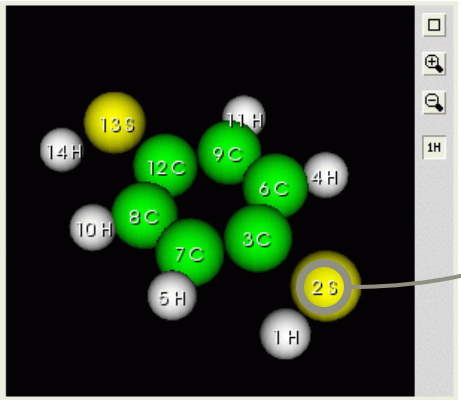
Advanced Visualization

Michael McLennan

HUBzero® Platform for Scientific Collaboration

Purdue University

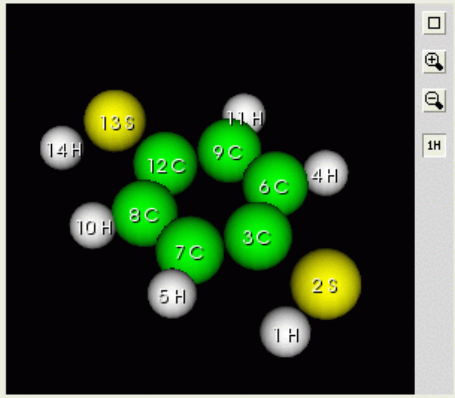
Include atoms/molecules



Turns atom labels on by default

```
<structure id="mol">
  <current>
    <components>
      <molecule>
        <about><emblems>on</emblems></about>
        <formula>pdt</formula>

        <atom id="0">
          <symbol>H</symbol>
          <xyz>-1.24935 -3.41562 0.0</xyz>
        </atom>
        <atom id="1">
          <symbol>S</symbol>
          <xyz>0.08092 -3.19426 0.0</xyz>
        </atom>
        …
      </molecule>
    </components>
  </current>
</structure>
```

Include atoms/molecules




```
<structure id="mol">
  <current>
    <components>
      <molecule>
        <pdb>ATOM      1   C    0.000   0.000   0.000
ATOM      2   C    1        0.000   0.000
ATOM      3   C    0        1.000   0.000
ATOM      4   C    0        0.000   1.000
        </pdb>
      </molecule>
    </components>
  </current>
</structure>
```

Generate within your program like this:

