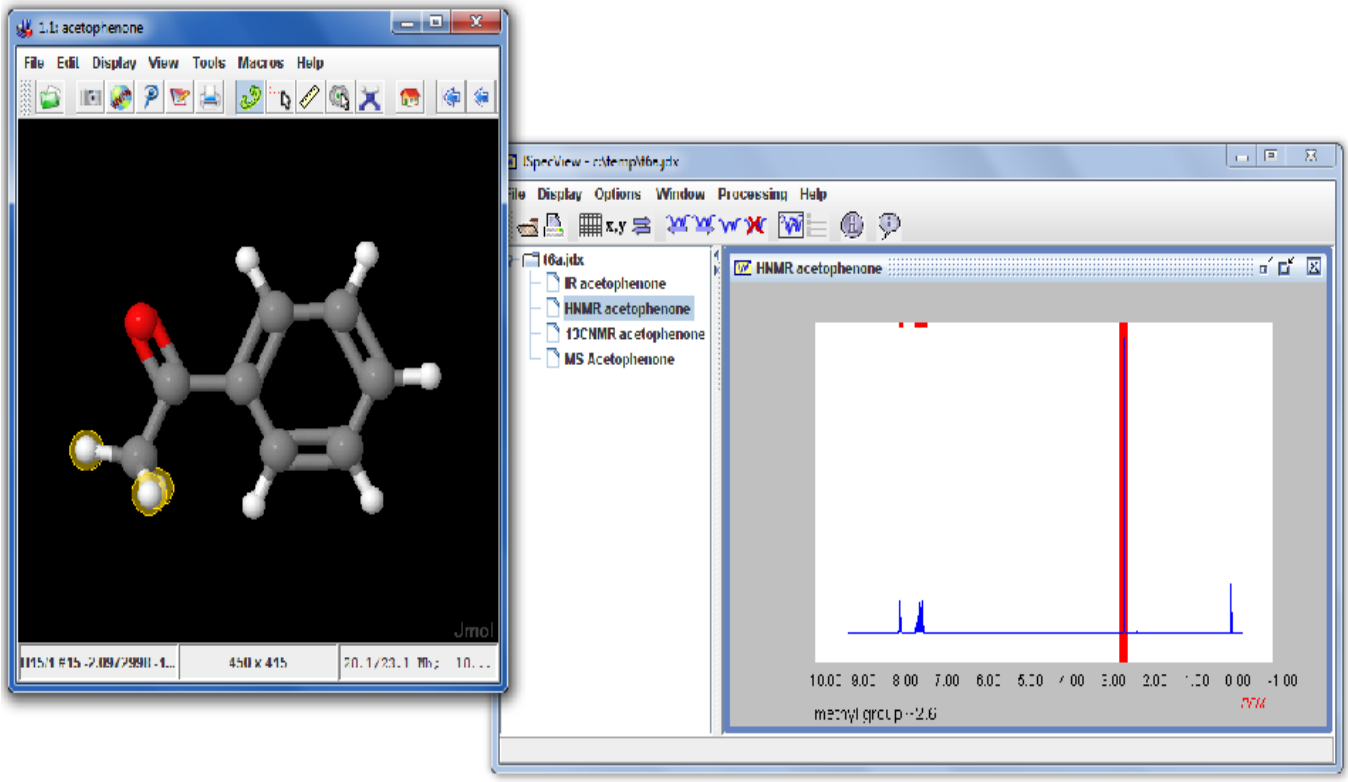
JCAMP-MOL: A JCAMP-DX extension to allow interactive model/spectrum exploration using Jmol and JSpecView

Specification Draft 4/1/2012

Robert M. Hanson hansonr@stolaf.edu
Robert J. Lancashire robert.lancashire@uwimona.edu.jm

draft modified 4/1/2012 -- xUnits, yUnits removed from <Peaks>; yMin, yMax removed from <PeakData>

for discussion



This specification describes a simple extension to the JCAMP-DX format using two user-defined-data-labels, ##$MODELS and ##$PEAKS, to add 3D Jmol-readable models to the file and also associate spectral bands with specific IR and RAMAN vibrations, MS fragments, and NMR signals. It can be applied to any sort of JCAMP-DX files, especially BLOCK files that contain multiple spectra of diverse types, but also simple JCAMP-DX files and files containing NTUPLE data such as 2D NMR data or real/complex (phasable) 1D NMR data.

