Journal of
Applied Crystallography
ISSN 0021-8898

Received 20 October 2010
Accepted 25 November 2010

J-ICE: a new *Jmol* interface for handling and visualizing crystallographic and electronic properties

Pieremanuele Canepa,[a]* Robert M. Hanson,[b] Piero Ugliengo[c] and Maria Alfredsson[a]

[a]School of Physical Sciences, University of Kent, Ingram Building, Canterbury, Kent CT2 7NH, England, [b]Department of Chemistry, St Olaf College, 1520 St Olaf Avenue, Northfield, MN 55057, USA, and [c]Dipartimento di Chimica IFM, Università degli Studi di Torino, and NIS (Nanostructured Interfaces and Surfaces) Centre of Excellence, Via P. Giuria 7, Torino 10125, Italy. Correspondence e-mail: pc229@kent.ac.uk

The growth in complexity of quantum mechanical software packages for modelling the physicochemical properties of crystalline materials may hinder their usability by the vast majority of non-specialized users. Consequently, a free operating-system-independent graphical user interface (GUI) has been developed to drive the most common simulation packages for treating both molecules and solids. In order to maintain maximum portability and graphical efficiency, the popular molecular graphics engine *Jmol*, written in the portable Java language, has been combined with a specialized GUI encoded in HTML and JavaScript. This framework, called *J-ICE*, allows users to visualize, build and manipulate complex input or output results (derived from modelling) entirely *via* a web server, *i.e.* without the burden of installing complex packages. This solution also dramatically speeds up both the development procedure and bug fixing. Among the range of software appropriate for modelling condensed matter, the focus of *J-ICE* is currently only on *CRYSTAL09* and *VASP*.

1. Introduction

The advent of many molecular graphics packages, such as *DLV* (Searle, 2001), *GaussView* (Gaussian Inc., Connecticut, USA; http://www.gaussian.com/gv_plat.htm), *Jmol* (Hanson, 2008), *MOLDEN* (Schaftenaar & Noordik, 2000), *MOLDRAW* (Ugliengo *et al.*, 1993, 1988; Ugliengo, 2006), *Materials Studio* (Accelrys Inc., California, USA; http://accelrys.com/products/materials-studio/), *PyMOL* (Schrödinger LLC, New York, USA; http://www.pymol.org/), *RASMOL* (Sayle & Milner-White, 1995) and *XCrysDen* (Kokalj, 2003, 1999), has transformed the way scientists deal with molecular and crystallographic structures. The capability of these packages to manipulate the outcome of either experiment or simulation goes beyond the simple representation of 'balls connected through sticks'. For instance, they can effectively superimpose electron charge-density maps on preloaded structures, visualize vibrational normal modes (IR or Raman), show symmetry operators, expand crystallographic cells to supercells, export and import files in several formats, prepare high-resolution printer-ready images, and many other functions.

In most cases, however, the lack of portability of molecular viewers across different operating systems becomes critical or, when such a transfer is feasible, it forces the user into long-winded and complex installation processes. This becomes dramatically true when, for instance, highly efficient graphical libraries are involved in rendering tasks. One exception is *Jmol*, which, being written in the portable Java language, does indeed run on any commodity computer, independent of the operating system adopted (Microsoft Windows, Apple Mac OSX or Linux). The great advantage of *Jmol* is that it can be used from the web-server side, saving the final user from any long-winded installation procedure on the client machine. Because of this, web sites exploiting the power of *Jmol* have flourished rapidly (see, for example, Gutow, 2006, 2010; Noel, 2008, 2009; Noel & Demichelis, 2009; Martz, 2009; McMahon & Hanson, 2008), especially where crystallographic information is a requirement. Indeed, *Jmol* is capable of reading structures from the most important crystallographic databases, such as the American Mineralogist Crystal Structure Database (http://rruff.geo.arizona.edu/AMS/amcsd.php; Downs & Hall-Wallace, 2003), the Cambridge Structural Database (http://www.ccdc.cam.ac.uk/products/csd/; Allen, 2002), the Inorganic Crystal Structure Database (ICSD, 2004) and the Protein Data Bank (PDB; http://www.pdb.org/pdb/home/home.do; Berman *et al.*, 2000).

