

# 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>2 * x + 5 / \sin(x)</math></hint>
      </about>
      <size>30x45</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

*HUBzero® Platform for Scientific Collaboration*

*Purdue University*

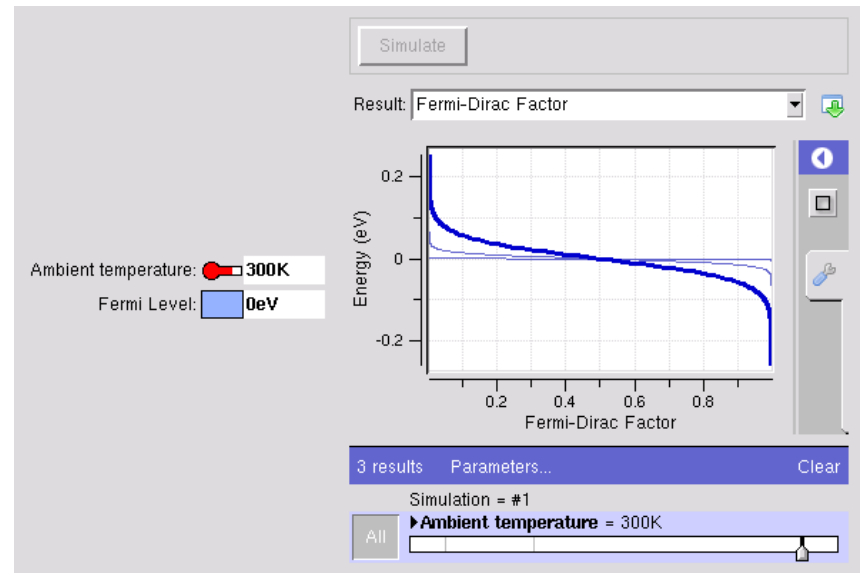
The usual way...

```
% 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
```

$f_{1/2}$	Energy (eV)
0.999955	2.33365
0.99995	2.33431
0.999944	2.33498

The Rappture way...



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

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

number

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

number

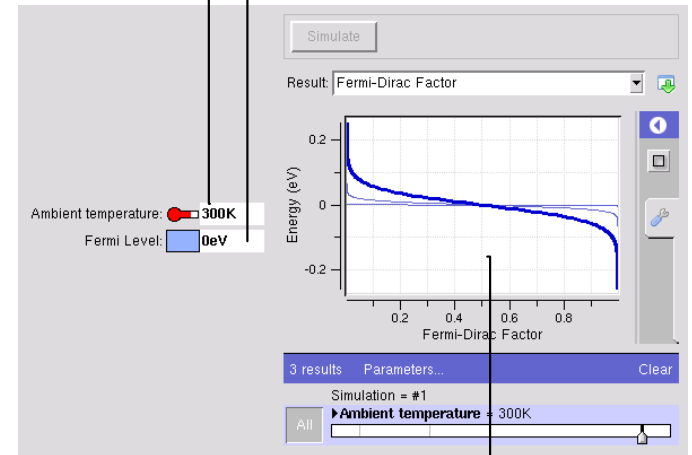
```
kT = 8.61734e-5 * T;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;

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

*physics*

