How to Calculate Magnetic Fields?

Peter Dunne

IPCMS - France

Today's Goal

- The aim of this tutorial is to show you how to calculate magnetic fields for some simple arrangements of magnets.
- It is *not* a tutorial on Python or scientific computing.

But First...

- How do we know which way to use the Gaussmeter?

- Which way is north?

watch_compass

- How do we know which way to use the Gaussmeter?

Remember: The north pole is a magnetic **south** pole

- Align the probe vertically with the black spot facing you and pointing in a North-South direction
- Set the gaussmeter to the higher sensitivity
- Vary the offset so that it is nearly zero
- When you rotate the probe to have the black spot facing you or facing away from you the Gaussmeter will flip from 'S' to 'N'

Assumptions

- Uniformly magnetised
- Only hard, permanent magnets, no soft magnetic material like pure iron
- Magnets are fully transparent to the magnetic fields
- We will consider only simple primitives: squares, cubes, cuboids, cylinders
- Only care about the magnetic field *outside* the magnets

Approach using Python

Python is a high level, object-oriented interpreted language

Pros

- Simple to use
- Real-time check of values using the interpreter
- Large collection of useful libraries

Cons

- Slow, much slower than compiled languages like C
- Diagnosing and fixing bugs can be difficult
- Can be difficult to maintain large code bases

This code

Feel free to look at the code afterwards. The design approach is as follows:

- We create a magnet object (square, cube, cylinder, etc....)
- Pass this object to a helper function that calculates the magnetic field
- Plot the resulting data using matplotlib

This version is not optimised for speed or efficiency, but using numpy it is already fast enough.

First Steps

Here we will import the modules we need

In [1]:
```
import import_modules # This module is used to tell Jupyter where to look for our magn
import lib.fields as mag # Magnetic library for these excercises

import numpy as np # we use numpy for handing vectors and matrices
from scipy.optimize import curve_fit # We will use the curve_fit function from scipy .

import matplotlib.pyplot as plt # Matplotlib is used to generate our plots

# Plotting backend, inline creates static figures, notebook creates dynamic ones
%matplotlib inline
# %matplotlib notebook

# This increases the resolution of plots displayed using the inline backend
%config InlineBackend.figure_format ='retina'
```

Magnetic Field Above A Cylinder

cylinder

The magnetic field directly above the centre is:

$$B_z = \frac{\mu_0 M_r}{2}\left[\frac{z+L}{\sqrt{(z+L)^2+R^2}} - \frac{z}{\sqrt{z^2+R^2}}\right] \tag{1}$$

Next we will call the reset function, this ensures there are no other magnet instances created in memory

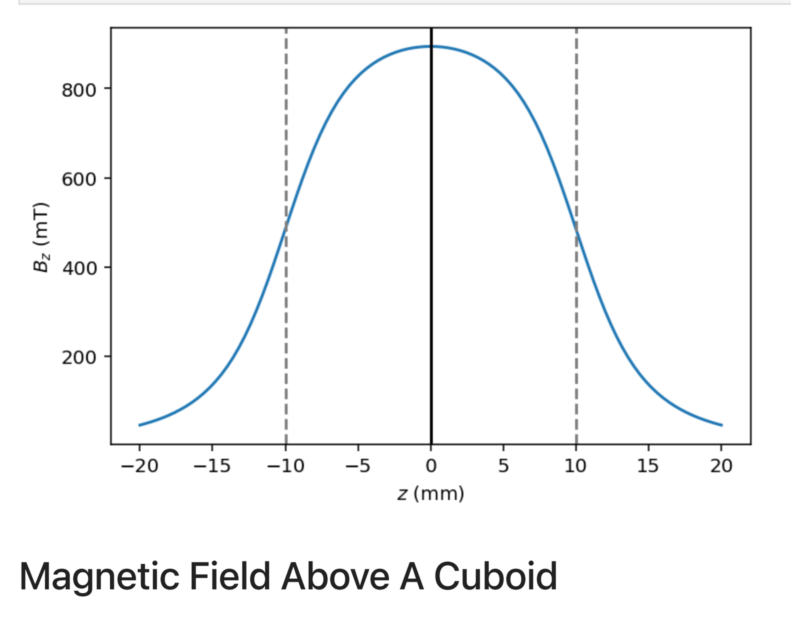
In [2]:
```
mag.reset_magnets()
```

We define the magnet parameters, writing $J_r = \mu_0 M_r$

In [3]:
```
R = 5e-3
L = 20e-3
m_cyl = mag.Magnet_Cylinder(R = R, L = L, Jr = 1.0, center=(0.0, 0.0, 0))
m_cyl
```

