

Adding Rappture to MATLAB Applications

```

<?xml version="1.0"?>
<run>
  <tool>
    <title><cephing Calculator</title>
    <about>Press Simulate to view results.</about>
    <command>python $tool/graph.py $driver</command>
  </tool>
  <input>
    <string id="formula">
      <about>
        <label>Formula</label>
        <hint><math>2x^2 + 5x + 1</math></hint>
      </about>
      <size>30x5</size>
    </string>
    <number id="min">
      <about><label>From x</label></about>
      <default>0</default>
    </number>
    <number id="max">
      <about><label>To x</label></about>
      <default>1</default>
    </number>
  </input>
  <output>
    <curve id="result">
      <about><label>Formula: y vs x</label></about>
    </curve>
  </output>
</run>

```



Michael McLennan
Software Architect
HUBzero™ Platform for Scientific Collaboration

Example: Matlab/Octave Tool

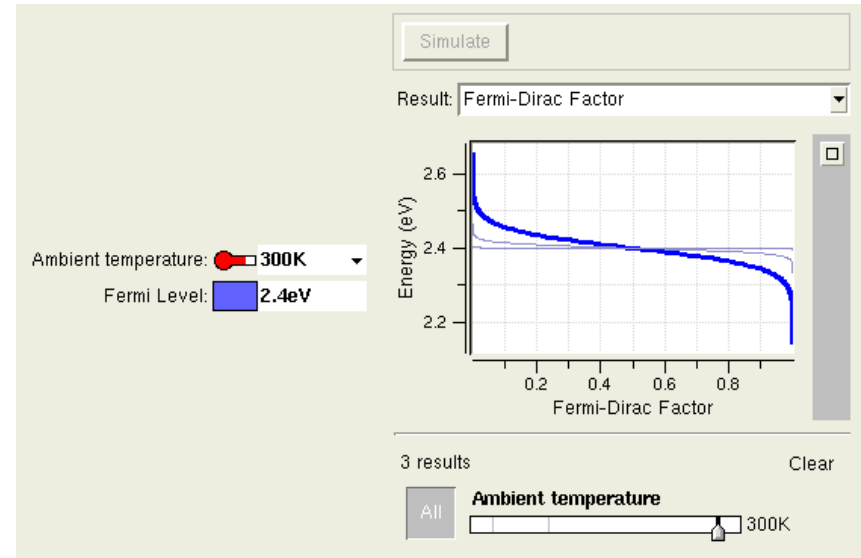
The usual way...

```
% matlab -nodisplay -r fermi
Enter the Fermi level (eV):
Ef = 2.4
Enter the temperature (K):
T = 77
```

```
% more out.dat
FERMI-DIRAC FUNCTION F1/2
```

<i>f1/2</i>	<i>Energy (eV)</i>
0.999955	2.33365
0.99995	2.33431
0.999944	2.33498

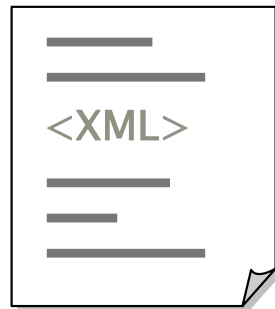
The Rappture way...



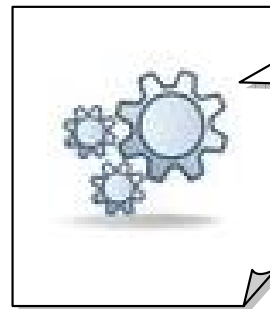
<https://developer.nanohub.org/projects/rappture>
source code: rappture/examples/app-fermi

How does it work?

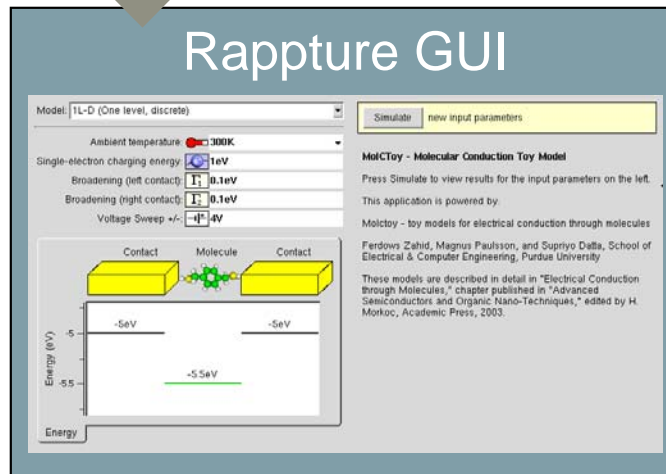
description of tool,
including inputs
and outputs