However, *Jmol* has one drawback that slows down its adoption by casual users: the main subtleties are only accessible *via* a command-line interface (the 'console'). Although this is a very powerful approach, as it allows complex scripts to be prepared, it also frightens many users as the learning curve can be rather steep. The aim of this work is to show a new framework for *Jmol*, called *J-ICE*, which literally 'dresses' *Jmol* and screens the user from the complexity of the *Jmol* engine. The framework is fully written in standard HTML and JavaScript, preserving the portability of *J-ICE*.

At present *J-ICE* focuses only on rendering and handling crystal structures and their properties, as resulting from the application of programs like *CRYSTAL09* (Dovesi *et al.*, 2005, 2009), *CASTEP* (Segall *et al.*, 2002), *FHI-AIMS* (Blum *et al.*, 2009), *QUANTUM ESPRESSO* (Giannozzi *et al.*, 2009), *VASP* (Kresse & Hafner, 1993; Kresse & Furthmuumller, 1996) and *WIEN2K* (Blaha *et al.*, 1990; Schwarz & Blaha, 2003).

2. Architectural considerations

This section covers briefly the technique used to assemble *J-ICE* (§2.1) and how the information is organized within the interface (§2.2).

2.1. What is under the bonnet?

J-ICE maintains, as does *Jmol*, the twofold nature of a standalone computer and a web interface. In fact, *J-ICE* can readily be browsed on the remote server provided at http://j-ice.sourceforge.net/ondemand/index.html. Alternatively, *J-ICE* can be downloaded from http://sourceforge.net/projects/j-ice/ and run on the user's own PC (or workstation). The major inconvenience of the latter is that the installation of *J-ICE* remains at the expense of the user. The two versions (stand-alone and web) do not differ at all from one another, which guarantees consistency within the *J-ICE* project.

As mentioned earlier, *J-ICE* relies fully on *Jmol*. Therefore, in *J-ICE*, all the graphical tasks and import and export functions are wholly managed by the *Jmol* applet (see Appendix *A*1 for a definition of a Java applet), downloadable from the public mirror (see above). This is of crucial importance, since *Jmol* is very powerful and fast in dealing with complex graphical processes. Secondly, many input and output formats can be imported and exported by *Jmol* itself (a complete list of the formats currently available in *Jmol* is available at http://jmol.sourceforge.net/#What%20Jmol%20can%20do).

In order to fulfil the idea of cross-compatibility between operating systems, together with the majority of current browsers (Microsoft Internet Explorer, Mozilla Firefox, Google Chrome and Apple Safari), the *J-ICE* interface is entirely written in standard HTML (Version 4.1). The widget tab structure adopted is based on previous projects developed by Noel (2008, 2009).

Communication between *J-ICE* and *Jmol* was made possible *via* two levels of interaction: (i) JavaScript and (ii) the *Jmol* scripting language. In the former, the existing JavaScript library, distributed along with *Jmol*, was expanded by several new ones to ensure maximum manipulation of all incoming and outgoing information accessible from *Jmol*. The latter makes possible the execution of commands within *Jmol*. However, such subdivision is not always maintained, as mixing between these two languages ensures maximum flexibility of the interface. Fig. 1 summarizes how the HTML interface interacts with the *Jmol* applet.

Third-party JavaScript libraries such as *Flot* (Laursen, 2010) and *Jmol Color Picker Box* (Gutow, 2010) are employed for plotting numerical data and visualizing colour boxes, respectively.

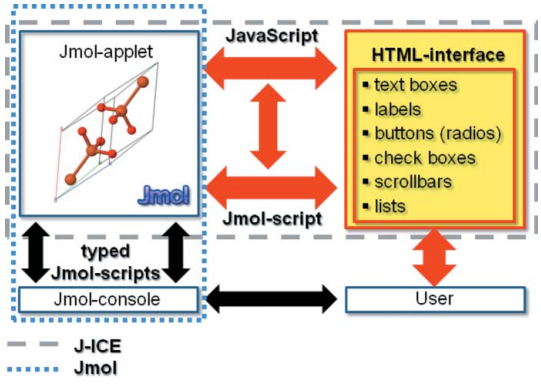

Figure 1
The communication scheme adopted by the *J-ICE* interface. The dotted box highlights the interface employed in *Jmol*, while the dashed one represents that employed in *J-ICE*.