Out[3]:
```
Cylinder
No. 1
J: 1.0 (T)
Size: [0.005 0.02 ] (m)
Center [0. 0. 0.] (m)
```

In [4]:
```
mag.plot.plot_1D_field(m_cyl);
```



Magnetic Field Above A Cuboid

cuboid

For a cuboid, the equation is a little bit more complicated:

$$B_z = \frac{\mu_0 M_r}{2}\left[\tan^{-1}\left(\frac{(z+L)\sqrt{a^2+b^2+(z+L)^2}}{ab}\right) - \tan^{-1}\left(\frac{z\sqrt{a^2+b^2+z^2}}{ab}\right)\right] \tag{2}$$

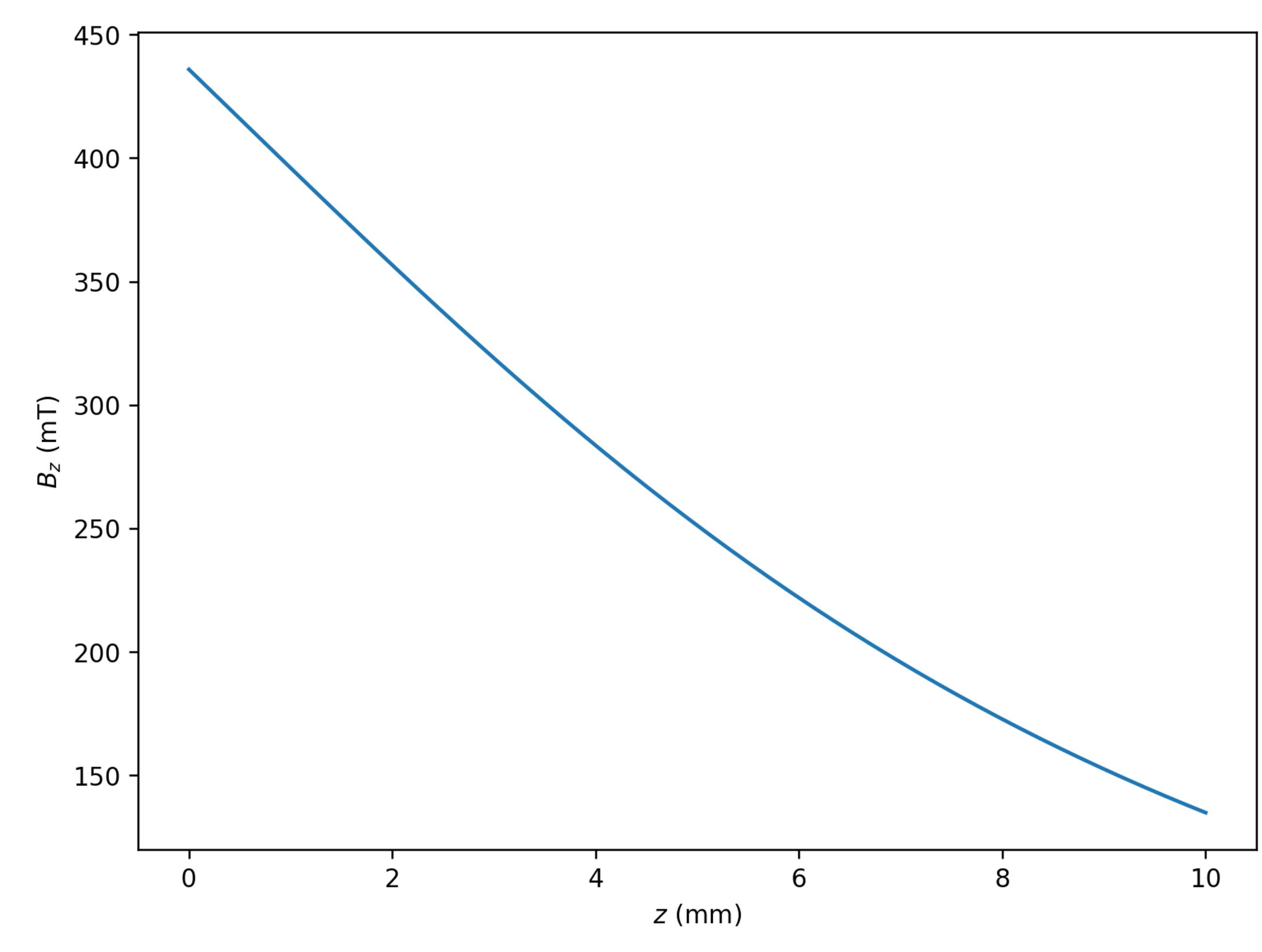
In [5]:
```
a = 10e-3
m_cube = mag.Magnet_Cube(a = a, Jr = 1, center=(0.0, 0, -a))
m_cube
```