The purpose of the JCAMP-MOL is to allow for a single file that can be read either by the stand-alone Jmol application (which now incorporates JSpecView) or by twin Jmol and

JSpecView applets on a web page. Clicking on an atom or selecting an IR/RAMAN vibration in Jmol highlights a band or peak or fragment on the spectrum. Clicking on the spectrum highlights one or more atoms, starts an IR vibration, or displays an MS fragment in Jmol.

The specification was implemented fully in Jmol 12.2.18 during February and March of 2012, and it works smoothly and flawlessly. Several demonstration sites are available:

http://chemapps.stolaf.edu/jmol/docs/examples-12/jspecview

http://wwwchem.uwimona.edu.jm/spectra/JSpecView2/sample

Data can be either real spectrometer data or simulated spectra. Especially with simulated spectra, there is huge potential for this format. It would be almost trivial to implement by web-based spectral prediction server developers in particular. The idea is to incorporate an API that allows Jmol to POST to the site a 3D MOL file and be returned a JCAMP-MOL file. With Jmol 12's ability to interface with the NCI CACTUS site, which can generate an uncountable number of organic compounds from a name, the potential is there to generate simulated spectra on the fly with interactive correlation to a 3D model on a web page.

The specification is simple enough that this sort of automated API should be able to be implemented very easily by server developers.

The added ##$MODELS records generally contain one MOL file (for NMR) or, for IR or RAMAN, a multi-model XYZ file with vibration information with an optionally associated MOL file that specifies bonding, or for MS a set of MOL files, one for each fragment.

The added ##$PEAKS records simply indicate which bands on a spectrum correlate with which models (IR, RAMAN, MS) or model atoms (NMR) in the ##$MODELS record(s).

Detailed specifications follow, with examples:

1a) There can be any number of ##$MODELS records in a JCAMP-MOL file.

1b) There can be at most one ##$PEAKS record per spectrum.

2) ##$MODELS and ##$PEAKS records take the form of simple XML data and may appear anywhere after the ##DATA_TYPE record of a block. These records contain line-formatted XML data that is listed using <Xxxx> and </Xxxx> tags on lines by themselves and <Xxxx … /> tags on single lines by themselves.

##$MODELS

```
##$MODELS=
<Models>
 <ModelData id="acetophenone" type="MOL">
acetophenone
  DSViewer             3D                         0

 17 17  0  0  0  0  0  0  0  0999 V2000
   -1.6931 0.0078     0.0000 C   0  0  0  0  0  0  0  0  0  1
...
M  END
  </ModelData>
 <ModelData id="1" type="XYZVIB" baseModel="acetophenone" vibrationScale=".1">
17
1  Energy: -1454.38826  Freq: 3199.35852
C            -1.693100               0.007800 0.000000   -0.000980              0.000120 0.000000
...
17
2  Energy: -1454.38826  Freq: 3191.02824
C            -1.693100               0.007800 0.000000   -0.000020   -0.000100           0.000000
...
  </ModelData>
</Models>
```

3) Each ##$MODELS record must have exactly one <Models> element. This element may start on the ##$MODELS line, but stylistically is better placed on the next line.

4) Each <Models> element may include any number of <ModelData> elements. Actual model file data is contained in the <ModelData> element, starting always with the next line. This data is line-based until the closing </ModelData> tag.

5) Each <ModelData> element must have an **id** attribute, which is used to identify this model within the file. Optional additional attributes include:

type (the format of the model data, such as XYZ or MOL; default is to let Jmol decide the type,

baseModel (a reference to a ModelData id used by Jmol only for assigning bonds for XYZ file data), and

vibrationScale (used by Jmol to adjust the vibration scale, usually to smaller magnitudes -- vibrationScale="0.1" -- than what is in the XYZ file itself).

6) Acceptable model types include any Jmol-readable file format, including for example, MOL, MOL2, PDB, CIF, and XYZ (aka XYZVIB). Any non-binary output type readable by Jmol is acceptable, including full Gaussian output log files or Web-MO models, with orbitals.

7) Generally no two models in a JCAMP-MOL file should have the same **id**. If they do, Jmol will read only the first and will ignore the second. This is fine as long as the model is the same -- no

error is thrown. So, for example, having independent models in NMR and IR sections both with id="acetophenone" is fine, because we don't need two copies of that file loaded by Jmol. Only the first will be loaded.