```
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);
```

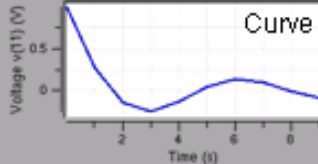
```
quit;
```



curve

**Rappture Builder** [Build] [Preview] [New...] [Open...] [Save As...]

**Object Types:**

- All
- Boolean:  yes
- Label: Choice2
  - Choice1
  - Choice2
  - Choice3
- Curve
 
- Group
- Documentation
- Grouping
- Inputs
- Outputs

**Tool Interface:**

- Tool:
- ⊕ Input:
  - Number: temperature** ← Define this input
  - Number: Ef
- ⊕ Output:
  - Curve: f12

Object: input.number(temperature)    Rename    Help    Delete

**Label:** Ambient temperature

**Description:** Temperature of the environment.

**Enable:**

**Default Value:** 300K

**Units of Measurement:** K

**Minimum Value:** 0K

**Maximum Value:** 500K

**Icon:**  Load... Save... Clear

**Rapture Builder** [Build] [Preview]

New... Open... Save As...

**Object Types:**

All

Boolean:  yes

Label: Choice2

- Choice1
- Choice2
- Choice3

Curve

Group

Documentation

Grouping

Inputs

Outputs

**Tool Interface:**

Tool:

- ⊕ Input:
  - Number: temperature
  - Number: Ef** ← Define this input
- ⊕ Output:
  - Curve: f12

Object: input.number(Ef) Rename Help Delete

**Label:** Fermi Level

**Description:** Energy at center of distribution.

**Enable:**

**Default Value:** 0eV

**Units of Measurement:** eV

**Minimum Value:** -10eV

**Maximum Value:** 10eV

**Icon:**  Load... Save... Clear

**Rappture Builder** [Build] [Preview]

New... Open... Save As...

**Object Types:**

All

Boolean:  yes

Label: Choice2

- Choice1
- Choice2
- Choice3

Curve

Group

Documentation

Grouping

Inputs

Outputs

**Tool Interface:**

Tool:

⊕ Input:

- Number: temperature
- Number: Ef

⊕ Output:

- Curve: f12

Object: output.curve(f12) Rename Help Delete

**Label:** Fermi-Dirac Factor

**Description:** Fermi function of order 1/2, representing the equilibrium distribution of fermion particles in energy space.

**Plotting Group:**

**X-axis Label:** Fermi-Dirac Factor

**X-axis Description:** Function f12(E)

**X-axis Units:**

**Y-axis Label:** Energy

**Y-axis Description:** Energy of particles in the equilibrium distribution.

**Y-axis Units:** eV

**Define this output**

**Rappture Builder** [Build] [Preview]

New... Open... Save As...

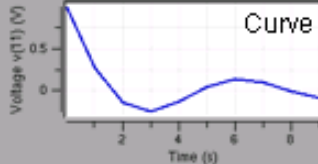
**Object Types:**

All

Boolean:  yes

Label: Choice2

- Choice1
- Choice2
- Choice3



Group

Documentation

Grouping

Inputs

Outputs

**Tool Interface:**

Tool:

- ⊕ Input:
  - Number: temperature
  - Number: Ef
- ⊕ Output:
  - Curve: f12

Object: tool Help

**Title:** Fermi-Dirac Calculator

**Description:** Press Simulate to view results.

**Program:** Octave



## Generated script: main.m

```

...
% get input value for input.number(temperature) and convert to K
str = rpLibGetString(i o, 'input.number(temperature).current');
[temperature, err] = rpUnitsConvertDbl(str, 'K');

% get input value for input.number(Ef) and convert to eV
str = rpLibGetString(i o, 'input.number(Ef).current');
[Ef, err] = rpUnitsConvertDbl(str, 'eV');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Add your code here for physics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kT = 8.61734e-5 * T;
Emi n = Ef - 10*kT;
Emax = Ef + 10*kT;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
E = linspace(Emi n, Emax, 200);
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% save output value for output.curve(f12)
% this assumes a vector 'x' and a vector 'y'
xydata = [x; y];
str = sprintf('%12g %12g\n', xydata);
rpLibPutString(i o, 'output.curve(f12).component.xy', str, 0);
...

```

*physics*



## Final script: main.m

```

...
% get input value for input.number(temperature) and convert to K
str = rpLibGetString(i o, 'input.number(temperature).current' );
[temperature, err] = rpUnitsConvertDbl (str, 'K' );

% get input value for input.number(Ef) and convert to eV
str = rpLibGetString(i o, 'input.number(Ef).current' );
[Ef, err] = rpUnitsConvertDbl (str, 'eV' );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kT = 8.61734e-5 * temperature;
Emi n = Ef - 10*kT;
Emax = Ef + 10*kT;

E = linspace(Emi n, Emax, 200);
f = 1.0 ./ (1.0 + exp((E - Ef)/kT));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