Out[5]:
```
Cube
No. 1
J: 1 (T)
Size: [0.02 0.02 0.02] (m)
Center [ 0.    0.   -0.01] (m)
```

In [6]:
```
z = np.linspace(0, a)

Bz = mag.magnetic_field_prism_1D(m_cube, z)

fig, ax = plt.subplots(figsize=(8,6), dpi=120)
plt.xlabel(r'$z$ (mm)')
plt.ylabel(r'$B_z$ (mT)')
plt.plot(z*1e3, Bz*1e3, label='Cube')
plt.show()
```



In [7]:
```
mag.list_magnets();
```

```
No Rectangle magnets instantiated
No Square magnets instantiated
No Prism magnets instantiated
Cube
No. 1
J: [0. 0. 1.] (T)
Size: [0.02 0.02 0.02] (m)
Center [ 0.    0.   -0.01] (m)

Cylinder
No. 1
J: 1.0 (T)
Size: [0.005 0.02 ] (m)
Center [0. 0. 0.] (m)
```

Testing Configurations

Cuboid of size 10 x 20 x 40 mm^3

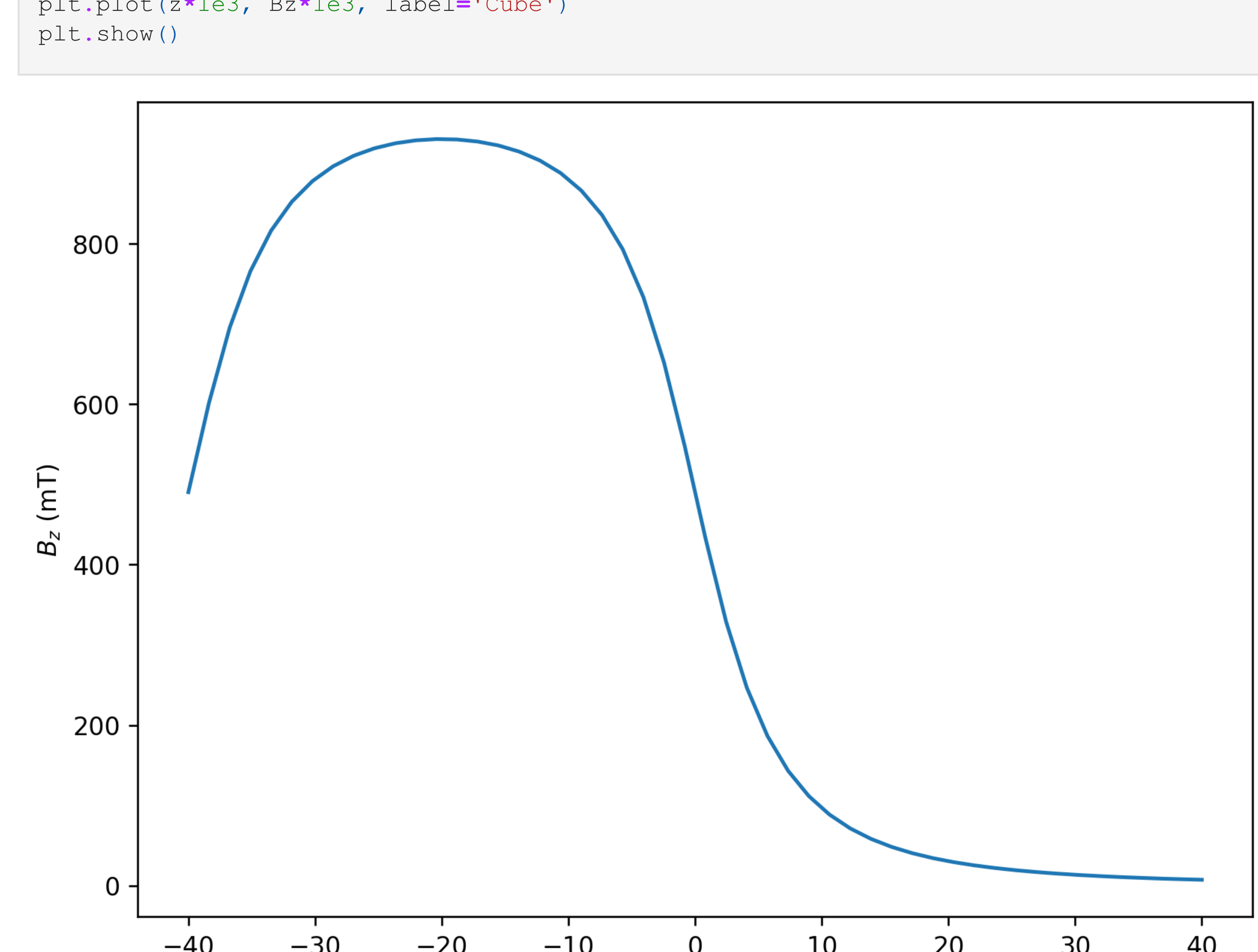
In [8]:
```
mag.reset_magnets();

a = 10e-3 / 2
b = 20e-3 / 2
c = 40e-3 / 2
center = (0, 0, -20e-3)

m_prism = mag.Magnet_Prism(a=a, b=b, c=c, Jr = 1.0, center = center)

z = np.linspace(-2*c, 2*c)
Bz = mag.magnetic_field_prism_1D(m_prism,z)

fig, ax = plt.subplots(figsize=(8,6), dpi=120)
plt.xlabel(r'$z$ (mm)')
plt.ylabel(r'$B_z$ (mT)')
plt.plot(z*1e3, Bz*1e3, label='Cube')
plt.show()
```