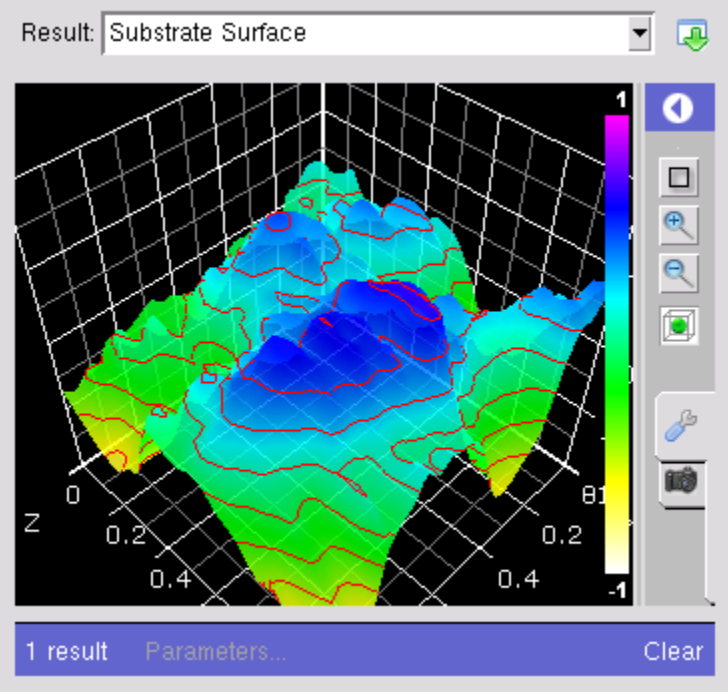
Fortran F77

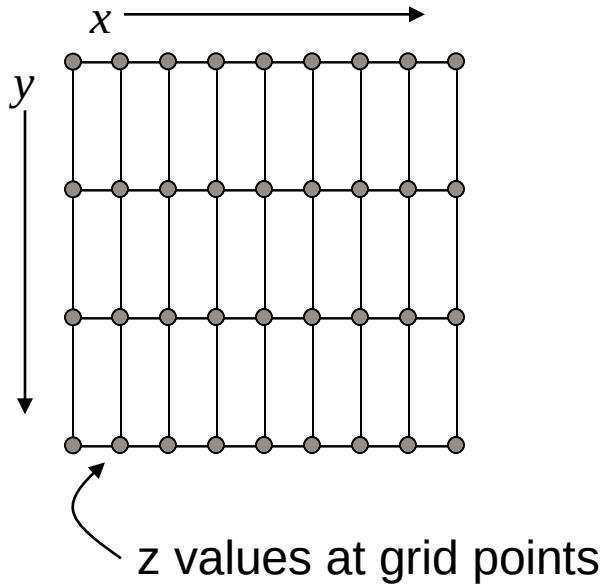
```
      call rp_lib_put_file(io,
     +    "output.(mol).current.components.molecule.pdb",
     +    "data.pdb", 1, 0)
```

0 = overwrite (not append)

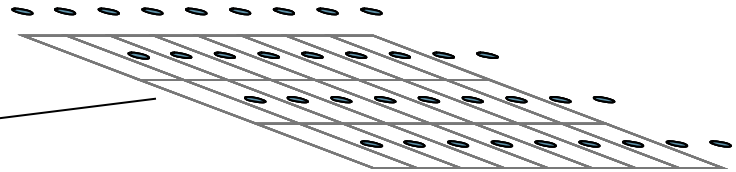
1 = compress data

Generate surface plots and contour plots





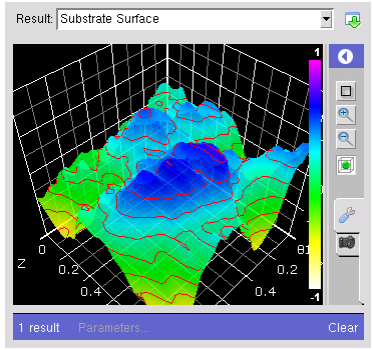

```
<field> = values
<unirect2d> = 2D mesh
```



Two separate objects, so you can use the same mesh with many fields

Should look like this in the run.xml file:

```
<output>
  <unirect2d id="mygrid">
    <about> <label>Energy Grid</label> </about>
    <xaxis>
      <label>Fermi-Dirac Factor</label>
      <min>0.0</min>
      <max>1.0</max>
      <numpoints>50</numpoints>
    </xaxis>
    <yaxis>
      <label>Energy</label>
      <min>0.0</min>
      <max>1.0</max>
      <numpoints>50</numpoints>
    </yaxis>
  </unirect2d>
  …
```

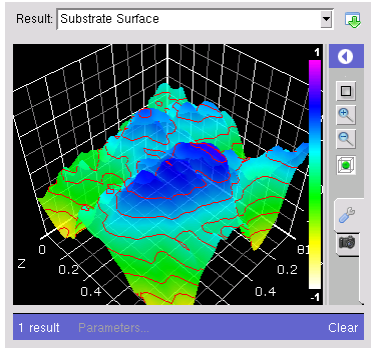


Generate within your program like this:

```
io.put('output.unirect2d(mygrid).xaxis.max', '1.0', append=0)
io.put('output.unirect2d(mygrid).xaxis.numpoints', '50', append=0)
```

Add field object to the run.xml file:

```
<output>
  <unirect2d id="mygrid">
     …as shown on previous page…
  </unirect2d>

  <field id="z">
    <about>
      <label>Substrate Surface</label>
    </about>
    <component>
      <mesh>output.unirect2d(mygrid)</mesh>
      <values>1.4358794e-01 1.1341028e-01 …
1.0426426e-01 1.1974895e-01 1.4802129e-01 …
1.5492613e-01 1.6212342e-01 1.8821293e-01 …
      </values>
    </component>
  </field>
```



Y-index varies fastest

v(1,1) v(1,2) v(1,3) …
v(2,1) v(2,2) v(2,3) …
v(3,1) v(3,2) v(3,3) …

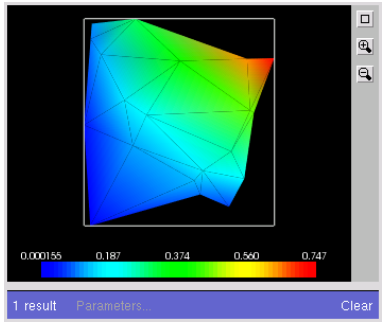
Generate within your program like this:

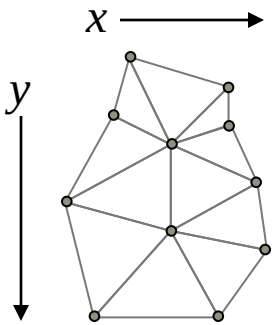
```
vals = reshape(z', npts*npts, 1);
str  = sprintf('%12g\n', vals);
rpLibPutString(io,'output.field(z).component.values', str,0);
```

