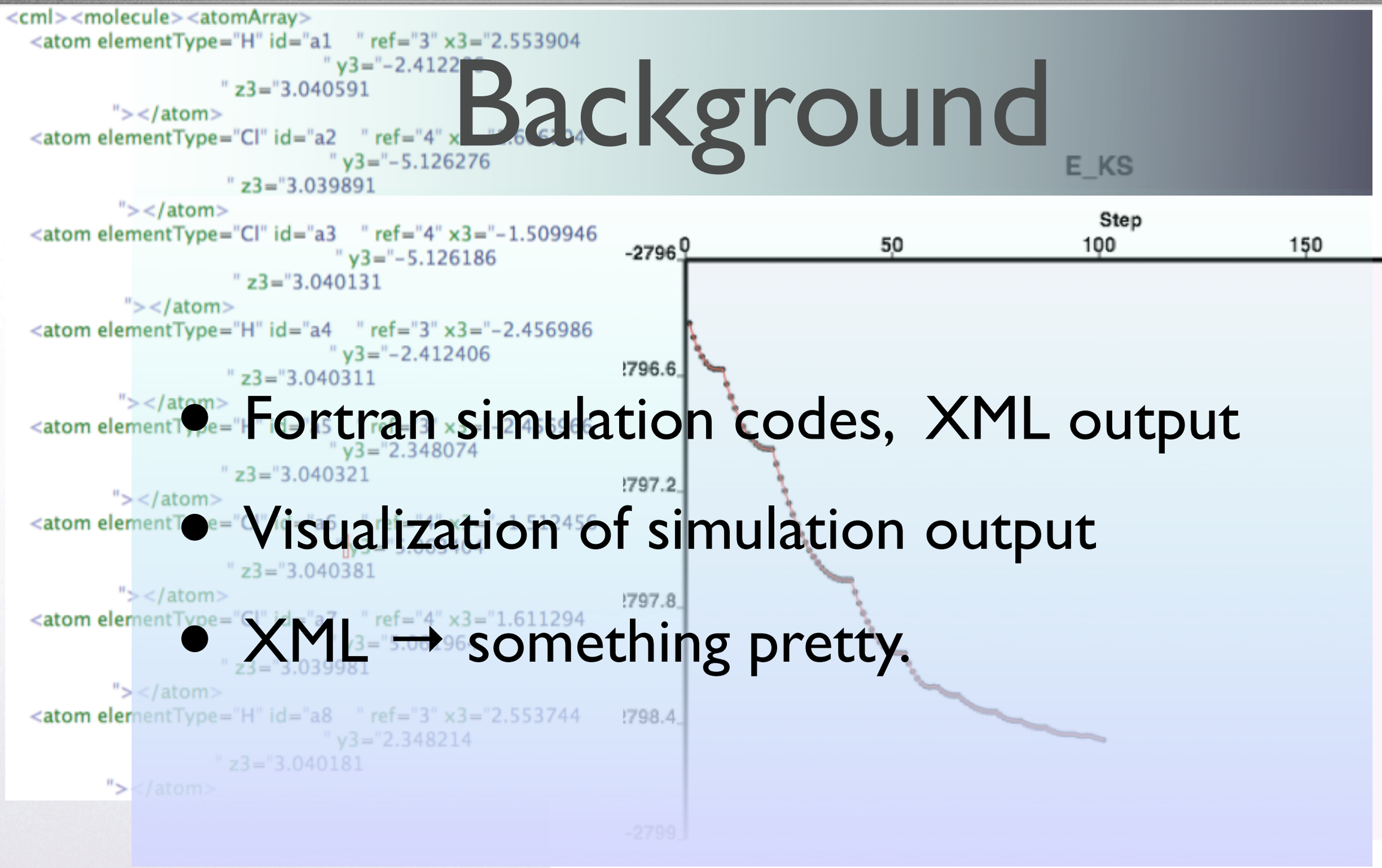
Use of SVG within a grid computing environment: the eMinerals experience

Toby White,
Richard Bruin,
Martin Dove

University of Cambridge

I: Technical Details

- Background to SVG use
- Extraction of XML data
- Pelote
- Integration into larger tool.