Standard Cascade Style Sheet (CSS) language is used for content organization, as well as to render the *J-ICE* graphical interface more appealing to the user.

J-ICE is distributed following the philosophy of the General Public License (GNU).

Documentation is provided through videos (available at http://j-ice.sourceforge.net/) which cover 80% of the options presently available in *J-ICE*.

2.2. Interface and data organization

J-ICE looks like a standard web page, containing classic HTML objects such as text boxes, scrollbars, lists, buttons, radio buttons and check boxes. Fig. 2 shows a snapshot of the current *J-ICE* interface.

The interface is composed of four parts (Fig. 2): (1) the widget menu (see Appendix *A*2 for a definition of a widget), at the top, which contains 12 items (widgets); (2) the *Jmol* applet, located on the left-hand side; (3) the interactive palette, on the right-hand side, which shows the contents of each of the 12 widgets; (4) the control bar, situated at the bottom of the interface and containing a series of general-purpose buttons.

The information accessible from *Jmol* is outlined by a widget menu containing 12 different classes of precise physical meaning [see (1) in Fig. 2]: `File`, `App.` (appearance), `Edit`, `Meas.` (measure), `Orient.` (orientate), `Cell`, `Poly.` (polyhedron), `IsoSur.` (isosurface), `Geom.` (geometry), `Freq.` (frequency), `E&M` (electronic and magnetic properties) and `Main`. The relevant ones will be covered in detail in §3.

Particular care was taken in planning such subdivision, in order to make *J-ICE* as practical as possible. From this perspective the user should gain an understanding of most of the *J-ICE* functions (and features) in a relatively short time.

The interactive palette area contains all the functions belonging to each of the 12 widgets. It is here where the user, acting through classic and radio buttons, checkboxes, and lists, manipulates (on the fly) what appears in the *Jmol* applet [on the left-hand side, see (3) in Fig. 2]. By clicking on the active widget menu items [see (1) in Fig. 2], the user can easily swap from one palette to another. The widget design adopted is of major importance, as all the information can be grouped into a limited area, despite keeping a rational organization of the data.

In the fourth area, *i.e.* the control bar [see (4) in Fig. 2], a variety of general-purpose buttons are clustered. The bar gives a third degree of

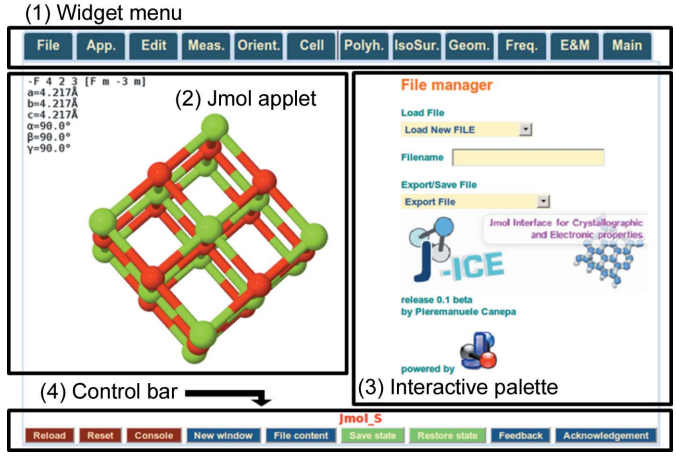

Figure 2
J-ICE interface scenario. Four areas of interest are discussed: (1) the widget menu; (2) the *Jmol* applet; (3) the interactive palette; and (4) the control bar.