8) A model may be implicitly “global” (referenced by ##$PEAKS record in more than one block), or implicitly “local” (referenced only by the ##$PEAKS record of the block containing the model. Jmol does not care whether a model is global or local. The peaks assignments are correlated with models only after all models and peaks have been read.

9) If two spectrum blocks (for example, HNMR and CNMR) use the same model, they should indicate that with the same **model** attribute in the ##$PEAKS section.

10) If blocks are to be totally independent (author’s choice), then each block needs to have unique model id values, provided the models are different.

11) Alternatively, all the models could be up front in a LINK block or first data block, and data blocks can then all refer to that set of models.

12) When there is a <ModelData> record with multiple molecular models within it (for instance a set of IR vibrations in an XYZVIB file format or a set of MS fragments in a multi-model SDF format), then these models are identified by added extensions .1, .2, .3, etc. For example, if the model id is “acetone”, then individual vibrations are referenced as “acetone.1” “acetone.2” etc.

13) It is perfectly fine if some numbered models are not referred to in ##$PEAKS records. Upon loading a file, Jmol will ignore any unreferenced models that have "." in their **id**. For example, an XYZVIB file has 45 models, but only 22 are assigned. So the unused 23 models are never loaded. The remaining models are numbered sequentially *prior to* removal of unused models, so in the end we might only have models “aspirin.1” “aspirin.5” “aspirin.11” etc., with missing numbers.

##$PEAKS

```
##$PEAKS=
<Peaks type="IR">
<PeakData id="1" title="asymm stretch of aromatic CH group (~3100 cm-1)" peakShape="broad" model="1.1" xMax="3121" xMin="3081" />
<PeakData id="2" title="symm stretch of aromatic CH group (~3085 cm-1)" peakShape="broad" model="1.2" xMax="3101" xMin="3071" />
<PeakData id="3" title="asymm stretch of CH group (~3060 cm-1)" peakShape="broad" model="1.3" xMax="3077" xMin="3047" />
…
</Peaks>
```

14) ##$PEAKS records contain exactly one <Peaks> element. The <Peaks> tag must have at least the attribute **type**. **type** is an author-defined descriptor, for example: “HNMR” or “13CNMR” or “GC/MS” or “MS” or “IR” or “DEPT-45”. Additional attributes include **xLabel** and **yLabel**,

which indicate the x- and y-scale labels and override ##XLABEL and ##YLABEL (which themselves override ##XUNITS and ##YUNITS).

15) Each <Peaks> element may contain any number of <PeakData.../> tag, one per line. The <PeakData.../> tag attributes specify the correlation between spectrum positions and models and (possibly) specific atoms in those models.

<PeakData id="1" title="methyl group ~2.6" peakShape="singlet" model="acetophenone" atoms="15,16,17" xMax="2.7" xMin="2.5" />

16) Jmol/JSpecView specifically use the attributes **title**, **model**, **atoms**, **xMin**, and **xMax**. All others are currently ignored.

title	a phrase to be associated with this model or group of atoms
model	a model **id**
atoms	particularly for NMR, the atoms (1-based numbering) associated with this signal. (See example below)
xMin	low value for range on spectrum. Units are specified in the <Peaks> tag and should be the same as given for ##XUNITS in the data.
xMax	high value for range on spectrum. Units are specified in the <Peaks> tag and should be the same as given for ##XUNITS in the data.

17) Jmol and JSpecView will add three attributes to the beginning of this tag: **file**, **index**, and **type**, when communicating. These names are reserved names and should not be used by the author of the JCAMP-DX file. They are, respectively, the file path or URL of the source file, a 1-based serial index of the <PeakData.../> tag in the file, and the type attribute from the Peaks tag.

<end of specification>

Intra-Application Communication

Jmol (and any Java application incorporating Jmol) fully incorporates JSpecView starting with Jmol 12.3.18. JSpecView can be opened independently from the Tools menu using the command **sync JSpecView**. In addition, JSpecView will be opened automatically if Jmol has read a JCAMP-MOL file of this sort and has been given and the user navigates to an IR vibration or MS fragment model or clicks on an atom having an associated peak in the NMR spectrum. The **sync** command can also be used to send commands to JSpecView. For instance, **sync JSpecView: load c:/temp/t.jdx**.