tool.xml

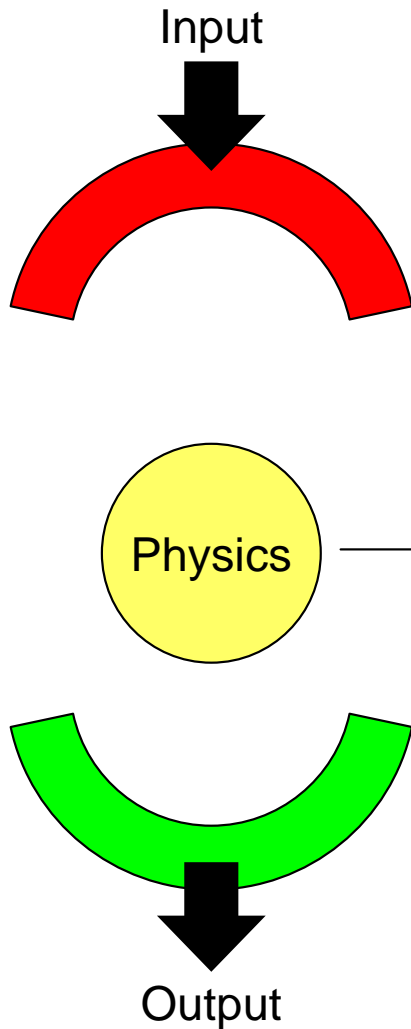


executable



Produces the
user interface
automatically!

What is the interface?



number

```
disp('Enter the Fermi Level (eV): ');
Ef = input(' Ef = ');
```

number

```
disp('Enter the temperature (K): ');
T = input(' T = ');
```

```
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;
```

```
E = linspace(Emin, Emax, 200);
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));
```

curve

```
fid = fopen('out.dat', 'w');
fprintf(fid, 'FERMI-DIRAC FUNCTION\n\n');
fprintf(fid, 'f1/2          Energy (eV)\n');
fprintf(fid, '-----\n');
fprintf(fid, '%12g %12g\n', [f; E]);
fclose(fid);
```

```
quit;
```

Describing inputs

```
disp('Enter the Fermi Level (eV): ');
Ef = input(' Ef = ');
```

```
disp('Enter the temperature (K): ');
```

```
<number id="Ef" >
  <about>
    <label>Fermi Level </label >
    <description>Energy at center of distribution. </description>
  </about>
  <units>eV</units>
  <min>-10eV</min>
  <max>10eV</max>
  <default>0eV</default >
</number>
```

Rappture description
of a number input

```
fprintf(fid, '\n2 Energy (eV)\n');
fprintf(fid, '-----\n');
fprintf(fid, '%12g %12g\n', [f; E]);
fclose(fid);

quit;
```

Describing inputs

```
disp(' Enter the Fermi level (eV): ');
Ef = input(' Ef = ');
```

```
disp(' Enter the temperature (K): ');
T = input(' T = ');
```

```
<number id="temperature">
  <about>
    <label>Ambient temperature</label>
    <description>Temperature of the environment.</description>
  </about>
  <units>K</units>
  <min>0K</min>
  <max>500K</max>
  <default>300K</default>
  <preset>
    <value>77K</value>
    <label>77K (liquid nitrogen)</label>
  </preset>
  ...
</number>
```

Describing outputs

```
disp(' Enter the Fermi Level (eV): ');
Ef = input(' Ef = ');
```

```
<curve id="f12">
  <about><labe l>Fermi -Dirac Factor</labe l></about>
  <xaxis>
    <labe l>Fermi -Dirac Factor</labe l>
  </xaxis>
  <yaxis>
    <labe l>Energy</labe l>
    <units>eV</units>
  </yaxis>
</curve>
```

```
(K): ');
```

```
;
(Ef)/kT));
```

```
fprintf(fid, 'FERMI-DIRAC FUNCTION\n\n');
fprintf(fid, ' f1/2 Energy (eV)\n');
fprintf(fid, ' -----\n');
fprintf(fid, '%12g %12g\n', [f; E]);
fclose(fid);
```