freedom to *J-ICE*, since its buttons are always visible, whichever palette is active. These buttons are `Reload`, `Reset`, `Console`, `New window`, `File content`, `Save state`, `Reload state`, `Feedback` and `Acknowledgment`. The first three allow the user (in sequence) to reload the last input or output, to reset *J-ICE* to its initial state and to open the *Jmol* console. This last is useful to enter extra commands not covered by *J-ICE*. As *Jmol* implements a large number of options spanning several scientific fields, it was decided to restrict *J-ICE* to only those options related to the fields of solid-state chemistry and physics. The `New window` button clones the current *Jmol* view in a new larger and expandable *Jmol* window pane (not shown here). The new window contains buttons to refresh its status, according to changes entered in the principal *Jmol* applet, and to export images and *POV-Ray* files. The `File content` button shows the content of the current loaded file. The two buttons `Save state` and `Reload state` enable the user to save and restore work before proceeding with new changes. This procedure is temporary and all data are lost when the user terminates or restarts the interface. The `Feedback` button is a convenient way of contacting the developer if bugs or problems are encountered when using *J-ICE*. The last button, `Acknowledgment`, is self-explanatory.

3. How does *J-ICE* manipulate and deal with data?

Here, more details are given about the relevant sections (palettes) of the *J-ICE* interface.

The first widget, `File`, groups two separate lists, `Load file` and `Export file` (Fig. 2). In the former, the user can open numerous input or output formats available in *Jmol*. Particular attention has been paid to the manipulation of *CRYSTAL09* and *VASP* outputs, as will be shown later. A great deal of effort has also been put into the capability to load and then overlay electron-density maps (electrostatic potential maps) onto periodic structures [*Jmol* reads many map formats, such as CUBE, JVXL (Hanson, 2009) and others]. The export list also includes current formats that are not available in *Jmol*. At present, *J-ICE* can generate input files for *CASTEP*, *CRYSTAL09*, *VASP* and *QUANTUM ESPRESSO*. This is an example where JavaScript and *Jmol* script languages are well mixed. In the case of *CRYSTAL09*, *J-ICE* prepares the input by exploiting the symmetry features built in. For *QUANTUM ESPRESSO* and *VASP*, the user can choose to export a structure in either fractional or Cartesian coordinates. In principle, the user can employ *J-ICE* as an import/export tool, transforming universal CIF or PDB structure files into an input format compatible with the simulation packages listed above.

The widgets `App.`, `Edit`, `Meas.` and `Orient.` are simply graphical representations of classical options available through the *Jmol* shell. In `App.`, the user can alter the look of atoms and bonds and atomic radii, and calculate hydrogen bonds. `Edit` enables the user to remove atoms, partially or totally, and to fix connectivity properties. Other functions related to the manipulation of crystal structures, such as atom replacement with new atomic species or the building of a supercell starting from the unit cell (available in the `Cell` widget to preserve the data organization), allow the user to modify a preloaded structure and therefore to prepare new input files for the simulator packages mentioned above. For the time being, functions capable of modifying cell parameters (apart from the supercell) and introducing new atoms are not present. The `Meas.` palette is for measuring bond lengths (in several units), bond angles and torsion angles. The orientation properties of periodic systems, such as orientation along cell parameters, total or partial translation and rotation of the entire periodic system, are carried out in the `Orient.` palette.

To analyse more advanced properties in depth, widgets such as `Cell`, `Poly.`, `Geom.`, `Freq.` and `E&M` are available. `Cell` encompasses most of the options necessary when creating a visual representation of the unit cell, which can then be forced to form a new supercell. Some of the functions available in `Cell` are mainly dedicated to *CRYSTAL09*, as it works on the primitive cell rather than on the conventional one, although the user can swap from the primitive to the conventional cell and *vice versa* with a simple click.

Fig. 2 highlights the symmetry part available in the `Cell` palette. This is a significant option for teaching crystallography to beginners, as well as for facilitating the interpretation of crystallographic properties. Of great importance are the space-group properties accessible within *J-ICE* (Hanson, 2010). These are the space-group name in the notations of Hermann–Mauguin and the crystallographic tables, the lattice type (cubic, orthorhombic, triclinic *etc.*), and the number of symmetry operators associated with the space group (see Fig. 3). Each of these can be overlaid on the structure, facilitating the interpretation of solid-state properties, which are very often linked to symmetry.