Irregular mesh:

```
<output>
  <cloud id="m2d">
    <about>
      <label>2D Mesh</label>
    </about>
    <units>um</units>
    <hide>yes</hide>
    <points>0.1251300601 0.06092156519
0.9087461861 0.2971503824
0.2064777497 0.2715395983
0.7660297065 0.6612785154
…
    </points>
  </cloud>
…
```

x y
x y
x y
…





<cloud>

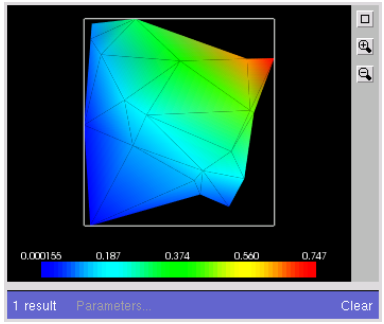
Shotgun blast of (x,y) points

Generate within your program like this:

```
for x,y in zip(xvec,yvec):
  str = '%g %g\n' % (x, y)
  io.put('output.cloud(m2d).points', str, append=1)
```

Add a field that uses the irregular mesh:

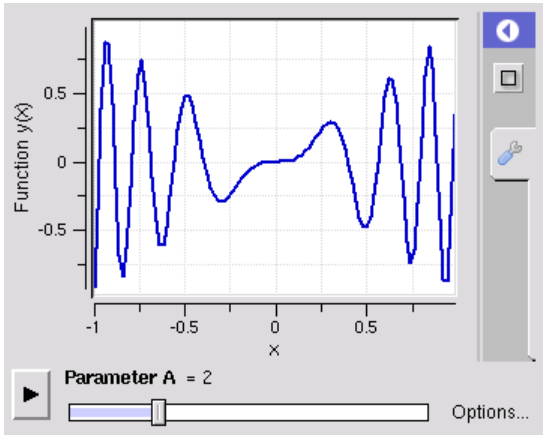
```
<output>
  <cloud id="m2d">
    <about>
      <label>2D Mesh</label>
    </about>
    <units>um</units>
    <hide>yes</hide>
    <points>0.125130060187 0.0609215651922
0.908746186136 0.297150382445
0.206477749723 0.271539598364 …
    </points>
  </cloud>

  <field id="one">
    <component>
      <mesh>output.cloud(m2d)</mesh>
      <values>1.4358794e-01
1.3522678e-01
1.2992344e-01
…
      </values>
    </component>
  </field>
```



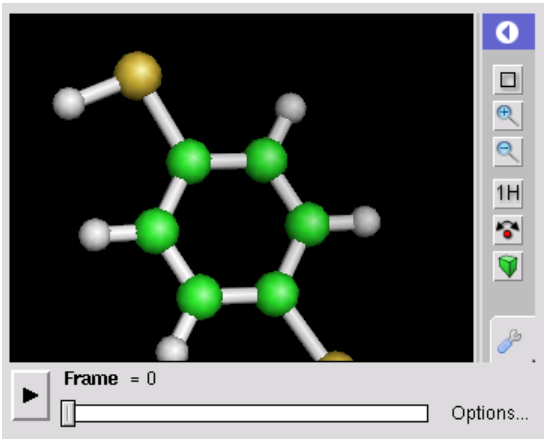
First value for first point, second value for second point, and so forth…

