decreasing *c*. The lower the symmetry of the search option, the more search directions will be explored. The lowest-symmetry option, “triclinic”, generates 728 different search directions and should be invoked only with extreme caution and with a great deal of time to spare.

The “isotropic” search option has slightly different behaviour from other cases. GASP will identify the largest cell parameter out of *a*,*b*,*c*. Each search step will vary this largest parameter by the step size, 0.01 Angstroms, and will scale the other parameters in proportion, so as to explore an isotropic expansion or compression of the unit cell.

Important note: after an initial GASP run in which the automated search identifies edges to the flexibility window, it is strongly advised that the user make further investigations, either by hand or using the automated routine, starting from near the apparent edges. While in most cases the automated routine will correctly find edges, it is possible, through accumulation of errors during the search, for the automated search to fall slightly short of the true edge. Additional investigations will identify such cases and elucidate the boundaries of flexibility windows with complex shapes.

The options currently accepted by the window search routine are listed in the following table with an indication of which parameters are varied in each case.

OPTION	VARIABLES
`cub` – cubic	1 variable: *a*=*b*=*c*; 2 search directions.
`iso` – isotropic	1 variable; (*a,b,c*) scaled isotropically; 2 search directions.
`six` – six axes	6 variables; each of *a,b,c,alpha,beta,gamma* is explored independently while the others are held constant; 12 search directions.
`tet` – tetragonal	2 variables; *a*=*b*, and *c*; 8 search directions.
`hex` – hexagonal	2 variables; *a*=*b*, and *c* (identical to `tet` in function); 8 search directions.
`ort` – orthorhombic	3 variables; *a*, *b*, and *c*; 27 search directions.
`rho` – rhombohedral	2 variables; *a*=*b*=*c* and *alpha*=*beta*=*gamma*; 8 search directions.
`mon` – monoclinic	4 variables; *a*, *b*, *c* and *beta*; 80 search directions.
`tri` – triclinic	6 variables; 728 search directions.
`aaa` – a	1 variable; *a* only; 2 search directions.
`bbb` – b	As above, *b* only.
`ccc` – c	As above, *c* only.

`alp` – alpha	As above, *alpha* only
`bet` – beta	As above, *beta* only
`gam` – gamma	As above, *gamma* only.

Screen output during window search

Screen output during a window search is more verbose and passes through several phases. In an initial preparatory phase, the system prepares the vectors to explore. Each vector is reported as a series of six symbols, representing changes to the cell parameters *a b c alpha beta gamma*. In this example, for a "cubic" search, the system prepares two vectors:

```
Hi, I'm the Window Search Routine!
You have asked for window type cub
Preparing CUBIC search type:
Direction key: [+++000]
Direction key: [---000]
Prepared 2 search directions.
Relaxing:
…
```

The system begins by seeking to relax the input structure, to determine whether the starting point lies within the flexibility window. If it does not, execution will stop and no search will take place.

```
…
Fitting cycle      278; Mismatch;    0.0009887 ; Clash    0.0000000 ; Move:
0.0001440643
Fitting complete:
Mismatch less than criterion 0.0010000000
GASP relaxation of starting structure completed.
Relaxation IN WINDOW with starting cell:   23.9410  23.9410  23.9410  90.0000
90.0000  90.0000 Volume: 13722.2984266210
```