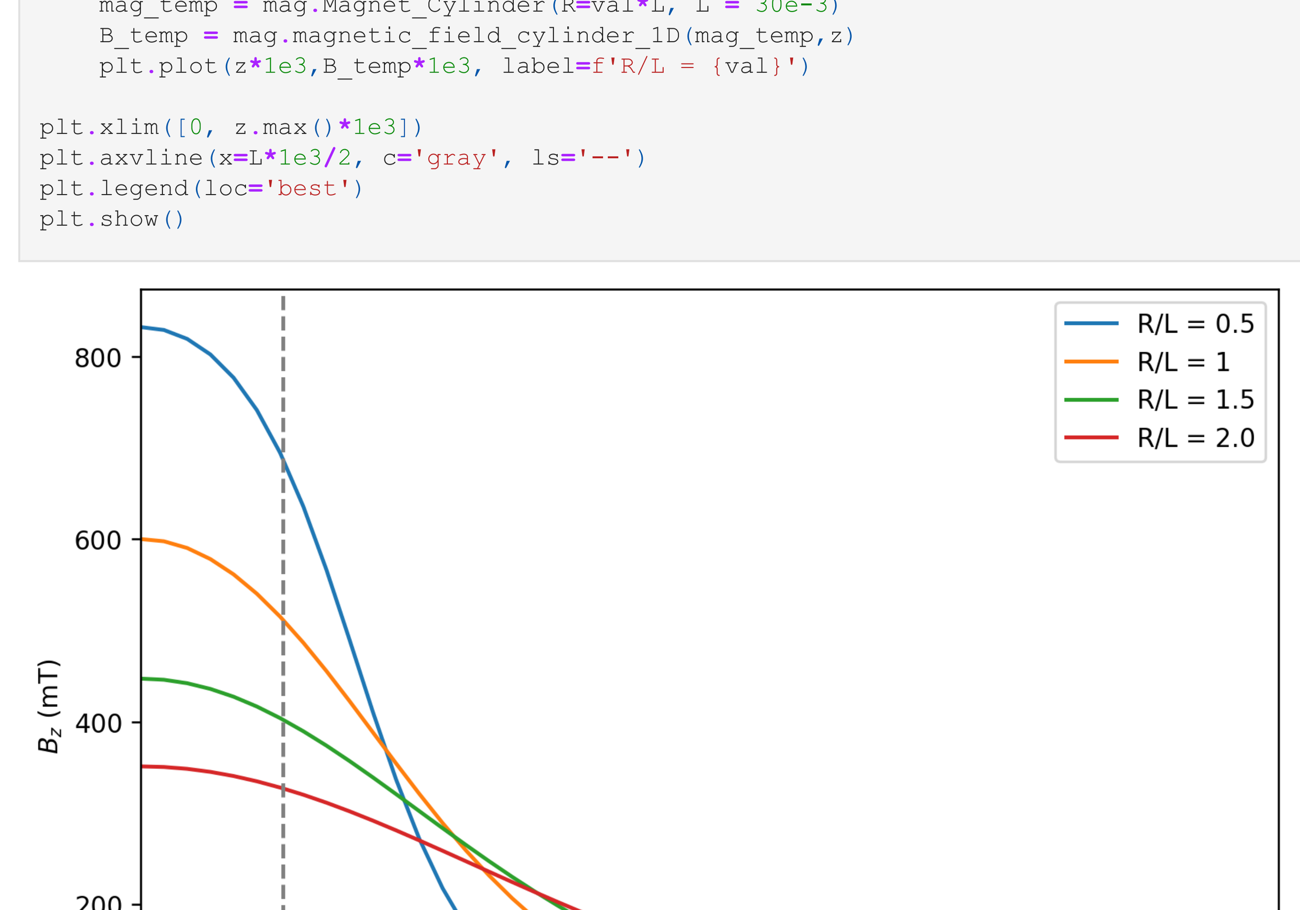


In [9]:
```
L = 20e-3
z = np.linspace(0, 4*L)

fig, ax = plt.subplots(figsize=(8,6), dpi=120)
plt.xlabel(r'$z$ (mm)')
plt.ylabel(r'$B_z$ (mT)')

for val in [0.5, 1, 1.5, 2.0]:
    mag_temp = mag.Magnet_Cylinder(R=val*L, L = 30e-3)
    B_temp = mag.magnetic_field_cylinder_1D(mag_temp,z)
    plt.plot(z*1e3,B_temp*1e3, label=f'R/L = {val}')

plt.xlim([0, z.max()*1e3])
plt.axvline(x=L*1e3/2, c='gray', ls='--')
plt.legend(loc='best')
plt.show()
```



Method of Images

method_of_images

- A sheet of soft ferromagnetic material acts as a mirror
- Doubles the effective length of a magnet