A command starting with **hidden true;...** will execute in JSpecView without bringing up the JSpecView window. This allows Jmol scripts direct behind-the-scenes access to spectroscopic data by selecting specific spectra and, for example, integrating spectra.

Starting with Jmol 12.3.18, Jmol's **getProperty** command and **getProperty()** function are extended to obtain information from JSpecView (application only). Thus, **getProperty JSpecView** gets a large list of data relating to currently open spectra, and

```
print getProperty("JSpecView")["items"][1]["spectra"][1]["header"]["##title"]
```

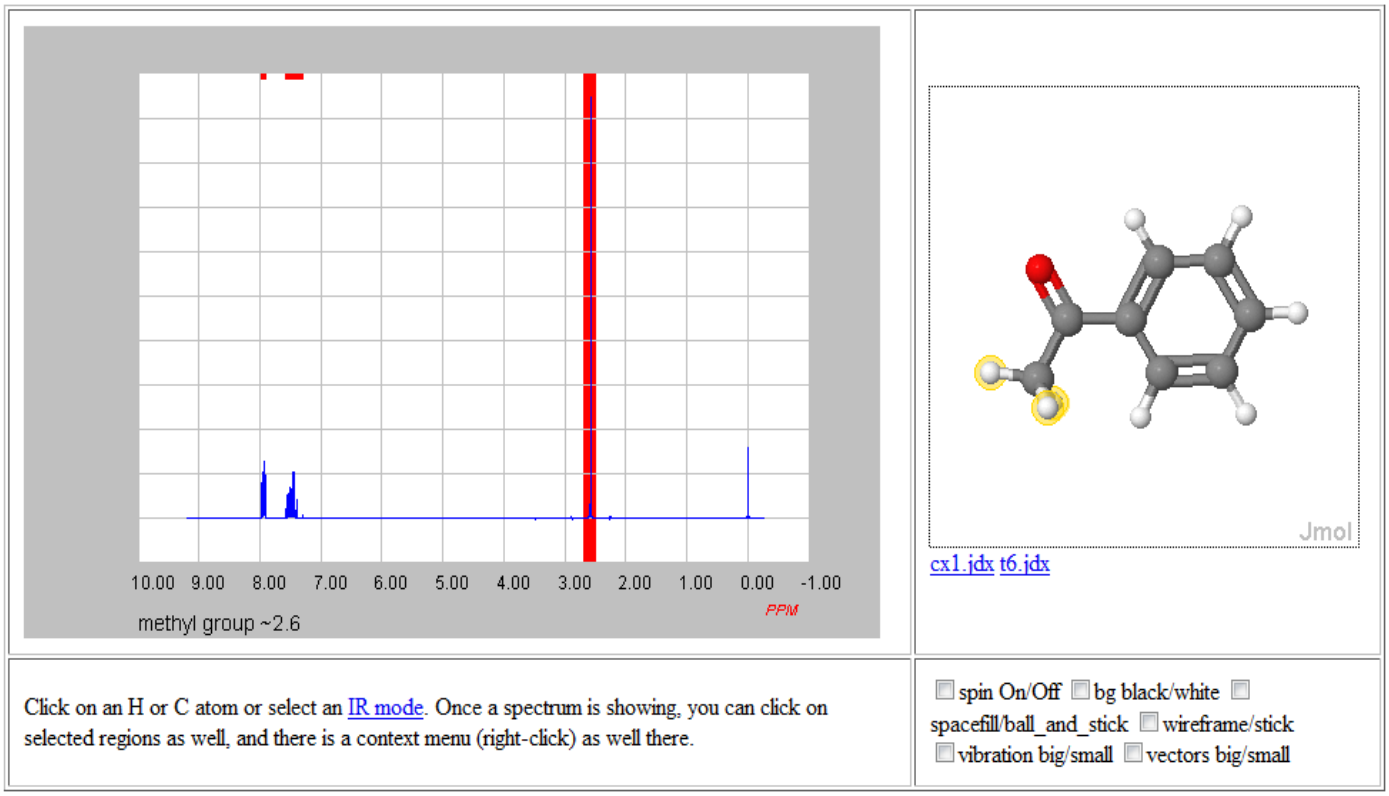
selects out a piece of that larger corpus. A more efficient way of getting this selective information is by indicating the key of interest in the getProperty command:

```
print getProperty("JSpecView", "##TITLE")["items"][1]["spectra"][1]["header"]["##title"]
```

In this case, JSpecView does not compile the entire set of parameters, but instead looks for that specific key in the header (of all spectra).