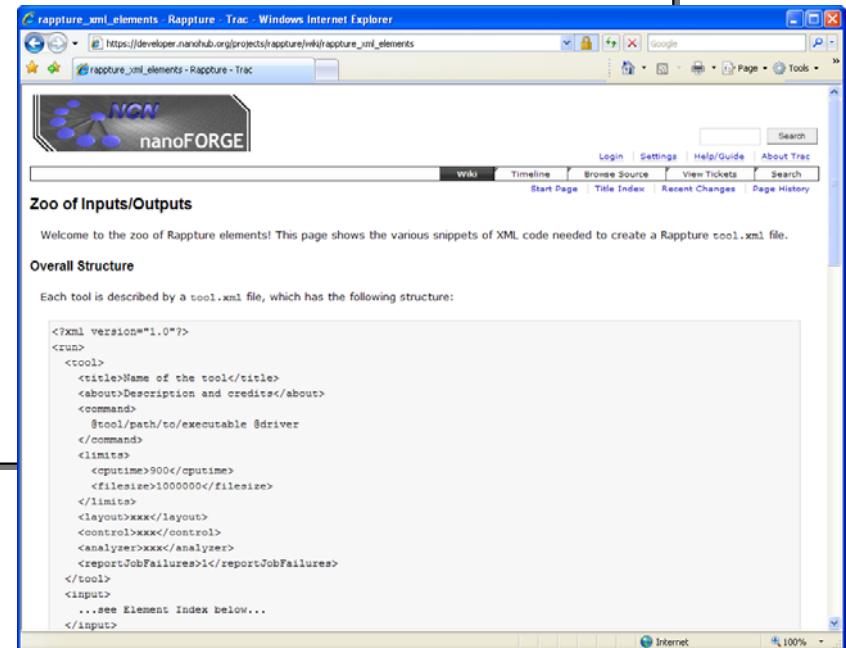
```
quit;
```

Describing the Whole Tool

File: tool . xml

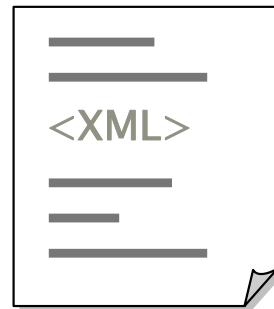
```
<?xml version="1.0"?>
<run>
  <tool >
    <about>Press Simulate to view results.</about>
    <command>matlab -nodisplay -r infile='@driver', path('@tool', path),
fermi </command>
  </tool >
  <input>
    <number id="temperature">...</number>
    <number id="Ef">...</number>
  </input>
  <output>
    <curve id="f12">...</curve>
  </output>
</run>
```

Basic structure from [this page](#):