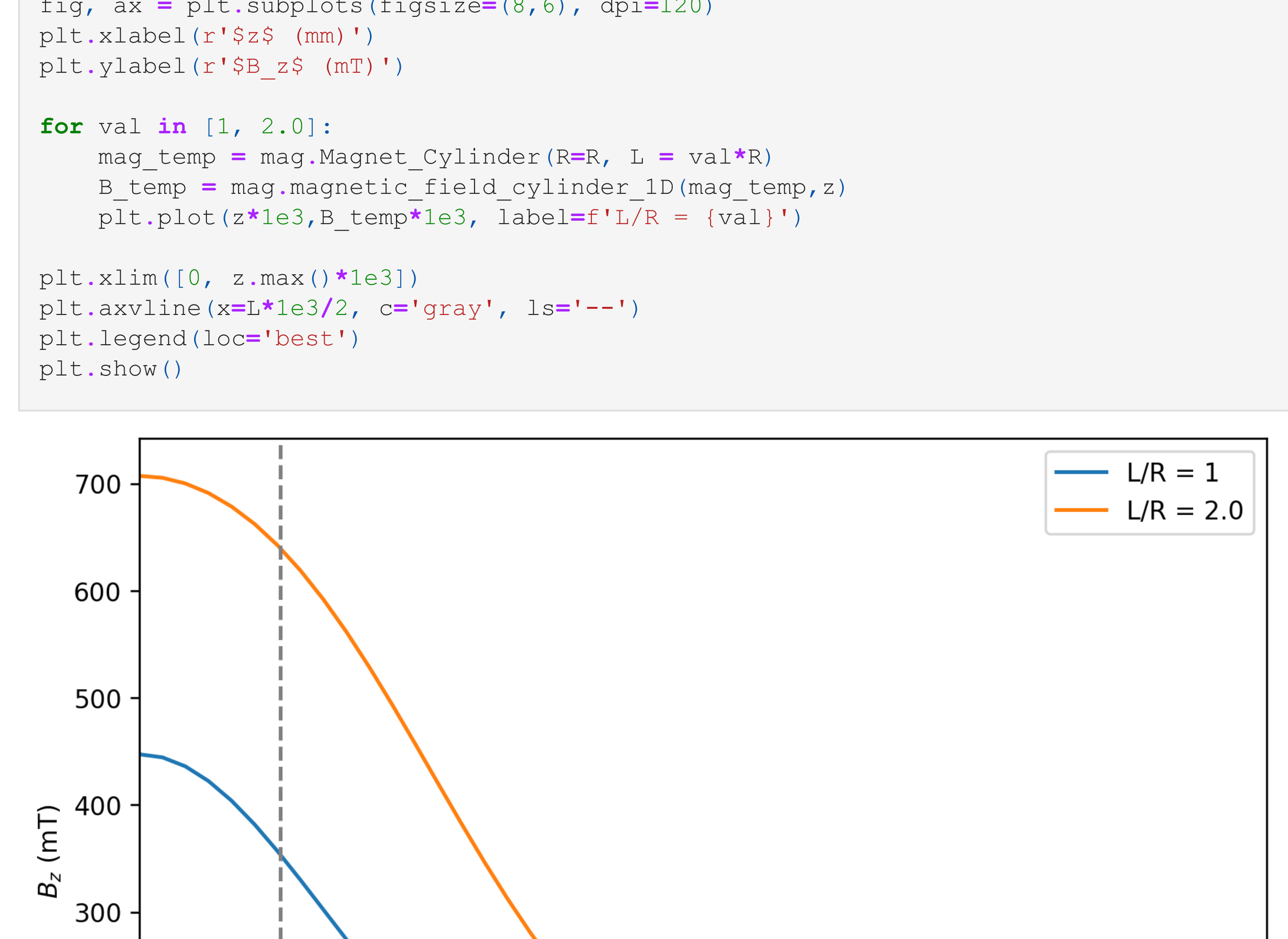
Plotting the effect of Soft Iron

In [10]:
```
R = 20e-3
z = np.linspace(0, 4*L)

fig, ax = plt.subplots(figsize=(8,6), dpi=120)
plt.xlabel(r'$z$ (mm)')
plt.ylabel(r'$B_z$ (mT)')

for val in [1, 2.0]:
    mag_temp = mag.Magnet_Cylinder(R=R, L = val*R)
    B_temp = mag.magnetic_field_cylinder_1D(mag_temp,z)
    plt.plot(z*1e3,B_temp*1e3, label=f'L/R = {val}')

plt.xlim([0, z.max()*1e3])
plt.axvline(x=L*1e3/2, c='gray', ls='--')
plt.legend(loc='best')
plt.show()
```



Measuring the remnant magnetisation of a magnet

By measuring the central field as a function of distance, we can fit this to the theoretical code to get J_r

$$B_z = \frac{\mu_0 M_r}{2}\left[\frac{z+L}{\sqrt{(z+L)^2+R^2}} - \frac{z}{\sqrt{z^2+R^2}}\right] \tag{3}$$

Example fit to previously recorded data measured_z_profile

Example fits:

fit_examples

Measured J_r Values

table_magnets

Any Questions?

End of Part 1/3