The `Poly.` widget deals with the representation of polyhedra available in *Jmol*, which, apart from enhancing the image quality, certainly gives more insight into the structure itself. The current release of *J-ICE* allows the user to create polyhedra, either following the standard method within *Jmol* or by tweaking some parameters involved in their construction.

Isosurfaces (see above) are manipulated through the widget `IsoSurf.` (not shown here). The user can create (on the fly) isosurfaces already preset in *Jmol* (such as classical van der Waals, solvent-accessible and molecular electrostatic potential isosurfaces) or alternatively manipulate preloaded ones. Currently, representation of isosurfaces remains a difficult task for most available viewers, so *J-ICE* implements some new features, recently introduced in *Jmol*, which allow the user to replicate the isosurface across space and to read off values by surveying the surface. *J-ICE* includes a series of lists and buttons for changing the colour-map and scale settings.

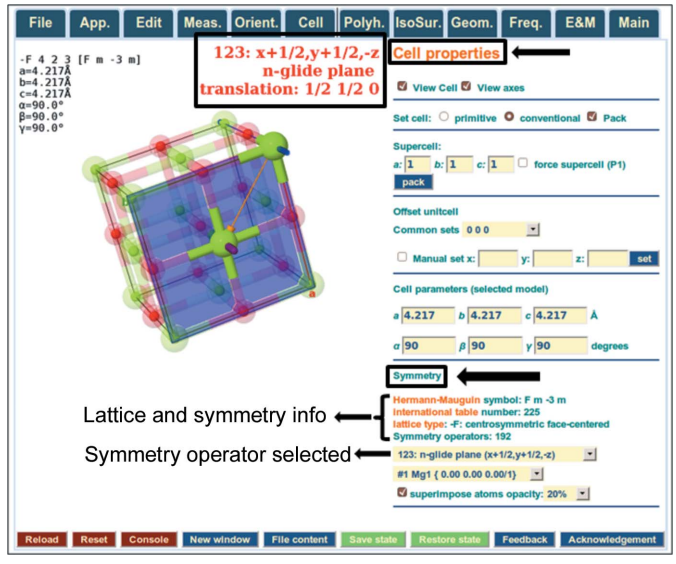

Figure 3
A snapshot of the `Cell` widget palette of *J-ICE*. On the MgO structure visible in the *Jmol* applet is drawn a glide plane chosen previously through the highlighted list. Other information about the cell and the space group is also visible.

3.1. Geometry optimization, frequencies and other properties

Special attention was paid to allow the user to follow the geometry optimization of periodic systems. At present *J-ICE* makes this option available only for outputs generated with *CRYSTAL09* and *VASP*. Such a limitation is imposed by the development of *Jmol*. All models belonging to the geometry optimization task are selected by means of a clickable active list within the `Geom.` widget. An idea of how the energy and force are dumped during the geometry optimization is given in two active graphs located beside the model list. These plot the variation in energy and force as the geometry optimization proceeds. The `Geom.` palette also provides a 'video tape recorder'-like control, which guarantees the user direct control of the geometry optimization. The possibility of exporting optimization frames to create movies is also present.

Next to the `Geom.` widget, the user can access the frequency palette (`Freq.`), as depicted in Fig. 4. This option is currently available for *CRYSTAL09* outputs and for *VASP*. Frequency eigenvalues are listed in an active list (when an element in the list is clicked, the animation of the matching vibration becomes active). Such a list can be altered by a variety of radio buttons, presented in the frequency-control area (see Fig. 4). There the user can list, for instance, just infrared (IR) or Raman frequencies. Similarly, frequencies can be separated on the basis of symmetry, and the user can still adjust the vibration amplitude and the length and size of the eigenvectors associated with the vibration (frequency controls, see Fig. 4). Noteworthy are the two zones next to the frequency-control area, namely the frequency diagram (top of Fig. 4) and the plot controls (bottom of Fig. 4). The former plots frequency values *versus* their simulated intensities, which gives a preview of what a possible spectrum might look like. In the panel below (plot controls, see Fig. 4), several options such as type of convolution, vibration mode, peak bandwidth and frequency range allow the user to simulate an accurate spectrum (see Fig. 5) starting from bare intensity data.