Clicking on an identified peak on any spectrum in the JSpecView window will load the specified model into Jmol and highlight it. If another model is already loaded in Jmol, it will be replaced. The spectrum can also be navigated using the left- and right-arrow keys of the keyboard.

Applet-Applet Communication



http://chemapps.stolaf.edu/jmol/docs/examples-12/jspecview/test1.html

The Jmol and JSpecView applets will communicate via Jmol's SYNC command, sending to each other the complete <PeakData.../> tag (including added **file**, **index**, and **type** attributes) preceded with "Select: " in the case of JSpecView-->Jmol and "XXXX__yyyy__JSpecView:" in the case of Jmol-->JSpecView. (The designation "XXXX__yyyy__" here is the full applet name for a Jmol applet, including its standard name attribute, two underscore characters, its pseudo-random syncID, and two underscores. For example: jmolApplet0__2938373623__).

The web page developer needs to include **SYNC ON; SET SYNCCALLBACK "xxxxx"** in Jmol and **SET SYNCCALLBACKFUNCTIONNAME xxxxx** in JSpecView, where xxxxx is a JavaScript callback function name. This JavaScript function is responsible for receiving the <PeakData.../> tag information (as the second parameter) and passing that along to the appropriate JSpecView or Jmol applet using that applet's public syncScript() method, possibly by checking the **type** attribute to determine where to route the sync request. AppletReadyCallbacks are also recommended, since the two applets will load simultaneously, and it is important that both be fully loaded prior to initiation of communication. Sample code is on the next page.

Note that the following code is using functionality provided by JSVfunctions.js and Jmol.js.

```
<javascript>
haveJmol = false
haveJsv = false

function readyCallback(app, msg, isReady) {
  if (!isReady)return
  if (app.indexOf("jmolApplet") == 0)
    haveJmol = true
  else
    haveJsv = true
}


function mySyncCallback(app,msg) {
  var toJmol = (haveJmol && msg.indexOf("Select:") == 0)
  var toJsv = (haveJsv && msg.indexOf("JSpecView:") >= 0)
  if (!toJmol && !toJsv) return 1 // some other sort of sync, or not ready
  _jmolFindApplet(toJmol ? "jmolApplet0" : "JSVApplet").syncScript(msg)
  return 0 // prevents further Jmol sync processing
}

</javascript>
```

And later in the file, to create the JSpecView and Jmol applets:

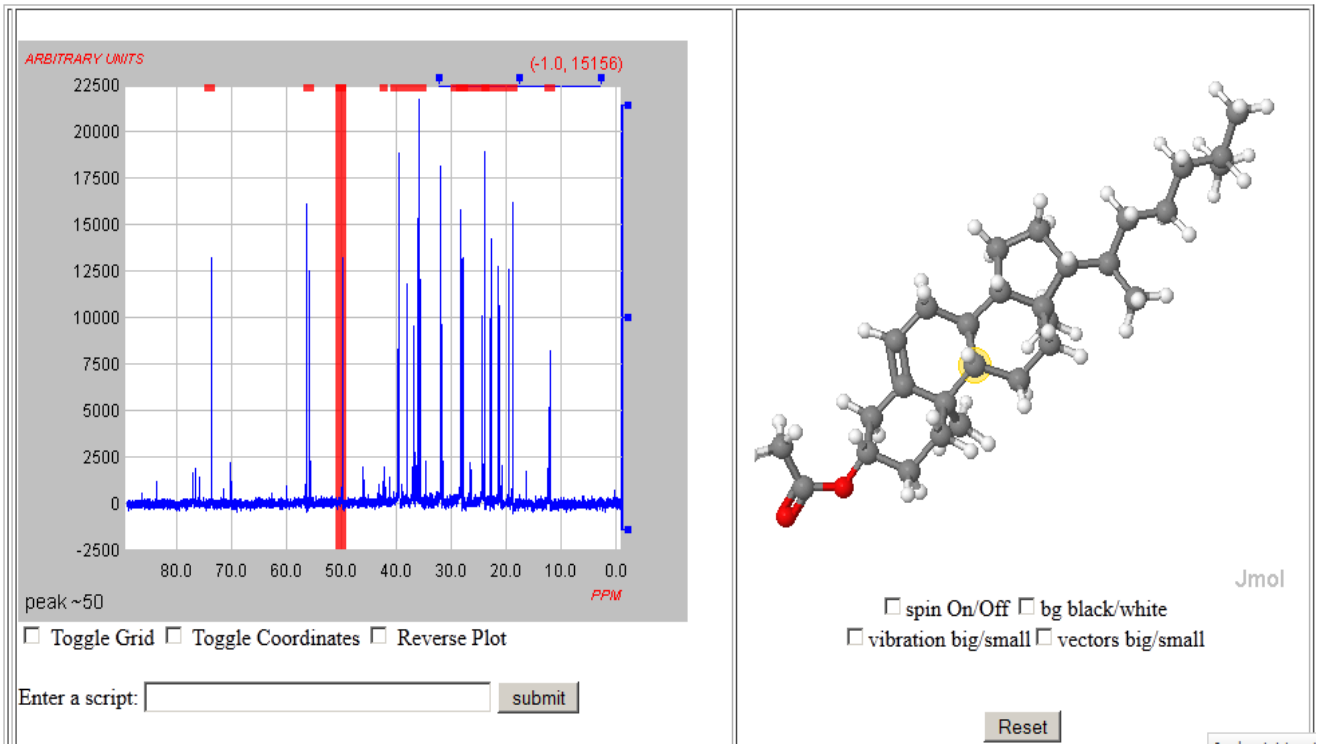
```
<javascript>
  jsvls="appletreadycallbackfunctionname readyCallback;synccallbackfunctionname mySyncCallback;load t6.jdx;";
  insertJSVObject("../jspecview.jar","JSVApplet","600","400",jsvls);

</javascript>

  <javascript>
              _jms= "load t6.jdx;background white; vibration off; vectors off;sync on";
              jmolSetCallback("syncCallback", "mySyncCallback");
              jmolSetCallback("appletReadyCallback", "readyCallback");
               jmolApplet(300,_jms,"0");
</javascript>
```