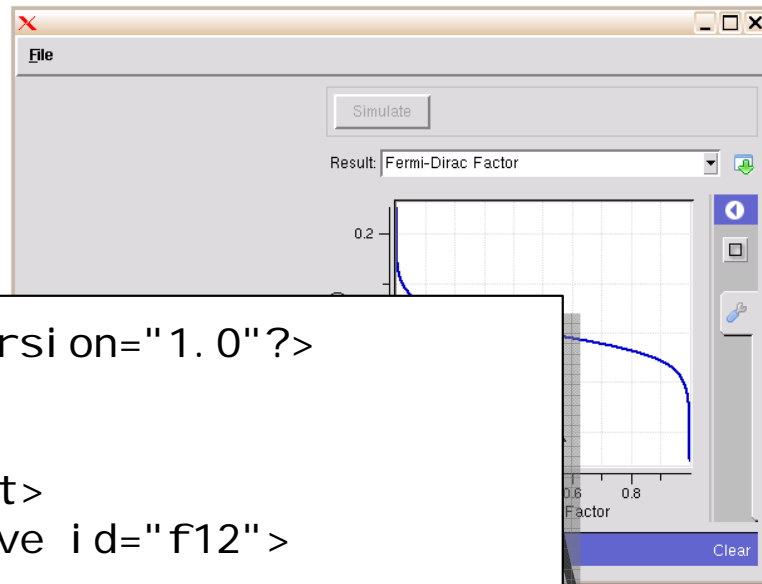
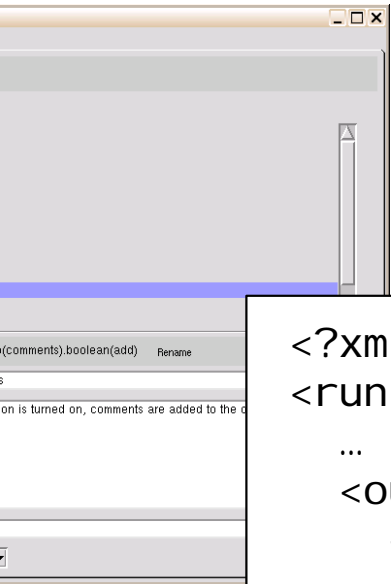
% save output value for output.curve(f12)
% this assumes a vector 'x' and a vector 'y'
xydata = [f; E];
str = sprintf('%12g %12g\n', xydata);
rpLibPutString(i o, 'output.curve(f12).component.xy', str, 0);
...

```

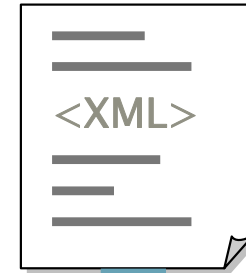
```
Color xterm
$ rappture
```

Builder

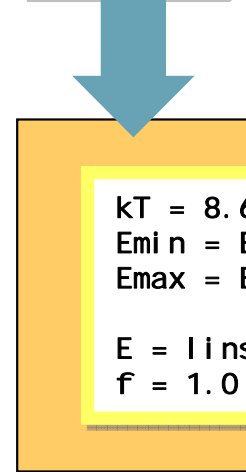
Runner



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

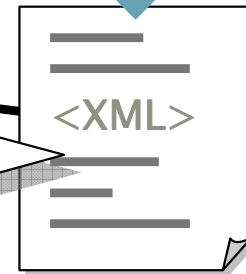


driver1827.xml



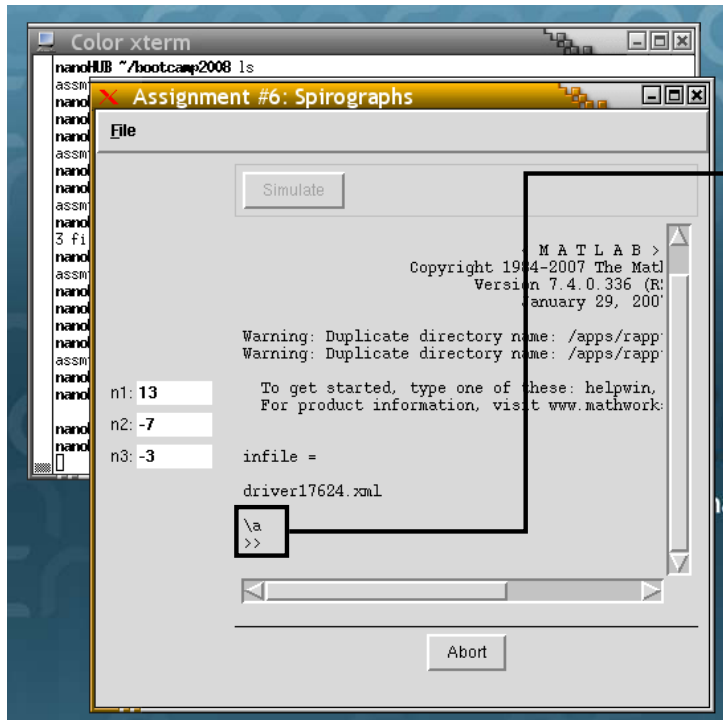
```
kT = 8.61734e-5 * temperature;
Emin = Ef - 10*kT;
Emax = Ef + 10*kT;