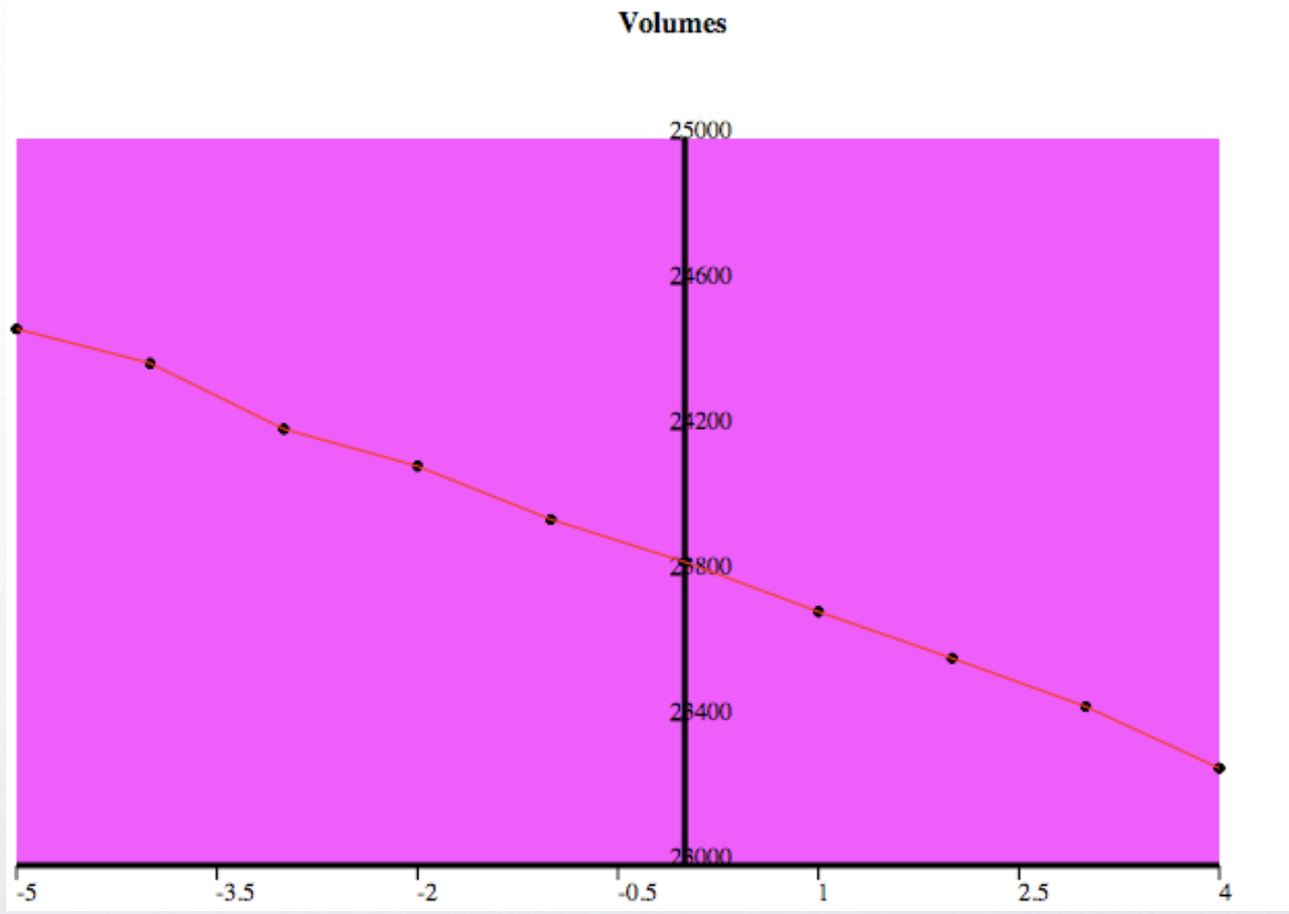
Background

- Fortran simulation codes, XML output
- Visualization of simulation output
- XML → something pretty.

Why SVG?

