

Using the IPython Notebook as Lab Notebook

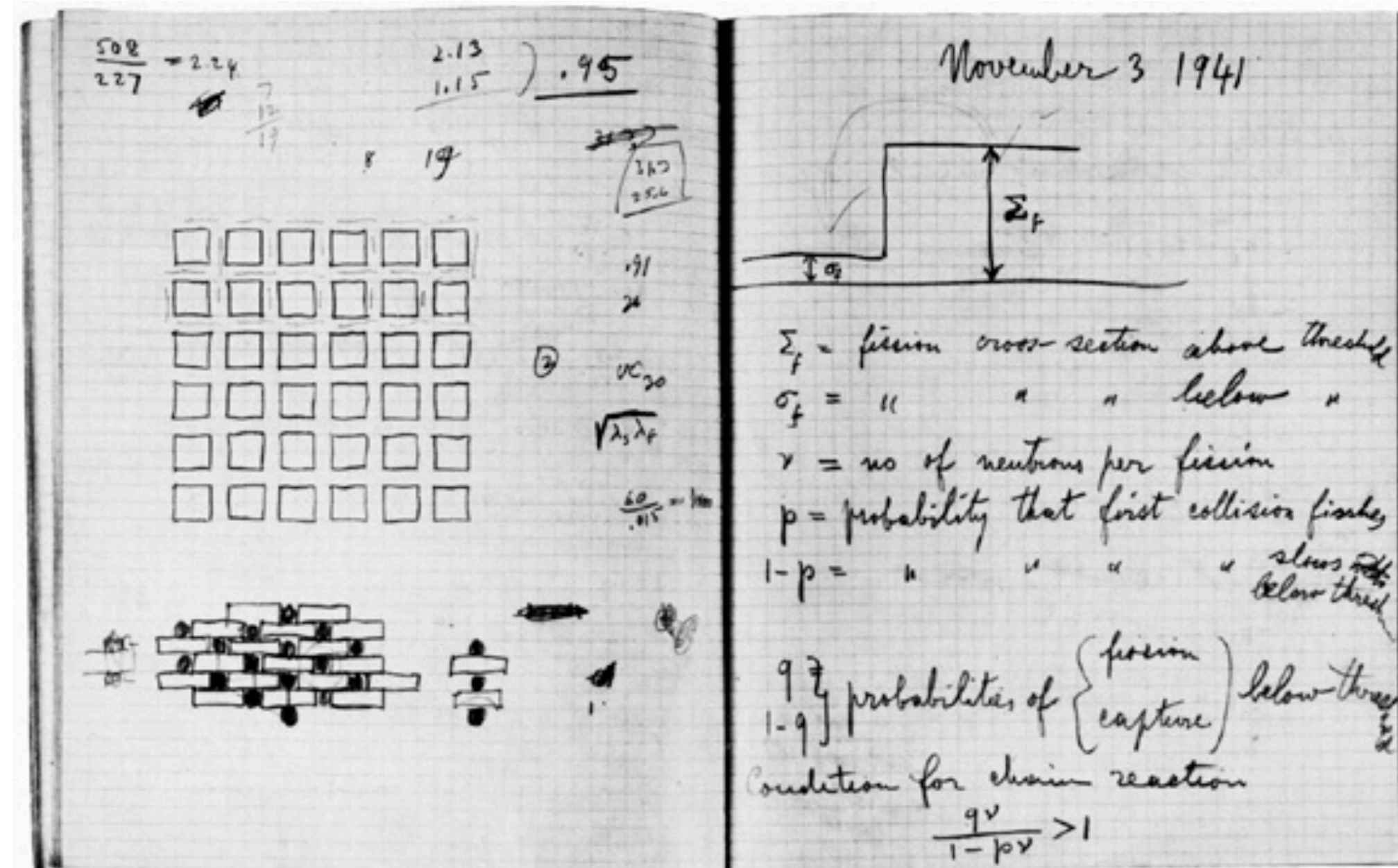
Jürgen Hasch, juergen.hasch@elbonia.de

Abstract

The IPython (Jupyter) notebook comes very close to the ideal electronic version of an engineer's lab notebook. It is easy to use, allows combining text, formulas, graphics, plots and tables, can do live calculations, and can be used to generate static documentation. Customizing the notebook with extensions significantly improves the workflow.