Time series of values -- like a movie

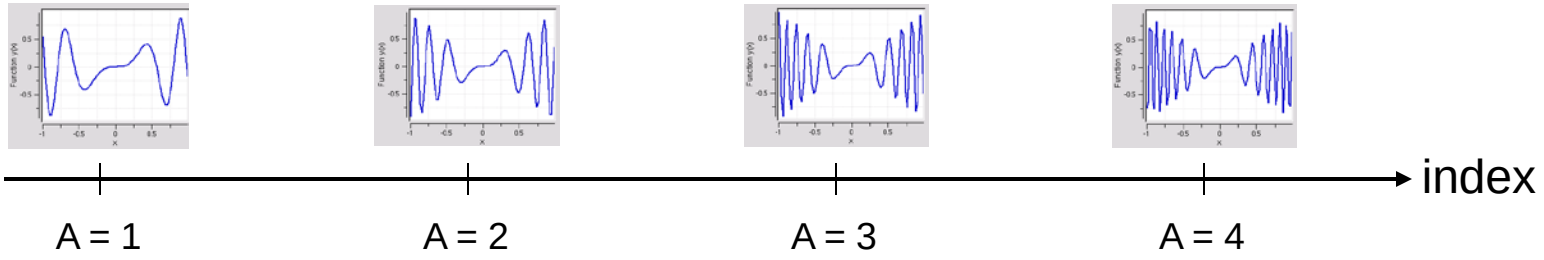
Sequence of images



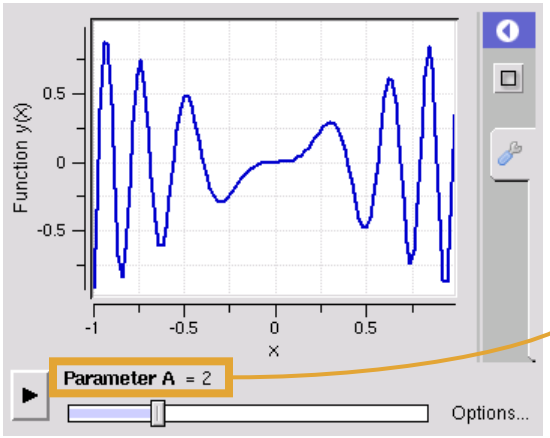
Sequence of curves



Sequence of molecules



Sequence of Curves



generate this:

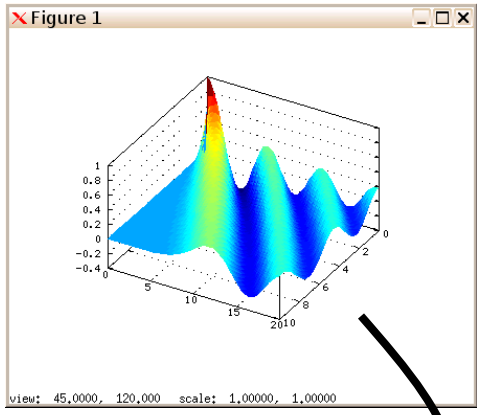
```
<output>
  <sequence id="outs">
    <about>
      <label>Sequence of Plots</label>
    </about>
    <index>
      <label>Parameter A</label>
    </index>

    <element id="1">
      <index>1</index>
      <curve>…</curve>
    </element>

    <element id=“2">
      <index>2</index>
      <curve>…</curve>
    </element>
    …
</sequence>
```

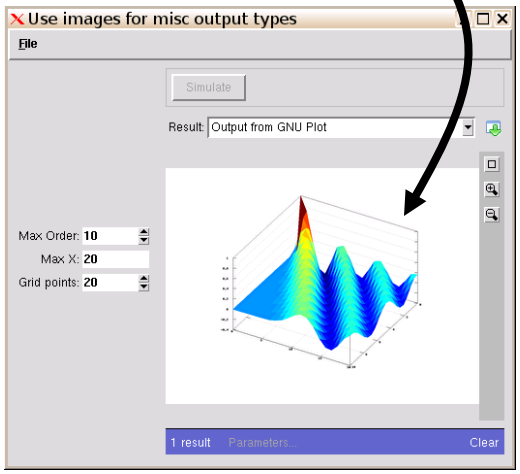
Usual curve object stuff, but inside the element along with the index

Suppose you have a program that generates a unique plot that you want to include in Rappture

In MATLAB/Octave:

```
nu = linspace(0,10,50);
x = linspace(0,20,50);
j = besselj(nu,x');

surf(nu,x',j);
shading('flat');
view(120,45);
```