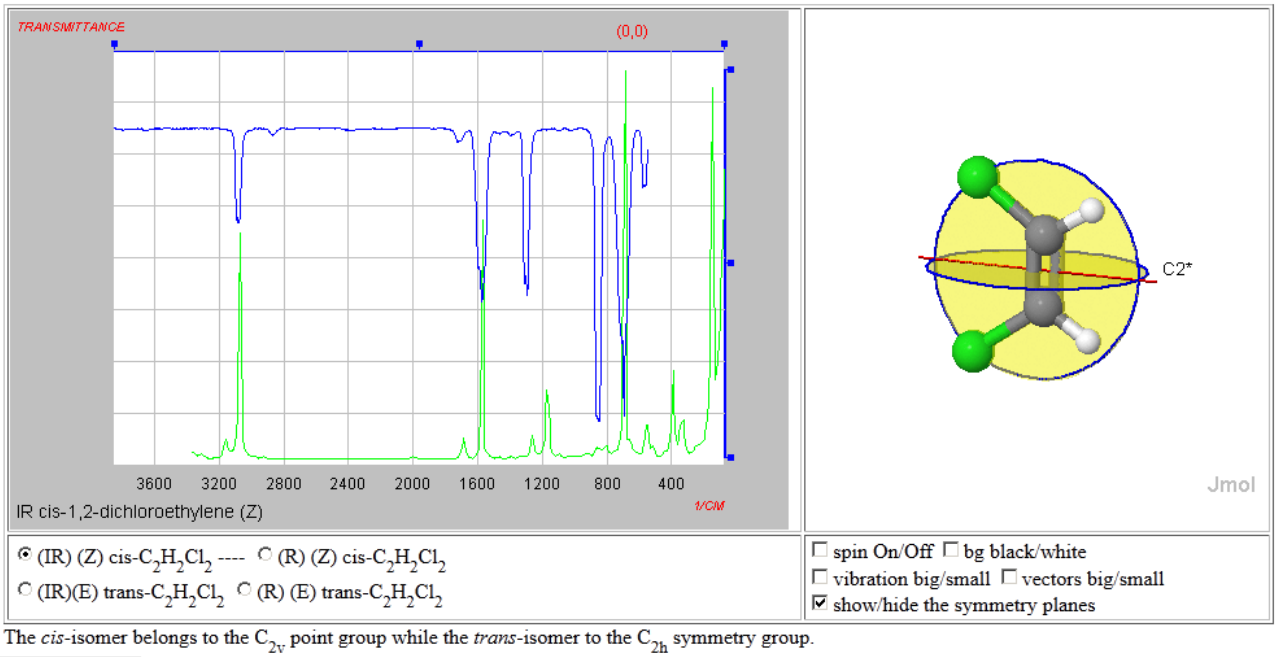
In conclusion, we have developed a simple method of specifying spectral-model correlations. The method allows reading of a single file by either Jmol or JSpecView and, with a bit of additional JavaScript on a web page, allowing interactive investigation of a molecule and its associated spectral data.