Motivation

Scientists and engineers keep a lab notebook to write down new ideas, do simple calculation or document measurements. It is still the single most important tool, and often the primary means of documentation, even in the computer age. The picture shows a page of Enrico Fermi's lab notebook:



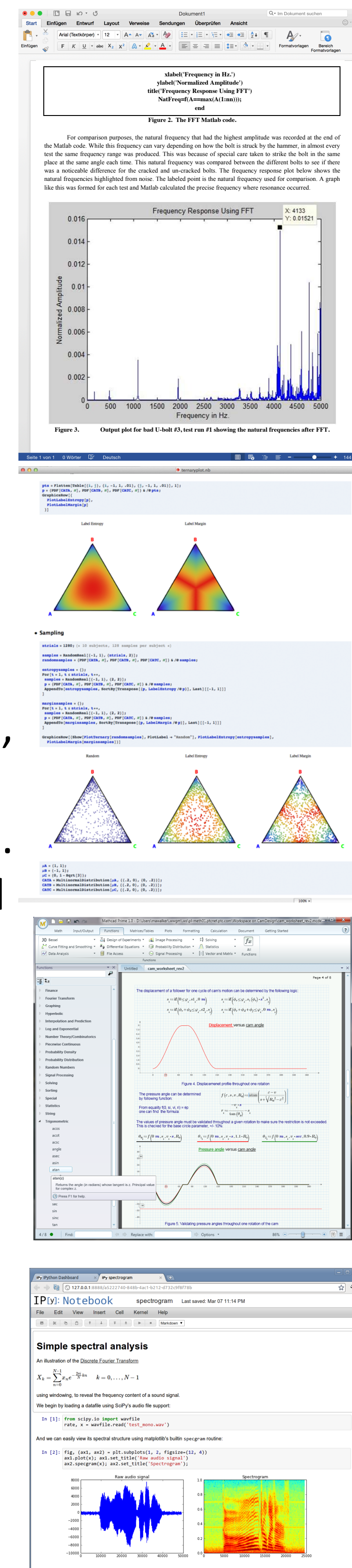
Entries in a lab notebook typically contain a heading, the current date, text, tables, math, drawings, and calculations.

As computers have become an integral part of research and engineering, having a computer-based tool that essentially allows the same workflow as with a handwritten lab notebook becomes inevitable. The proper tool...

- Is easy to use (think pencil and paper)
- Can do text, formulas, graphics, plots, tables
- Can do calculations
- Can be reused
- Is useful for documentation

Computer Tools

- **Word:** The main tool being used even nowadays is Microsoft Word. Shocking but true. For calculations and generating plots, additional tools like Matlab are used.
- **Mathematica:** A powerful tool geared towards mathematical formulation of problems. It is not easy to use (think: pencil and paper), unless one spends considerable time learning it.
- **Mathcad:** Explicitly designed as engineering tool. Allows putting in formulas in mathematical notation and doing actual calculations. My experience: Limited capabilities, closed to the outside world, bugs.
- **IPython notebook:** Allows combining text, formulas, and graphics. And gives you the power of Python.



The Python Ecosystem

So what makes Python and the IPython notebook a good choice?

- Python is a modern general-purpose language with a focus on making it fast and easy to use.
- The ecosystem: There's a package for that! For mathematics, plotting, optimization, data formats, internet protocols.
- The notebook frontend running in a web browser.
- The extensibility: IPython extensions allow adding functionality to the interpreter and the frontend.