The system will then begin to explore the first search direction, altering the cell parameters and trying to relax once more. The search point will then be reported as GOOD or BAD. Once a BAD point is identified the system will backtrack until an "edge" is identified.

```
Preparing to search direction:   0.01000000   0.01000000   0.01000000   0.00000000
0.00000000   0.00000000
TRYING:  23.95100000  23.95100000  23.95100000  90.00000000  90.00000000
90.00000000
…
…
Fitting complete:
Mismatch less than criterion 0.0010000000
GOOD point:  23.95100000  23.95100000  23.95100000  90.00000000  90.00000000
90.00000000 Volume: 13739.50075435
TRYING:  23.97100000  23.97100000  23.97100000  90.00000000  90.00000000
90.00000000
…
…
Fitting complete:
Move less than criterion 0.0000001000
BAD point:  24.49100000  24.49100000  24.49100000  90.00000000  90.00000000
90.00000000
TRYING:  24.47100000  24.47100000  24.47100000  90.00000000  90.00000000
90.00000000
…
…
Fitting complete:
```

```
Mismatch less than criterion 0.0010000000
GOOD point:  24.46100000  24.46100000  24.46100000  90.00000000  90.00000000
90.00000000 Volume: 14636.00748418
FOUND WINDOW EDGE:  24.46100000  24.46100000  24.46100000  90.00000000  90.00000000
90.00000000 Volume: 14636.00748418
Finished search direction:   0.01000000   0.01000000   0.01000000   0.00000000
0.00000000   0.00000000
Finished search 1 of 2
Writing structure to fauwin.01.xtl
Done writing to file fauwin.01.xtl
```

Once the exploration of the first direction is complete, the next direction is explored, starting from the initial input position again:

```
Preparing to search direction:  -0.01000000  -0.01000000  -0.01000000   0.00000000
0.00000000   0.00000000
TRYING:  23.93100000  23.93100000  23.93100000  90.00000000  90.00000000
90.00000000
…
…
GOOD point:  22.50100000  22.50100000  22.50100000  90.00000000  90.00000000
90.00000000 Volume: 11392.14381750
FOUND WINDOW EDGE:  22.50100000  22.50100000  22.50100000  90.00000000  90.00000000
90.00000000 Volume: 11392.14381750
Finished search direction:  -0.01000000  -0.01000000  -0.01000000   0.00000000
0.00000000   0.00000000
Finished search 2 of 2
Writing structure to fauwin.02.xtl
Done writing to file fauwin.02.xtl
```

"Logfile" output during an automated window search provides a summary of the screen output information, in the following order. On the first line, a record of the search option requested. Then, a list of all search direction keys to be explored. Then, the result of the geometric relaxation of the input structure. If the input structure lies within its flexibility window, the search will proceed. For each search direction, the logfile reports each set of cell parameters tried, on a line beginning with "`TRYING:`" followed by a list of cell parameters; the next line will begin either "`GOOD point`" or "`BAD point`", followed by a list of cell parameters and a note of the cell volume. This means that lines of points lying within the flexibility window can be found by searching for lines starting with "`GOOD`", e.g. using the command "`grep GOOD logfilename`". When a point lying on the edge of the window is identified, it is reported as a line beginning "`FOUND WINDOW EDGE:`" followed by cell parameters and cell volume. There follows a note giving the serial number of the direction searched, as "`Finished direction (N) of (total)`"; and a line "`Writing to file: (basename).(N).xtl`".

Once a window edge is identified, the user may wish to generate a series of structures representing stages of the process as the cell parameters change from the input values to those of an edge. The `gradual NN` option is convenient for this process, generating a series of `NN` relaxed structures whose cell parameters vary linearly from the input to those given in a `new cell` line. Thus the user can use the `relax` and `gradual` options and provide a cell edge using `new cell`. The gradualname option in the OUTPUT block provides a base name for this series. Remember that you should NOT use the `search` option and `gradual` simultaneously!

Appendix I : geometric relaxation method.

The heart of the geometric simulation approach is the use of a rigid-body template or "ghost" to represent the bonding geometry in a group of atoms. A harmonic penalty is placed on mismatches between atoms and template vertices. This penalty represents all distance and angle constraints within the cluster. If the template is defined as a regular polyhedron, GASP will generate a geometrically regular template of the appropriate shape; for non-polyhedral clusters, the template geometry is based on the atomic geometry of the input structure.

A single cycle of geometric relaxation involves the following steps. Firstly, the template is placed with its centre coincident with the centre of the corresponding group of atoms. Secondly, the template is rotated to minimise the total mismatch between template vertices and atomic positions using a least-squares fit (Mathematical note: the rotation engine is based on Clifford algebra rotors, which are quite similar to quaternions). Thirdly, the position of each atom is updated to minimise its mismatches to template vertices and its steric overlap with any other nonbonded atom. In this step the penalty function is the sum of squares of vertex mismatches, plus a damping factor times the sum of squared steric overlap distances. This damping factor is set to 0.5 by default but can be altered using the `dampclash` option, see Appendix II. Effectively the damping allows atoms in contact to move out of contact without "springing" too far, which causes numerical instability; smaller values can be useful in cases of large steric overlap.