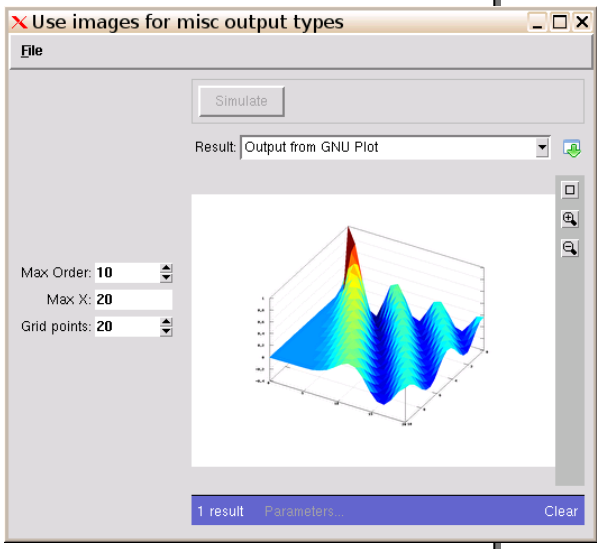


Include it in Rappture as an image

If all else fails, use an image

Octave program: main.m

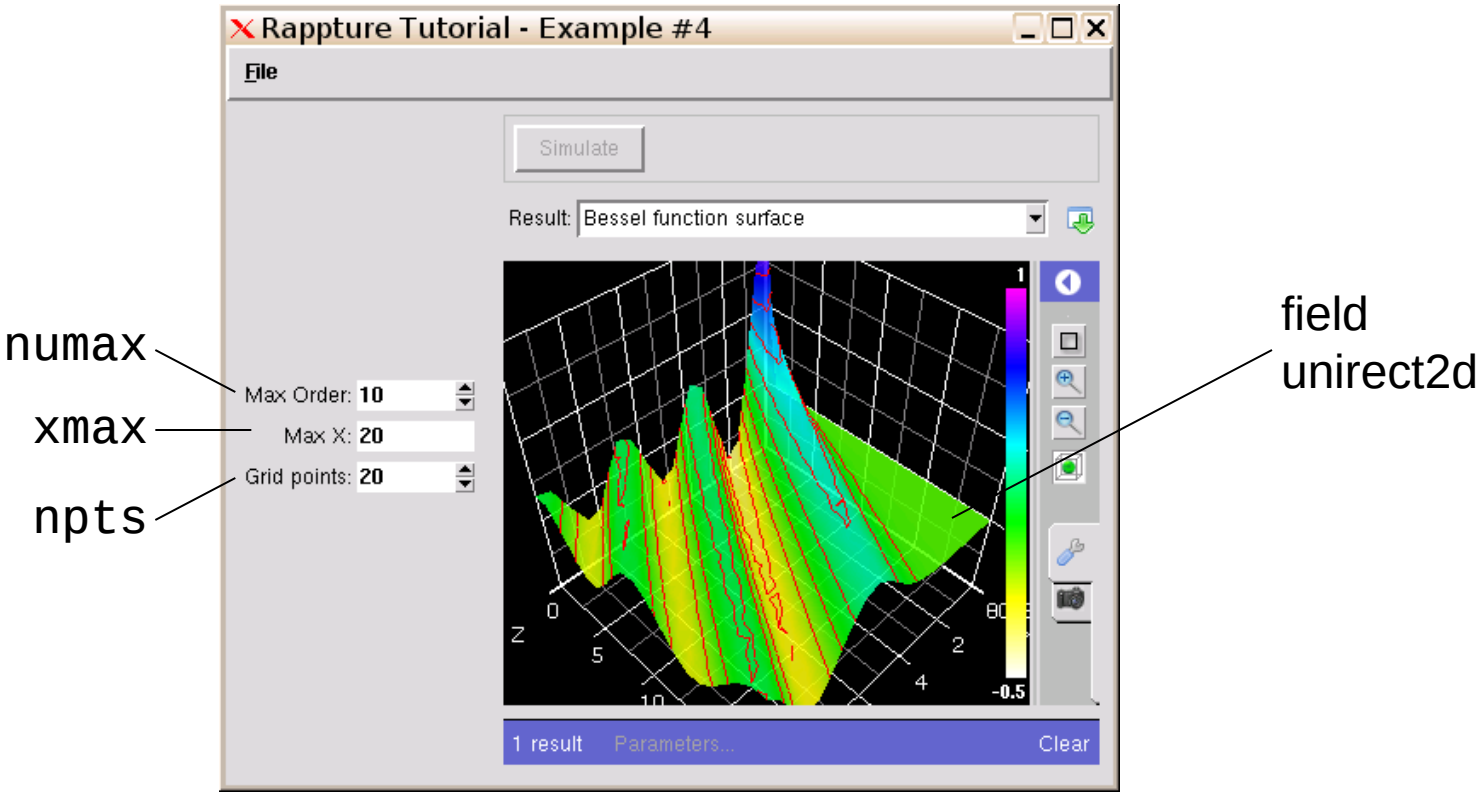
```
…
nu = linspace(0,numax,npts);
x  = linspace(0,xmax,npts);
z  = besselj(nu,x');

% keep the plot from popping up on the screen
set(gcf,'Visible','off');

% generate the surface plot
surf(nu,x',z);
shading('flat');
view(120,45);

% flush out drawing and take a snapshot
drawnow;
print -dpng output.png;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Save output values back to Rappture
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rpLibPutFile(io,'output.image(snapshot).current','output.png', 1, 0);
rpLibResult(io);
quit;
```



1 = compress image

0 = overwrite (not append)

Assignment #9: Bessel functions in Rappture



In MATLAB/Octave:

```
nu = linspace(0,numax,npts);
x = linspace(0,xmax,npts);
z = besselj(nu,x');
```