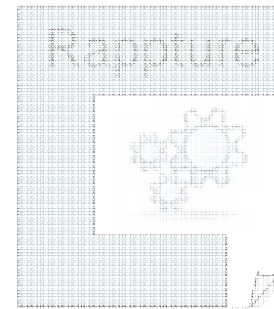


Half-way there

Once you've created the tool.xml file, you can test the GUI



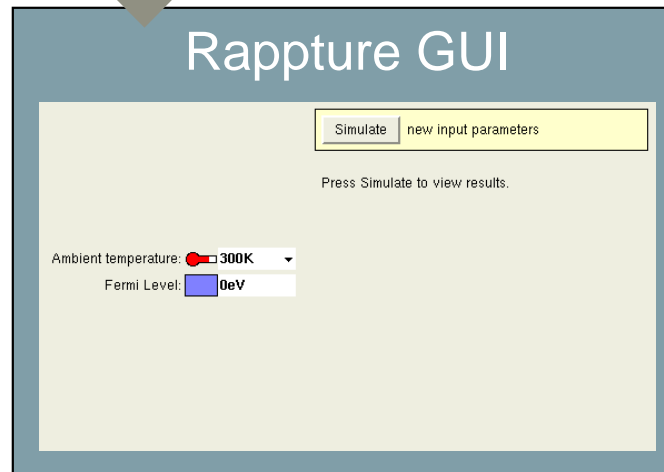
tool.xml



executable

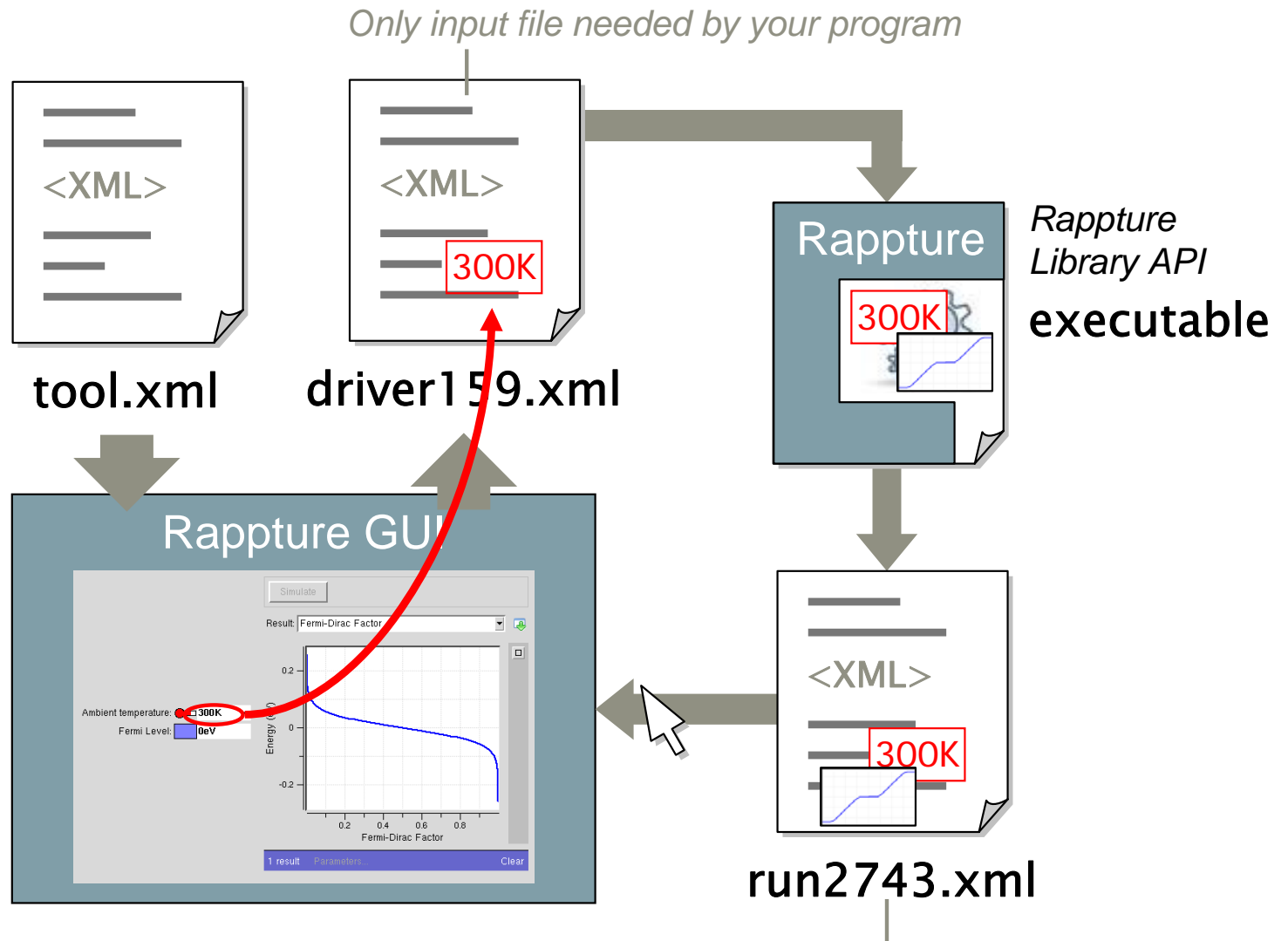
Fix this next...

```
% cd tooldir
% ls
tool.xml
% rappture
```



How does it work?