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



run12703129102.xml

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



```
\a
>>
```

Waiting for you to type a MATLAB command

Click *Abort* instead

```
\a>>
??? Error using ==> mtimes
Inner matrix dimensions must agree.

Error in ==> spirograph at 15
z = exp(i*2*pi*n1*t) + exp(i*2*pi*n2*t) + exp
```

Run it by hand:

```
$ ls
```

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

```
$ use rappture
```

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

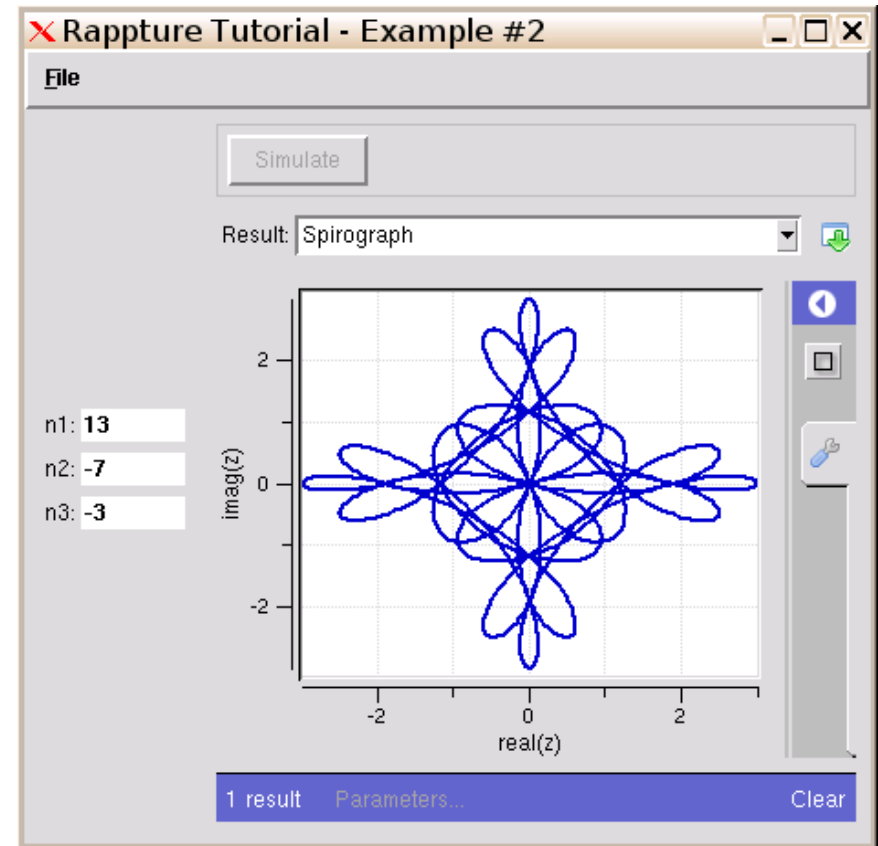
n1  
number

n2  
number

n3  
number

Spirograph equation

z  
curve



*In MATLAB/Octave:*

```
t = linspace(0, 1, 1000);  
z = exp(i * 2 * pi * n1 * t) + exp(i * 2 * pi * n2 * t) + exp(i * 2 * pi * n3 * t);  
plot(real(z), imag(z));
```