Three sorts of spectroscopic convolution are included: stick, Gaussian and Lorentzian. Pressing the button marked `Simulate spectrum` (plot control area, see Fig. 4) causes a new window to pop up containing the convoluted spectrum (Fig. 5). Buttons contained in the spectrum windows allow the user to rescale on the fly and to export the spectrum as a classic ASCII file.

The `E&M` (electronic and magnetic properties) widget has only a few options. Here the user can superimpose, when available, Mulliken population analyses, spin arrangements (see Fig. 6) and magnetic moment values onto the periodic structure. The novel idea of superimposing spin arrangements, magnetic moments or Mulliken charges onto the structure will possibly help the user to understand the role played by atoms and spins in complex structures.

The last widget, `Main`, gathers functions linked directly to the interface itself, such as background colour, perspective options, anti-alias, light-setting and text size of messages.

4. Conclusion

We have developed a new web-server framework, *J-ICE*, which allows users to visualize, build and manipulate complex input/output results (derived from modelling) entirely *via* a web server, *i.e.* without the burden of installing complex packages. *J-ICE* is available at http://j-ice.sourceforge.net/ondemand/index.html. *J-ICE* can also be downloaded (at the address http://sourceforge.net/projects/j-ice/) and run on the user's own local machine, at the cost of installation.

Video documentation for *J-ICE* is available at http://j-ice.sourceforge.net/. In the future, we plan to expand *J-ICE* to accommodate other simulation packages available for modelling condensed

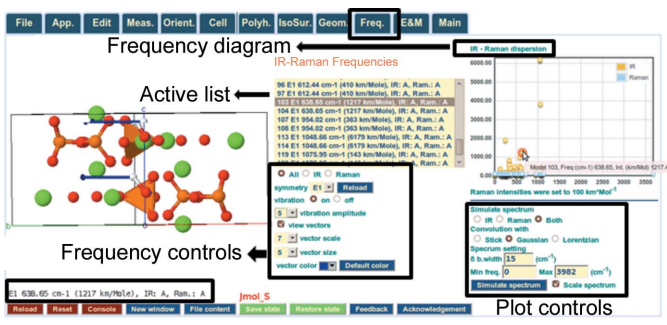

Figure 4
The `Freq.` widget palette of *J-ICE*. The information is partitioned into four separated areas: active list, frequency controls, frequency diagram and plot controls. The vibration of the hydroxyapatite bulk, depicted in the figure, was activated by hovering on the frequency diagram.

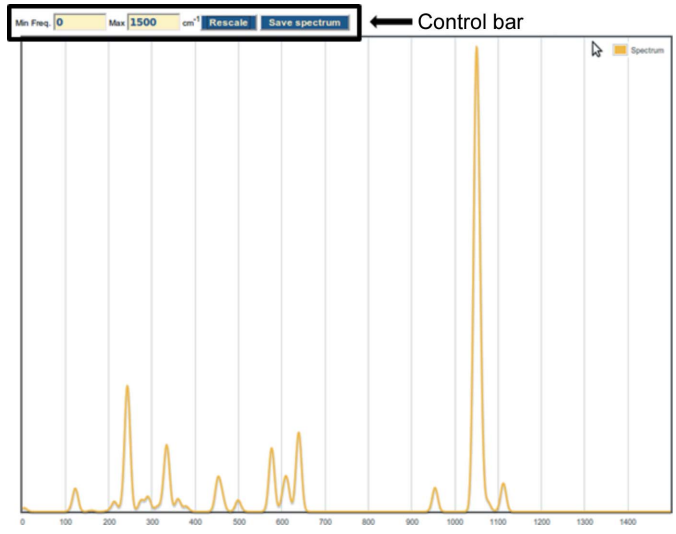

Figure 5
The spectrum of the hydroxyapatite bulk obtained by convoluting Gaussians with the single-frequency data simulated with *CRYSTAL09*. Noteworthy is the control bar, which allows users to rescale the spectrum and later export it.