- 2D plots - prototypical vector graphics

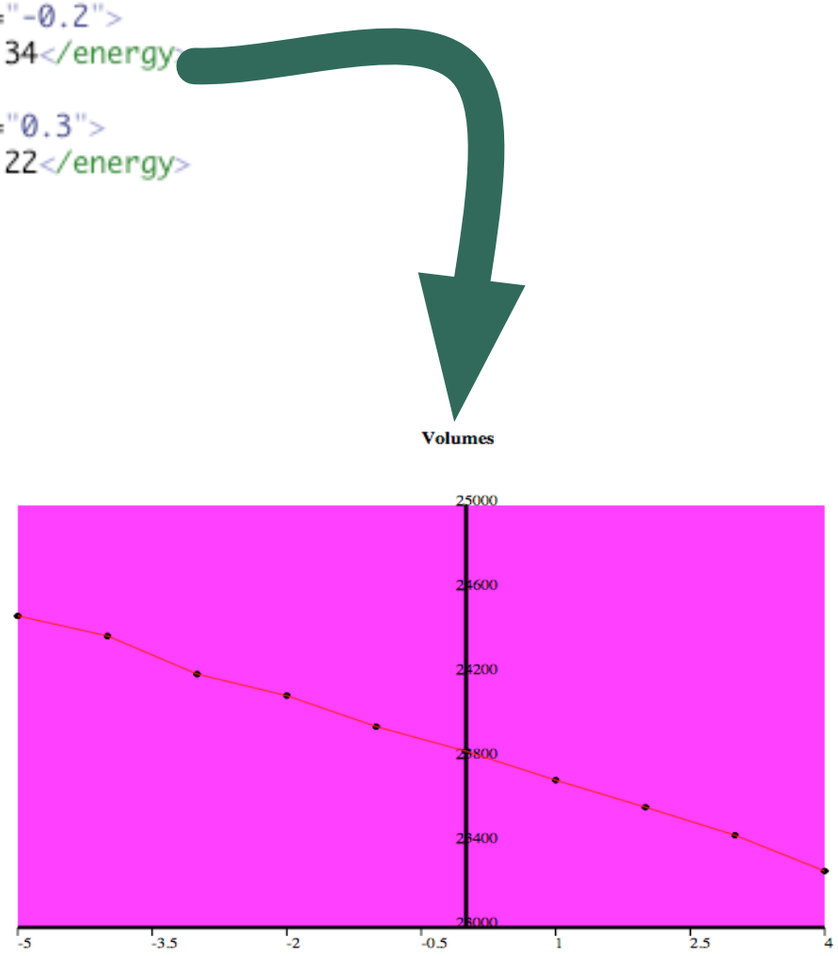
-5.0	24476.23
-4.0	24381.78
-3.0	24200.76
-2.0	24098.37
-1.0	23951.72
0.0	23835.01
1.0	23697.98
2.0	23569.92
3.0	23436.96
4.0	23267.29



XML2SVG

```
<pressure value-="-0.5">
   <energy>23356.78</energy>
</pressure>
<pressure value-="-0.2">
   <energy>23889.34</energy>
</pressure>
<pressure value-="0.3">
   <energy>24098.22</energy>
</pressure>
```

MyML



SVG Plot

- MyML → SVG?
  - Fiddly, but do-able
- Arbitrary XML ??