description of tool,
including inputs
and outputs

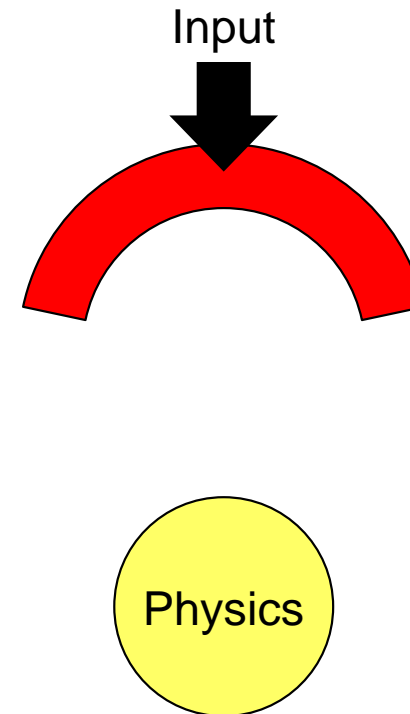


All inputs and outputs—complete record of the run

Updating our original script

fermi.m Matlab script

```
disp('Enter the Fermi level (eV):');  
Ef = input(' Ef = ');  
  
disp('Enter the temperature (K):');  
T = input(' T = ');  
  
% do fermi calculations (science)...  
kT = 8.61734e-5 * T;  
Emin = Ef - 10*kT;  
Emax = Ef + 10*kT;  
  
E = linspace(Emin, Emax, 200);  
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));  
...
```



Read the fermi level

fermi.m Matlab script

```
% open the driver file
lib = rpLib(infile);
```

```
% get the input values and convert to correct units
```

```
Ef = rpLibGetString(lib, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(Ef, 'eV');
```

"240meV"
2.4

```
T = rpLibGetString(lib, 'input.number(T).current');
[T, err] = rpUnitsConvertDbl(T, 'K');
```

```
% do fermi calculations (sci
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;
```

```
E = linspace(Emin, Emax, 200);
f = 1.0 ./ (1.0 + exp((E - Ef) / kT));
...
```

tool.xml:

```
<?xml version="1.0"?>
<run>
...
  <input>
    <number id="Ef">
      <current>2.4eV</current>
    </number>
  </input>
...
</run>
```

input
.number(Ef)
.current

Read the temperature

fermi.m Matlab script

```
% open the driver file
lib = rpLib(infile);

% get the input values and convert to correct units
Ef = rpLibGetString(lib, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(Ef, 'eV');

T = rpLibGetString(lib, 'input.number(temperature).current');
[T, err] = rpUnitsConvertDbl(T, 'K');

% do fermi calculations (science)...
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;

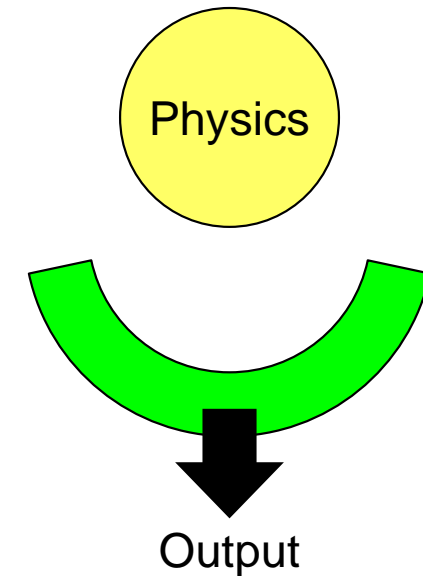
E = linspace(Emin, Emax, 200);
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));
...
```

" 100C"
373.15

Updating our original script

fermi.m Matlab script

```
...  
E = linspace(Emin, Emax, 200);  
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));  
  
fid = fopen('out.dat', 'w');  
fprintf(fid, 'FERMI-DIRAC FUNCTION F1/2\n\n');  
fprintf(fid, 'f1/2          Energy (eV)\n');  
fprintf(fid, '-----\n');  
fprintf(fid, '%12g %12g\n', [f; E]);  
fclose(fid);  
  
quit;
```

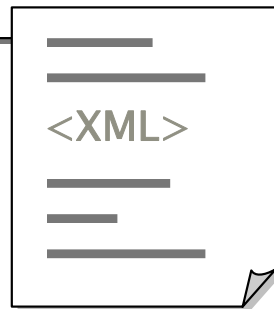


Save the results

fermi.m Matlab script

```
...  
E = linspace(Emin, Emax, 200);  
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));  
  
% store output in curve  
putStr = sprintf('%12g %12g\n', [f; E]);  
rpLibPutString(lib, 'output.curve(f12).component.xy', putStr, 0);  
  
% save the output  
rpLibResult(lib);  
  
quit;
```