GASP will continue to carry out cycles until a stop criterion is satisfied, i.e. the atom-position update moves become small or the mismatches become small. This generally takes only a few seconds for systems of hundreds of atoms within the flexibility window.

In more complex structures it is possible for GASP’s minimisation routine to encounter difficulties and “jam” for many cycles rather than resolving. This is inconvenient when using the `search` option as it prevents efficient progress. GASP therefore now includes two features to avoid jamming during geometric relaxation. The first is a trend-following scheme which effectively converts from steepest-descent to a line search process after the first 1000 iterations of relaxation, providing faster convergence in difficult systems. The second is a down-scaling system which multiplies the calculated moves by a linear factor ranging from 1 at the outset to 0 after 100,000 iterations. This has no effect on rapid relaxations but guarantees that in jammed situations the process will halt, reporting that mismatches could not be resolved, after no more than 100,000 cycles. This allows a window search to identify an edge and proceed rather than being jammed indefinitely.

Appendix II : additional options

Several settings can be altered in the option block if desired. The `relax` option is discussed in the main text.

`grid` option. GASP places atoms into a coarse grid of cells so as to make neighbour-finding order(N), as only adjacent cells need to be searched. By default a grid cell size of 3.0 Angstroms is used. The cell size needs to be at last twice the largest steric radius in the structure, and so a larger spacing can be defined thus: `grid 3.5` .

`polypad` option. By default GASP seeks polyhedral vertices within 1.3 times the given bond length from the central atom. This factor can be increased or decreased: e.g. `polypad 1.5`, if polyhedra need to be found in a more distorted input structure.

`smallmove` option. By default GASP will stop refinement when the size of the largest move generated for any atom is less than 1e-6. This value can be tuned: e.g. `smallmove 1e-5` if you think higher accuracy is not required, or `smallmove 1e-7` for a more demanding simulation.

`smallmis` option. By default GASP only monitors the move size as a stop criterion. If however this option is given, it will also monitor the size of the worst atom-to-vertex mismatch and stop if it becomes less than the given value. For example, we might set `smallmis 1e-3` so that relaxation of a polyhedral framework halts when all polyhedral geometry is good to within 0.001 Angstroms; this is the usual criterion for a flexibility window investigation.

`dampclash` option. By default the penalty for a steric clash is half that for a vertex mismatch. The fitting process can become numerically unstable if this multiplier increases towards 1, as atoms tend to jump in and out of contact instead of resolving clashes smoothly. If instability still occurs you can try setting a lower multiplier, e.g. `dampclash 0.1`. This is recommended if large clashes are present in the input structure.

`nodeloc` option. This deactivates the special rules for rigidity in sp2 systems; see Appendix III.

`label` option. This causes xtl output to include numeric serial numbers for all atoms.

`r14` option. This takes a numerical argument of 1.0 or less (e.g. `r14 0.8`) and scales the contact radii for atoms that are in a 1-4 bonded relationship, that is, the A-D contact interaction in a chain of bonded atoms A-B-C-D. This can be important in structures containing non-rigid rings, for example if a MOF has flexible ligands containing rings such as cyclohexane. It can also be significant if the `nodeloc` option is used so that sp2 rings are not unified into a single rigid cluster. Without this option, the use of a realistic steric radius for carbon (1.5-2.0 Angstroms) would be impossible, as carbons opposite each other in the ring would register as a steric clash.