Some sample screenshots:
Cholesteryl acetate 13C NMR



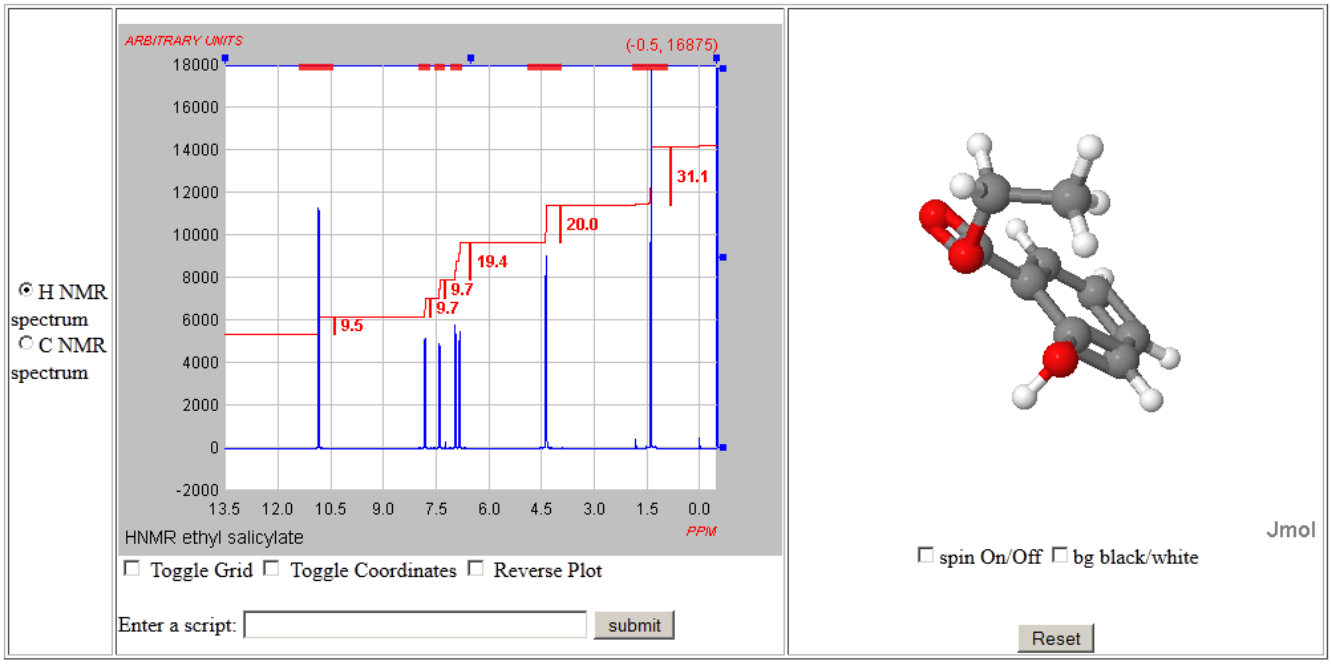
cis- and trans-1,2-dichloroethylene study with IR and Raman spectra

Interpretation of the IR and Raman spectra of the (Z) *cis*- and (E) *trans*- isomers of 1,2-dichloroethylene



The *cis*-isomer belongs to the C_{2v} point group while the *trans*-isomer to the C_{2h} symmetry group.

H NMR of ethylsalicylate (with integration feature)



The spectra are "hot-linked" to the molecular graphic display such that selecting a peak in the NMR spectra will cause the appropriate H or C atoms to be highlighted.

For devices that can make use of HTML5 and not Java, it is possible to use JSpecView to export a SVG document that can be inserted into a web page.

Export your JCAMP-DX file as a SVG from JSpecView
then insert the file into the body of the web page
Most browsers will now display this OK, except perhaps Opera and the native Andoid browser but Firefox on Android works..