Means
“overwrite”
(no “append”)



run1128018316.xml

Save the results

fermi.m Matlab script

```
...  
E = linspace(Emin, Emax, 200);  
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));  
  
% store output in curve  
putStr = sprintf('%12g %12g\n', [f; E]);  
rpLibPutString(lib, 'output.curve(f12).comp  
  
% save the output  
rpLibResult(lib);  
  
quit;
```

```
<?xml version="1.0"?>  
<run>  
...  
<output>  
  <curve id="f12">  
    ...  
    <component>  
      <xy>  
        0.999955  2.33365  
        0.99995  2.33431  
        0.999944  2.33498  
      ...  
    </output>  
</run>
```

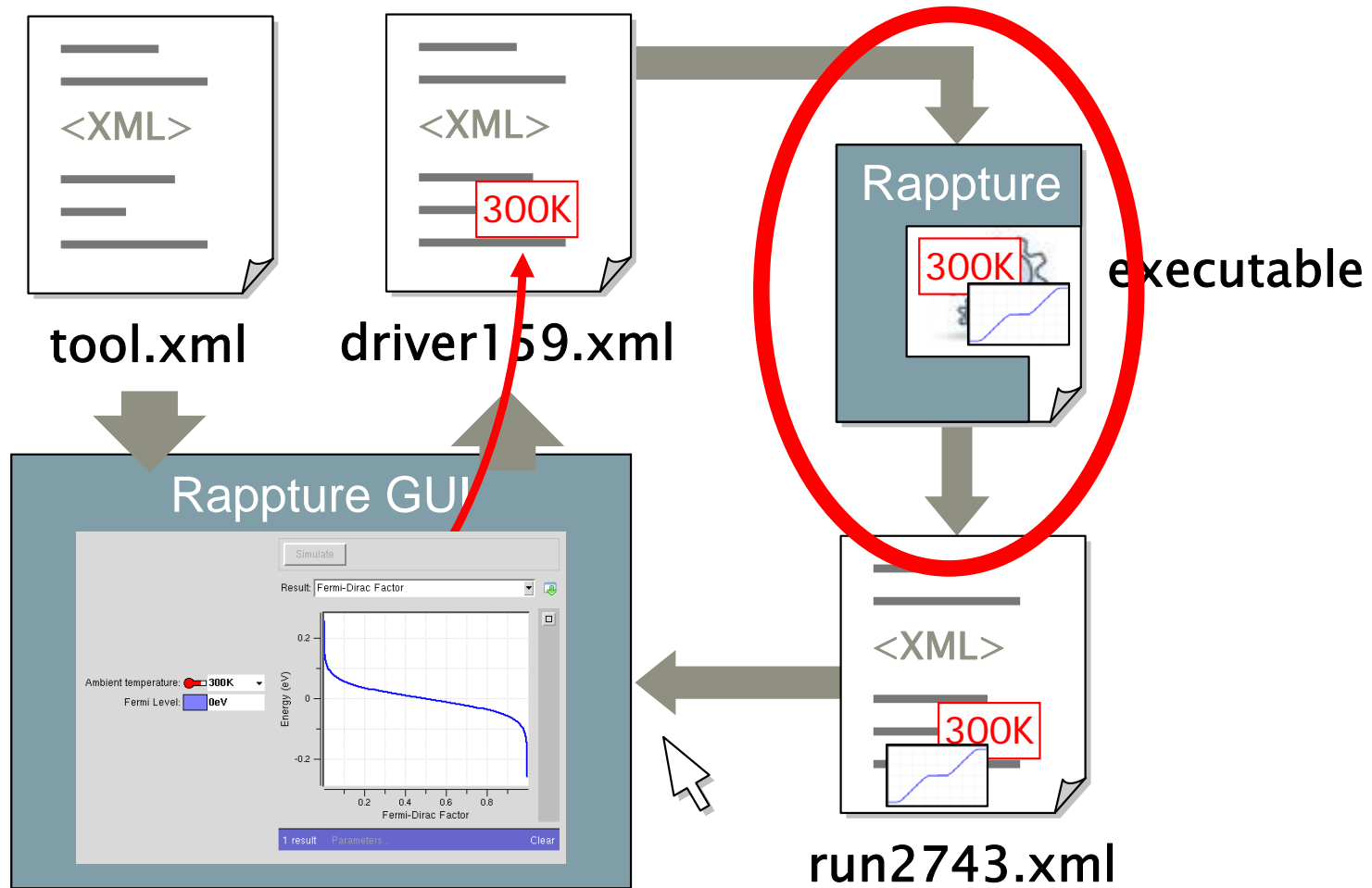


<XML>

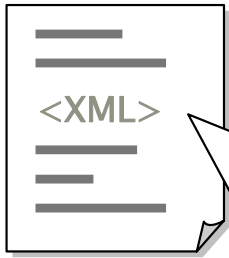
run1128018316.xml

Run the program

How does the program get invoked?