`imperfect` option. If provided, this option disables the setup step in which GASP builds geometrically ideal templates for the polyhedral bonded groups. As a result, the polyhedral groups will simply retain the geometry they have in the input structure, just like a non-polyhedral bonded group. For most polyhedral types this is equivalent to replacing the POLY block with a BOND block. The main utility of this option is that it can be used with the BAR polyhedral type so as to represent the bonding as a set of bars with fixed length but without angle constraints. Thus if the input structure contains, for example, an MO_6 octahedron with non-ideal geometry, as is common for example in MOFs, the bonding could be represented using a `bar m o typical-bondlength` definition and the `imperfect` option. This will set each M-O bond to the length it has in the input structure, but will not place any constraint on the O-M-O angles, allowing some flexibility in the unit. The use of a BOND block specifying M-O bonding, by constraint, would unite the MO_6 unit into one rigid unit, fixing all O-M-O angles.

`xtl` option. If passed, GASP will expect structure input and output in `.xtl` format (see Appendix V) rather than `.cif`.

Appendix III : rigidity in sp2 systems

The bonding routine will initially build a cluster for each "central" atom and its bonded neighbours. Vertex atoms – those that do not appear first in any poly or bond specification – do not get their own clusters. In molecular systems with multiple C-C, C-N etc. bonds, clusters will overlap with each other along the shared bond. Ordinarily this overlap constrains bond angles but allows free variation of the bond dihedral angle. If, however, delocalised bonding extends over the system, it may be more appropriate to lock certain dihedrals. This is achieved by uniting adjacent clusters into one. GASP runs an iterative cluster-unification procedure until all atoms connected by rigid bonds belong to unified clusters.

If a bonding input file is given, a rigid bond is labelled by giving it 6 constraints instead of 5. This will prompt GASP to unite the clusters.

In the internal bonding routine, GASP uses a very simple system to spot potentially rigid bonds. Any atom whose species is `C` or `N` and which has three or fewer neighbours will be labelled as a potential sp2 site. When two such sites are adjacent in the bonding, the bond between them is labelled as rigid. This will trigger cluster unification, and the bond will be reported as having 6 bars if a bonding output file is written.

This delocalisation-induced rigidity can be removed by the user if desired, e.g. to compare and contrast the behaviour of the system with and without rigidity. There are two means to edit the rigidity of bonds. If only certain individual bonds are to be edited, it is advisable to (i) run GASP on the system and request bonding output; (ii) identify the relevant bonds in the bonding output file, and reset the constraint count from 6 to 5; (iii) provide the edited file as bonding input for subsequent GASP runs. Thus for example a line:

```
1 30   6  c n  #6 bars
```

might be edited to read:

```
1 30   5  c n   #5 bars
```

making the bond free to rotate.

If on the other hand it is desired to remove all delocalisation effects from the system, the `nodeloc` option can be given in the `OPTIONS` block. This will prevent any application of the special rules for sp2 systems. The `r14` option discussed in Appendix II may be appropriate in this case to avoid spurious across-ring steric interactions.

Appendix IV: supercell formation, duplicate atom removal.

As well as the `gasp.exe` and `polycomp.exe` programs, GASP comes with two more small utilities, `super.exe` and `duper.exe`.

`super.exe` can be used to create a supercell. Usage of the code is as follows:

```
super NameIn NX NY NZ NameOut
```

`NameIn` is the name of a P1 structure input file in `cif` format; `Nx`, `Ny`, `Nz` are integers. The code will generate an `Nx*Ny*Nz` supercell of the input and write it to a file called `NameOut`. This can be useful in generating a large enough supercell for flexibility effects to become apparent. As a rough guide, if any cell parameter in the input is less than about 10 Angstroms, it is advisable to form a supercell. Use of the `super` utility is particularly recommended if a single rigid unit would otherwise extend across the entire system – this would cause instability in the rotor fitting routine within GASP.

`duper.exe` detects and removes duplicate atoms from an xtl file. Preparation of a P1 xtl file from a high-symmetry input can occasionally lead to atom duplication, for example the same atom existing with fractional coordinates of 0.0 and 1.0; or symmetry copies lying at effectively the same position but being reported as multiple atoms instead of one. Usage of the code is as follows:

```
duper NameIn NameOut
```

It will parse the input `cif` file, detect any duplications, and report only non-duplicated atoms to the output file.

Appendix V: xtl file format.

Previous versions of GASP have made use exclusively of the `.xtl` file format for structure input and output. This format is briefly described here for legacy purposes.