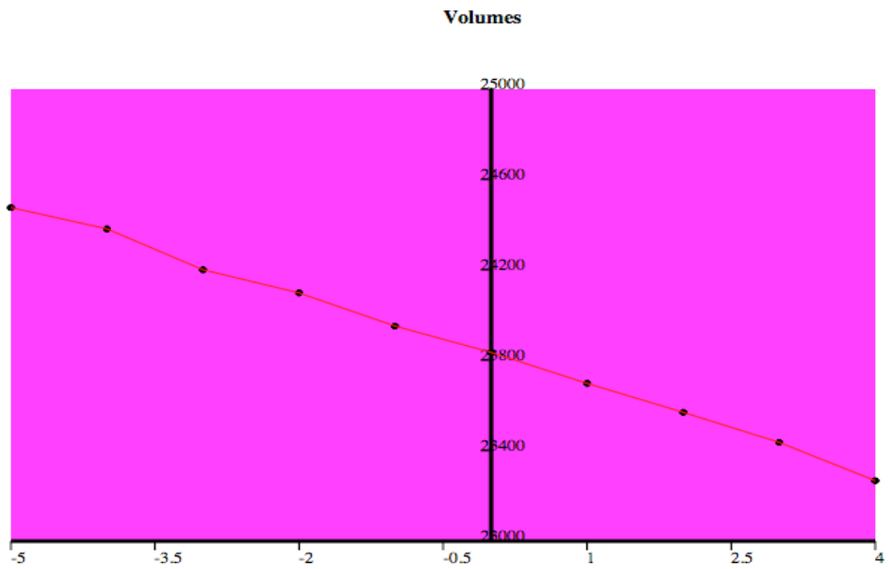
```
<pressure value-="-0.5">
   <energy>23356.78</energy>
</pressure>
<pressure value-="-0.2">
   <energy>23889.34</energy>
</pressure>
<pressure value-="0.3">
   <energy>24098.22</energy>
</pressure>
```

MyML

HisML

IR

HerML



SVG Plot

Markup Languages

- single strings, emphasis, etc. ***HTML***
  - web browser
  - braille display
  - screen reader
- 2D data?
  - HTML tables ...

Extracting data

- Hundreds of languages
- Hundreds of interfaces

Java
Javascript
SAX
Perl
DOM
Python
Xpath
...

✦ Examples in XSLT

✦ Programming with one hand tied behind back → K.I.S.S.

```
<module>
  <propertyList>
    <property dictRef="siesta:E_KS">
      <scalar units="eV">-45000</scalar>
    </property>
  </propertyList>
<module>
...
```

```
<xsl:for-each select="module">
  <xsl:value-of select="position()"/>
  <xsl:text>, </xsl:text>
  <xsl:value-of select="propertyList/property[dictRef='E_KS']/scalar/text()"/>
  <xsl:text>
</xsl:text>
</xsl:for-each>
```

```
1,   45000
2,   44677
3,   47834
...
```

```
<pointList>
  <xsl:for-each select="module">
    <point>
      <xsl:attribute name="x">
        <xsl:value-of select="position()"/>
      </xsl:attribute>
      <xsl:attribute name="y">
        <xsl:value-of select="propertyList/property[dictRef='E_KS']/scalar/text()"/>
      </xsl:attribute>
    </point>
  </xsl:for-each>
</pointList>
```

```
<pointList>
  <point x="1" y="45000"/>
  <point x="2" y="44677"/>
  <point x="3" y="47834"/>
</pointList>
```

```
<module>
  <propertyList>
    <property dictRef="siesta:E_KS">
      <scalar units="eV">-45000</scalar>
    </property>
    <property dictRef="siesta:fmax">
      <scalar units="eVpA">-5.314</scalar>
    </property>
    <property dictRef="siesta:fres">
      <scalar units="eVpA">-0.417</scalar>
    </property>
  </propertyList>
<module>
...
```

```
<xsl:for-each select="module[position()=1]/propertyList/property/@dictRef">
  <xsl:variable name="dictRef">
    <xsl:value-of select="."/>
  </xsl:variable>
  <pointList>
    <xsl:for-each select="module">
      <point>
        <xsl:attribute name="x">
          <xsl:value-of select="position()"/>
        </xsl:attribute>
        <xsl:attribute name="y">
          <xsl:value-of select="propertyList/property[dictRef=$dictRef]/scalar/text()"/>
        </xsl:attribute>
      </point>
    </xsl:for-each>
  </pointList>
</xsl:for-each>
```

```
<pointList>
...
</pointList>
<pointList>
...
</pointList>
<pointList>
...
</pointList>
```

Number formats

- XSLT, XPath, XSD 1.0:

  1.0 ✓

  +1.0, 1.0e0 ✗

  1.0E0, 1.0D+000, 1.0d0 ✗✗✗

  NaN, NaN, everywhere

Number formats

- Inconsistent standard handling:
  - xsltproc, 4suite, SAXON: all different results
- XPath, XSLT, XSD 2.0
  - Better, but still not Fortran-complete, not standardized or implemented anyway.

Number formats

Massage your numbers first!