How does the program get invoked?



tool.xml

Replaced with directory containing tool.xml file

Ex: /apps/yourtool/current

Replaced with name of driver file

Ex: driver159.xml

```
<?xml version="1.0"?>
<run>
  <tool >
    <about>Press Simulate to view results.</about>
    <command>matlab -nodisplay -r infile='@driver', path('@tool', path), fermi
  </command>
  </tool >
  <input>
    ...
  </input>
  <output>
    ...
  </output>
</run>
```

Set infile to the name of driver file

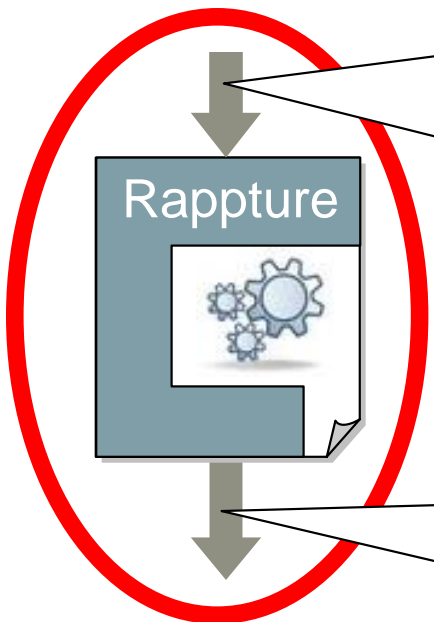
Add tool directory to search path

Invoke fermi.m

How does the program get invoked?

```
matlab -nodisplay -r infile='@driver', path('@tool', path), fermi
```

```
matlab -nodisplay -r infile='driver159.xml', path('/apps/yourtool/current', path), fermi
```

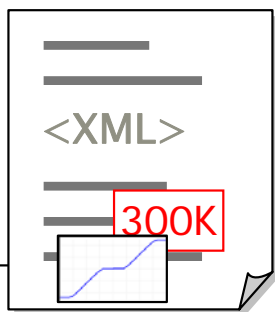


```
% open the file
lib = rpLib(infile);

% get the input values and convert to correct units
Ef = rpLibGetString(lib, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(Ef, 'eV');
...
```

```
...
% save the output
rpLibResult(lib);

quit;
```

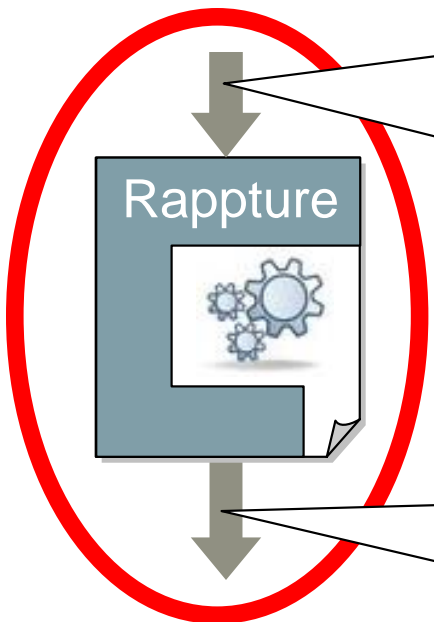


run2743.xml

Using Octave 3

```
octave --silent --eval infile='@driver', path('@tool', path), fermi
```

```
octave --silent --eval infile='driver159.xml', path('/apps/yourtool/current', path), fermi
```

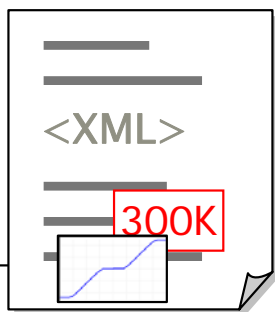


```
% open the input file
lib = rpLib(infile);

% get the input values and convert to correct units
Ef = rpLibGetString(lib, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(Ef, 'eV');
...
```

```
...
% save the output
rpLibResult(lib);

quit;
```