XTL format for input

This format simply provides cell parameter information followed by a list of elements and fractional coordinates for all atoms. The header and footer of an example file are as follows:

```
TITLE Faujasite
CELL
23.941 23.941 23.941 90 90 90
ATOMS
NAME X Y Z
O 0 0.8932 0.1068
O 0 0.3932 0.6068
…lines omitted…
Si 0.78515 0.3732 0.5525
Si 0.78515 0.8732 0.0525
EOF
```

The last line (`EOF` = end of file) is required for a clean termination of file read. What follows `TITLE` is largely irrelevant. Space delimiters are safer than tabs. The format can easily be created in a spreadsheet from a list of coordinates exported from popular viewers such as Crystalmaker.

Optional labelled version of `xtl` file: if the `label` option is provided in the `options` block of the GASP input file, then any output `xtl` file will also include a serial number for the atom, thus:

```
…
ATOMS
NAME X Y Z
o   1.00000000   0.90159218   0.09840782 1
o   1.00000000   0.40159218   0.59840782 2
…
si   0.78810190   0.37748623   0.55152986 575
si   0.78810190   0.87748623   0.05152986 576
EOF
```

These labels may be helpful in correlating the information in the bonds and angles output files to the structure input. The labels are ignored if the `xtl` file is read by GASP or `xtl2cif`.

A standalone routine, `xtl2cif.exe`, is provided with GASP. When executed as `xtl2cif.exe xtlfile ciffile` it will read `xtlfile` and generate `ciffile` in CIF format. This is helpful in transforming GASP `xtl` output into an easily viewable form. The reverse operation is carried out by the `cif2xtl.exe` routine. Routines `xtlsuper.exe` and `xtlduper.exe` duplicate `super` and `duper` but expect `xtl` format for structures.

Appendix VI: "milestoning" using the `new structure` *command*.

The INPUT block allows the user to provide several forms of structural information using the `structure`, `new structure`, `new cell` and `bonding` commands. In this appendix we briefly discuss the use of these commands to carry out a "milestoning" investigation of structural compression. Suppose we have an initial input structure, `ambient.xtl`, of a cubic MOF structure with *a*=25.0 Angstroms. The molecular geometry of this input structure needs to be retained during the investigation, so in all cases we will use the line "`structure ambient.xtl`" in the INPUT block. If the *a* parameter is changed significantly, we may find that GASP starts to take longer and longer to converge, as it must find larger and larger rotations of the molecular clusters in the linker sections of the MOF structure. It is then advantageous to record a compressed structure as a milestone, and to provide information on this structure as a starting point for further compressions. We will consider, firstly, the creation of a milestone, and secondly, the use of one.

Let us suppose we have already found the structure can be compressed from *a*=25.0 to *a*=23.0 Angstroms. Further compression appears possible but convergence has become slow. We will establish a milestone by exporting the structure and bonding topology. The INPUT and OUTPUT blocks will appear as follows:

```
INPUT
structure ambient.xtl
new cell 23.0 23.0 23.0 90 90 90
/INPUT
OUTPUT
structure at-a23.0.xtl
bonding mybonding.txt
/OUTPUT
```

We now investigate further compression to a=22.0 Angstroms, using this milestone. The INPUT and OUTPUT blocks for the next run will be as follows:

```
INPUT
structure ambient.xtl
bonding mybonding.txt
new structure at-a23.0.xtl
new cell 22.0 22.0 22.0 90 90 90
/INPUT
OUTPUT
structure at-a22.0.xtl
/OUTPUT
```

The effect of these commands is as follows. GASP reads the initial input geometry from `ambient.xtl` and the bonding topology from `mybonding.txt`, and uses them to build the template geometry of the molecular clusters. It then reads the cell parameters and fractional coordinates from the milestone structure `at-a23.0.xtl` and updates the cell and atom positions, but the template cluster geometries do not change. Next the simulation cell is altered again according to the `new cell` command. The fractional coordinates of the atom remain as given in the milestone structure and the template cluster geometries do not change. The relaxation of the structure will then proceed from this state.