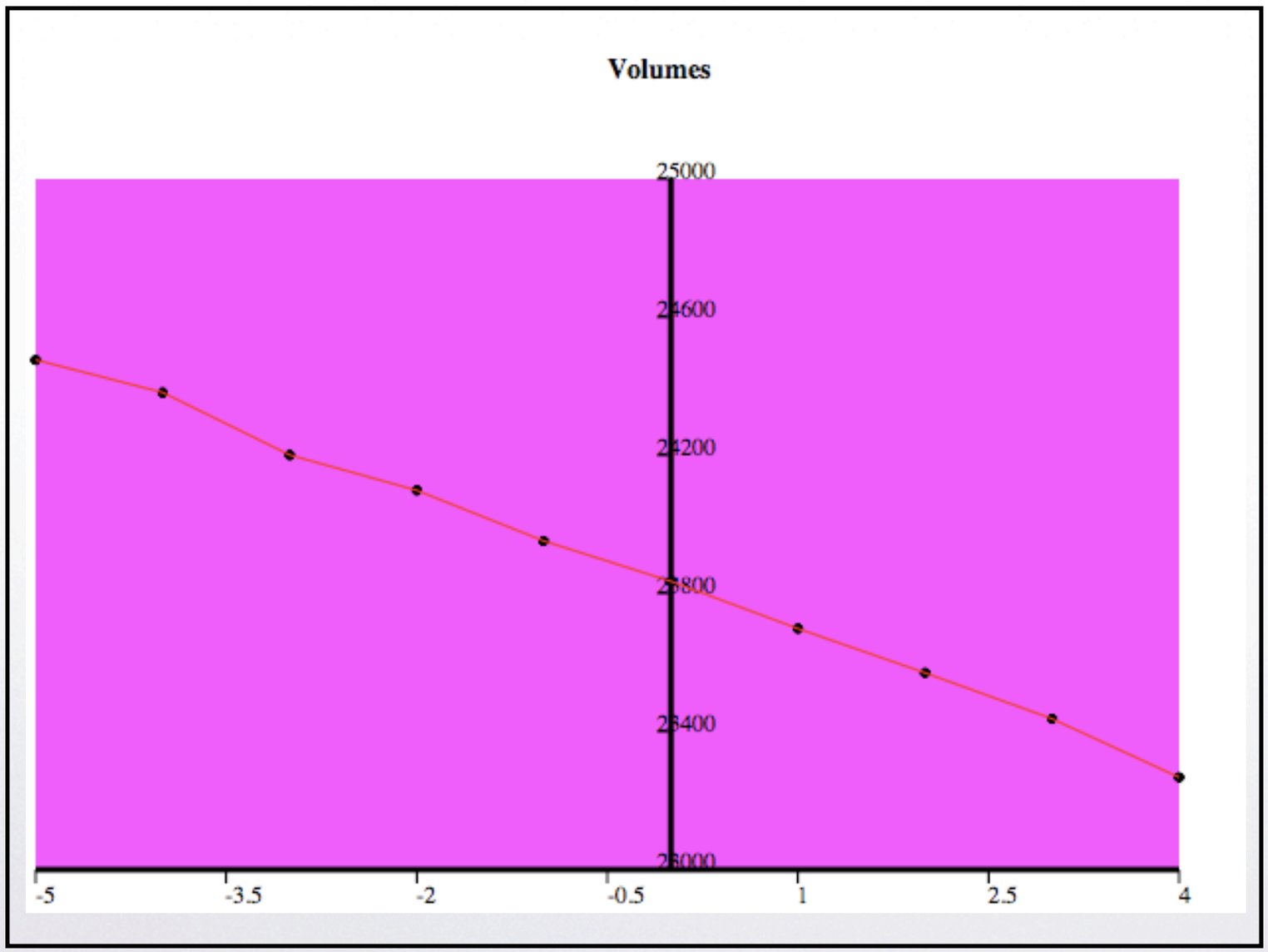
→ known textual format

★ tohw_numbers.xsl

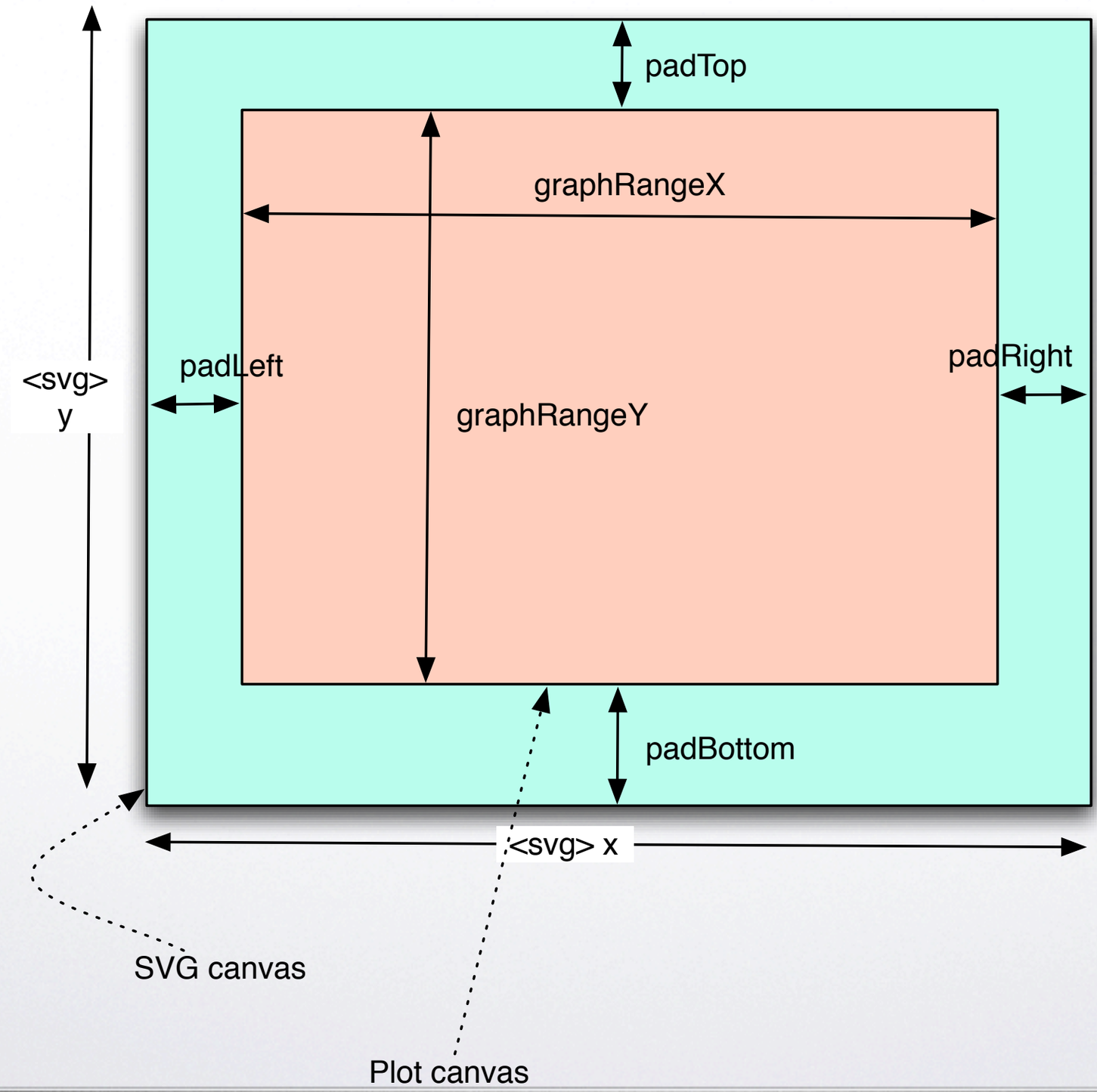
- `tohw:number()`
- `tohw:launderTree`

```
[+-]?[0-9]*(\.[0-9]*)?([dDeE][+-]?[0-9]+)?
```

SVG Anatomy




```
<svg>
<defs>
<style>
<text>
<line>
<polyline>
<rect>
<circle>
```


padTop
graphRangeX
padLeft
padRight
graphRangeY
<svg>
y
padBottom
<svg> x
SVG canvas
Plot canvas

Conversion

$$(x_{\text{svg}}, y_{\text{svg}}) = (x_0, y_0) + (x_{\text{scale}} \times x_{\text{data}}, y_{\text{scale}} \times y_{\text{data}})$$

- Get x_0, x_{scale}:
  - specify directly
  - heuristics for defaults
- Calculate x_{svg}
  - Convert by hand
  - Use `<svg:g transform=""/>` to provide scaled canvas.