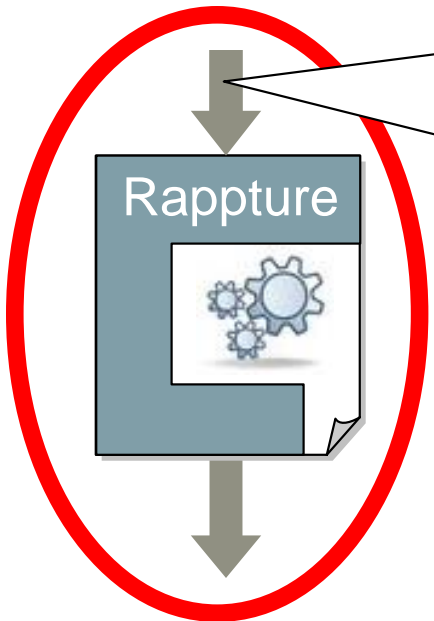
run2743.xml

Using Octave 2

octave --silent --error-infile=@driver', path('@tool', path), fermi

octave --silent @tool /fermi.m @driver

octave --silent /apps/yourtool /current/fermi.m driver159.xml



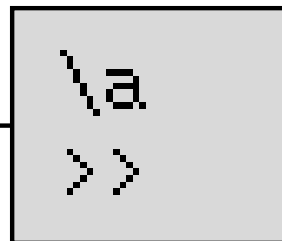
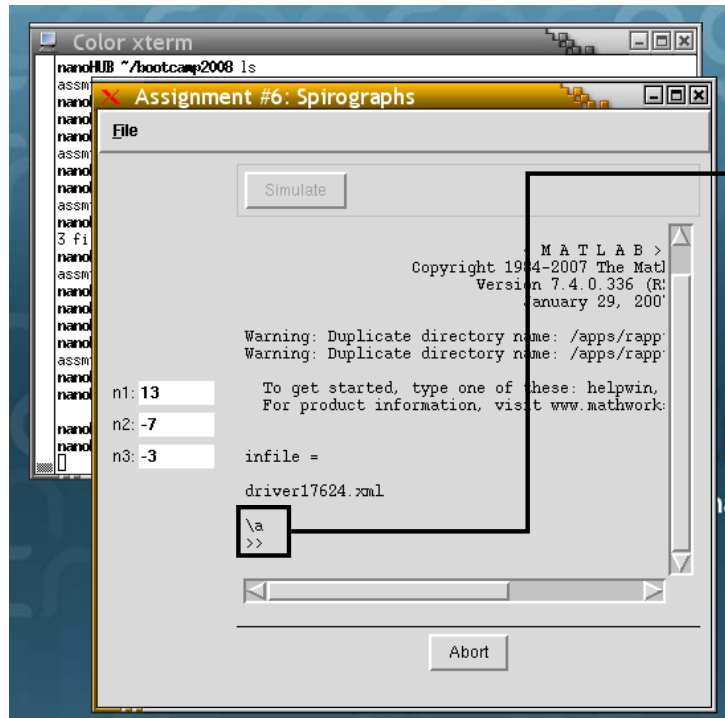
```
% open the driver file
lib = rpLib(argv{1});

% get the input values and convert to correct units
Ef = rpLibGetString(lib, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(Ef, 'eV');
...
```

Watch out! Use curly braces here

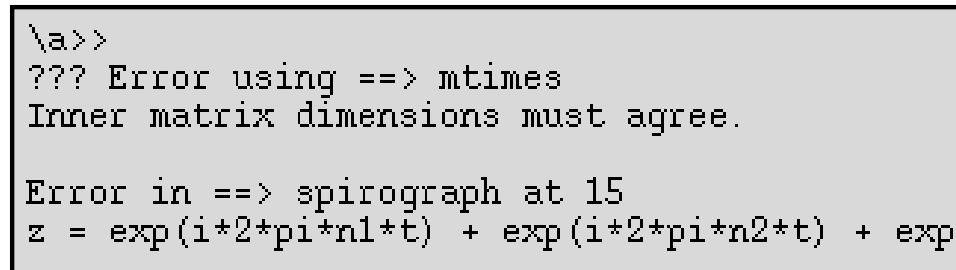
Debugging

If something goes wrong, MATLAB goes into “debug” mode:



Waiting for you to type a MATLAB command

Click *Abort* instead



Run it by hand:

```
$ ls
```

```
driver1529.xml  fermi.m  tool.xml
```

```
$ use rapture
```

```
$ matlab -r infile='driver1529.xml', fermi
```