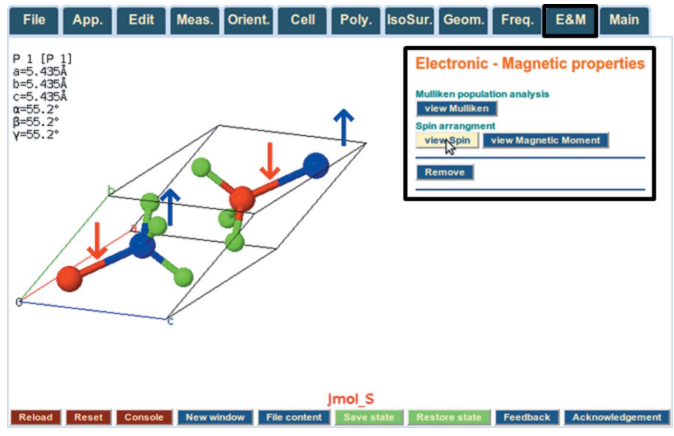

Figure 6
A view of the `E&M` widget palette of *J-ICE*. The black rectangular box shows the options available in this palette, such as Mulliken population analyses, spin arrangements and magnetic moments. The figure also depicts an antiferromagnetic spin arrangement of α-Fe_2O_3.

matter, as well as to introduce more features devoted to the manipulation of crystal structures.

APPENDIX *A*

A1. The Java applet

'An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. When you use a Java technology-enabled browser to view a page that contains an applet, the applet's code is transferred to your system and executed by the browser's Java Virtual Machine.' (Oracle, 2010.) Thus, Java applets provide extra features not available in pure HTML. Depending on the task the Java applet is designed for, it can capture mouse inputs, including buttons, check boxes and images, and many other options. Java applets can also render sophisticated three-dimensional images (and models) relying on the powerful Java3D libraries. All these qualities make Java applets useful tools suitable for demonstration, visualization and teaching.

A2. Widgets

A widget is a small independent application which is embedded in or separate from the main program window or web page (Lal & Chava, 2010). This can be seen either as a box (or several boxes) embedded in the main program area, or as a separate window (in the detached case). Widgets contain input/output elements, such as text boxes, buttons, scroll bars *etc.*, which either operate directly on their contents or commit changes to the main program area. In the specific case of *J-ICE*, widgets are incorporated into the main program window and used to handle all the information available for crystal structures in rational chunks. Widgets are generally written in highly portable programming languages compatible with the main browsers available on the network, such as DHTML, JavaScript (in its multiple facets, *e.g.* Ajax, as well as XML, jQuery and others) and Adobe-Flash.

The authors thank several people who have contributed to this idea. Firstly, Y. Noel at the Université Pierre et Marie Curie (Paris, France) for inspiring us to develop such an interface. Also Professor R. Dovesi and the theoretical chemistry group at the University of Turin (Italy) for useful suggestions during the development of the interface, and Fabio Chiatti, Marta Corno and Massimo Dellepiane (at the University of Turin, Italy) for having thoroughly tested *J-ICE*. We are also grateful to the University of Kent for funding part of this project, and to all the PhD students of Room 104 of the Physical Sciences Department for useful discussions. We also thank the *Jmol* team for their endless patience and helpful suggestions.