Pelote

- Pelote: Intermediate Representation mentioned above
- Small XML language describing 2D data and their representation

Pelote anatomy

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"
                 href="http://www.uszla.me.uk/xsl/1.0/pelote/pelote.xsl"?>
<plot xmlns="http://www.uszla.me.uk/xsl/1.0/pelote">
  <title>Volumes</title>
  <paramSet>

    <range floorX="-10" ceilingX="10" floorY="22000" ceilingY="25000"/>
    <axis type="x" position="23000"/>
    <axis type="y" position="0">
      <tickList numTicks="5"/>
    </axis>
  </paramSet>
  <pointList>
    <point x="-5.00000000000" y="24476.2352941176"/>
    <point x="-4.00000000000" y="24381.7843137255"/>
    <point x="-3.00000000000" y="24200.7647058824"/>
    <point x="-2.00000000000" y="24098.3725490196"/>
    <point x="-1.00000000000" y="23951.7254901961"/>
  </pointList>
  <pointList>
    <point x="0.00000000000" y="23835.0196078431"/>
    <point x="1.00000000000" y="23697.9803921569"/>
    <point x="2.00000000000" y="23569.9215686275"/>
    <point x="3.00000000000" y="23436.9607843137"/>
    <point x="4.00000000000" y="23267.2941176471"/>
  </pointList>
</plot>
```

```
<plot>

<title>

<pointList>

<point>

<range>

<axis>

<tickList>

<tick>
```

Content/style separation I

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"
  href="http://www.eminerals.org/XSLT/pelote.xsl"?>
<plot xmlns="http://www.uszla.me.uk/xsl/1.0/pelote">
  <title>Volumes</title>
  <paramSet>
    <range floorX="23000"/>
    <axis type="y" position="2">
      <tickList numTicks="2"/>
    </axis>
  </paramSet>
  <pointList>
    <point x="-5.00000000000" y="24476.2352941176"/>
    <point x="-4.00000000000" y="24381.7843137255"/>
    <point x="-3.00000000000" y="24200.7647058824"/>
    <point x="-2.00000000000" y="24098.3725490196"/>
    <point x="-1.00000000000" y="23951.7254901961"/>
    <point x="0.00000000000" y="23835.0196078431"/>
    <point x="1.00000000000" y="23697.9803921569"/>
    <point x="2.00000000000" y="23569.9215686275"/>
    <point x="3.00000000000" y="23436.9607843137"/>
    <point x="4.00000000000" y="23267.2941176471"/>
  </pointList>
</plot>
```

Plotting instructions

Data

Content/style separation II

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml"
  href="http://www.eminerals.org/XSLT/pelote.xsl"?>
<plot xmlns="http://www.uszla.me.uk/xsl/1.0/pelote">
  <style><![CDATA[
      .thickRed {
        stroke: #00ff00;
        stroke-width: 10px;
      }
  ]]>
  </style>
  <pointList jh="hjk" class="thickRed">
      <point x='1' y='2'/>
      <point x='2' y='4'/>
  </pointList>
</plot>
```

Styling instructions

Data

Implementation

- XSLT 1.0 + EXSLT (common, math)
  - Portable across all standard XSLT processors
- Exposes one template interface
  - can be embedded in larger project.

Pelote

- Demonstrate:
  - All defaults
  - Alter range
  - Alter axis position
  - Additional axes
  - Change tick values
  - Change styling

ccViz

- ccViz: generic CML simulation output viewing tool
- Demonstrate embedding of Pelote into ccViz.
- Full XSLT solution.

Conclusions

- XSLT - platform-agnostic tool for data extraction
- ¨ ¨ ¨ for simple data manipulation to SVG
- Pelote