References

Allen, F. H. (2002). *Acta Cryst.* B**58**, 380–388.
Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalow, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res.* **28**, 235–242.
Blaha, P., Schwarz, K., Sorantin, P. & Trickey, S. B. (1990). *Comput. Phys. Commun.* **59**, 399–415.
Blum, V., Gehrke, R., Hanke, F., Havu, P., Havu, V., Ren, X., Reuter, K. & Scheffler, M. (2009). *Comput. Phys. Commun.* **180**, 2175–2196.
Dovesi, R., Orlando, R., Civalleri, B., Roetti, C., Saunders, V. R. & Zicovich-Wilson, C. M. (2005). *Z. Kristallogr.* **220**, 571–573.
Dovesi, R., Saunders, V. R., Roetti, R., Orlando, R., Zicovich-Wilson, C. M., Pascale, F., Civalleri, B., Doll, K., Harrison, N. M., Bush, I. J., D'Arco, P. & Llunell, M. (2009). *CRYSTAL09*. User's Manual. University of Turin, Italy.
Downs, R. T. & Hall-Wallace, M. (2003). *Am. Mineral.* **88**, 247–250.
Giannozzi, P. *et al.* (2009). *J. Phys. Condens. Matter*, **21**, 395502.
Gutow, J. (2006). *Dr Gutow's Hybrid Atomic Orbital Site*, http://www.uwosh.edu/faculty_staff/gutow/Orbitals/Cl/Cl_AOs.shtml.
Gutow, J. (2010). *Jmol Color Picker Box*. http://www.uwosh.edu/faculty_staff/gutow/Jmol_Web_Page_Maker/JmolColorPicker/Jmol%20Color%20Picker%20Test.html.
Hanson, R. M. (2008). *Jmol: an Open-Source Java Viewer for Chemical Structures in 3D*. http://www.jmol.org/.
Hanson, R. M. (2009). *The Jmol Voxel (JCVXL) File Format*, http://chemapps.stolaf.edu/jmol/docs/misc/JVXL-format.pdf.
Hanson, R. M. (2010). *J. Appl. Cryst.* **43**, 1250–1260.
ICSD (2004). Inorganic Crystal Structure Database. FIZ-Karlsruhe, Germany. http://icsd.fiz-karlsruhe.de/icsd/.
Kokalj, A. (1999). *J. Mol. Graph. Model.* **17**, 176–179.
Kokalj, A. (2003). *Comput. Mater. Sci.* **28**, 155–168.
Kresse, G. & Furthmüller, J. (1996). *Phys. Rev. B*, **54**, 11169–11186.
Kresse, G. & Hafner, J. (1993). *Phys. Rev. B*, **47**, 558–561.
Lal, R. & Chava, L. C. (2010). *Developing Web Widget with HTML, CSS, JSON and AJAX: A Complete Guide to Web Widget*. Paramount: CreateSpace.
Laursen, O. (2010). *Flot: a Javascript Plotting Library for jQuery*. http://code.google.com/p/flot/.
Martz, E. (2009). *MolviZ.Org Molecular Visualization Resources*. http://www.umass.edu/microbio/chime/.
McMahon, B. & Hanson, R. M. (2008). *J. Appl. Cryst.* **41**, 811–814.
Noel, Y. (2008).*Vibration Modes on a Web Page Using Jmol: Forsterite*, http://www.pmmp.jussieu.fr/yves/vibs/interfaces/forsterite/.
Noel, Y. (2009). *Vibration Modes on a Web Page Using Jmol*, http://www.theochem.unito.it/crystal_tuto/mssc2008_cd/tutorials/webvib/index.html.
Noel, Y. & Demichelis, R. (2009). *Nanotube Systems*, http://www.theochem.unito.it/crystal_tuto/mssc2008_cd/tutorials/nanotube/nanotube.html.
Oracle (2010). *Code Samples and Apps: Applets*, http://java.sun.com/applets/.
Sayle, R. A. & Milner-White, E. J. (1995). *Trends Biochem. Sci.* **20**, 374–376.
Schaftenaar, G. & Noordik, J. H. (2000). *J. Comput. Aid. Mol. Des.* **14**, 123–134.
Schwarz, K. & Blaha, P. (2003). *Comput. Mater. Sci.* **28**, 259–273.
Searle, B. G. (2001). *Comput. Phys. Commun.* **137**, 25–32.
Segall, M. D., Lindan, P. J. D., Probert, M. J., Pickard, C. J., Hasnip, P. J., Clark, S. J. & Payne, M. C. (2002). *J. Phys. Condens. Matter*, **14**, 2717–2744.
Ugliengo, P. (2006). *MOLDRAW: a Program to Display and Manipulate Molecular and Crystal Structures*. http://www.moldraw.unito.it.
Ugliengo, P., Borzani, G. & Viterbo, D. (1988). *J. Appl. Cryst.* **21**, 75.
Ugliengo, P., Viterbo, D. & Chiari, G. (1993). *Z. Kristallogr.* **208**, 383.