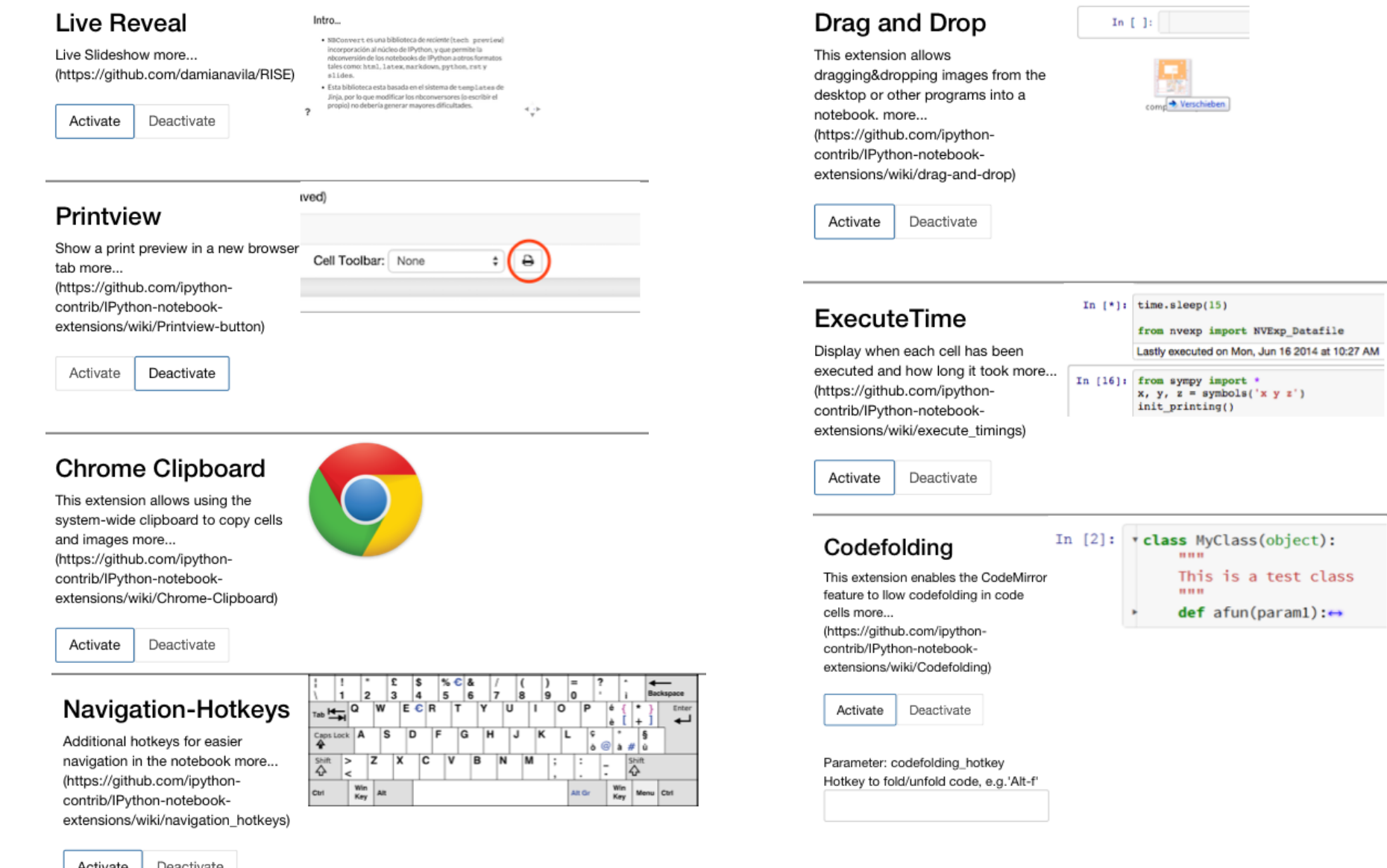
Customization

- IPython extension 'PhysicalQuantities'

```
In [1]: 1m + 10 cm
Out[1]: 1.1 m

In [2]: a = 50 km/h
Out[2]: PhysicalQuantities.Quantity.PhysicalQuantity(a)
```

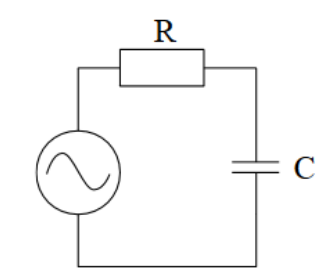
- Notebook extensions



Example

Example: Calculating an RC Circuit

We have a simple RC circuit, consisting of a voltage source U_0 , a resistor with a value R and a capacitor with value C :



Let's start with calculating the capacitance of a simple parallel plate capacitor:

$$C = \frac{\epsilon_0 \cdot \epsilon_r \cdot A}{d}$$

```
In [1]:
# Approximation 2
A = 1 cm * 1 cm # Plate area
d = 1 um # Plate distance
epsilon_r = 3.5 # Permittivity of dielectric material
C = (epsilon_r * A) / d
```

The capacity is C = 0.0556 pF

The resistor is a thick film resistor:

$$R_0 = \frac{\rho}{t} \quad R = R_0 \cdot \frac{L}{W}$$

```
In [2]:
rho = 10 Ohm * 1 mm # Material resistivity
t = 10 um # Sheet thickness
L = 150 um # Structure length
W = 10 um # Structure width
R = rho * L / W
```

Resistor value R = 15.00 kOhm

Calculate transient response of circuit

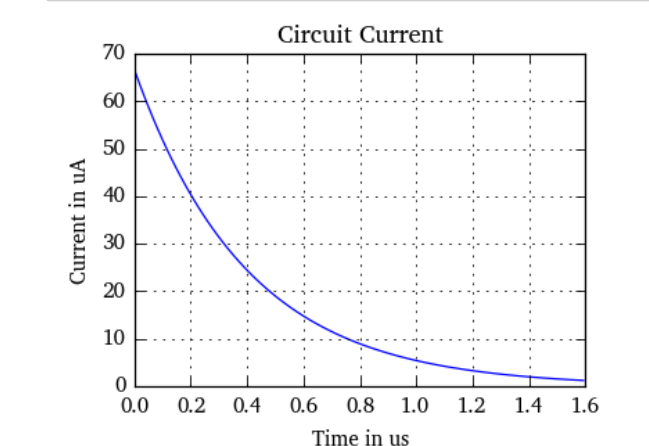
```
In [3]:
u0 = 1 V
tau = L/(R*C)
t = np.linspace(0, 4*tau, 100).us

In [4]:
i0 = u0/R
t = (10*tau - 1*tau).us

The initial current i_0 = 66.67 uA
```

Plot Current and Voltage over Time

```
In [5]:
plt.plot(t, i, 'b-')
plt.grid()
plt.title('Circuit Current')
plt.xlabel('Time in us')
plt.ylabel('Current in mA')
```



Under the Hood

- Graphics

```

```

- Formulas

```
Let's start with calculating the capacitance of a simple parallel plate capacitor:
$$ C = \frac{\epsilon_0 \cdot \epsilon_r \cdot A}{d} \cdot \epsilon_r \cdot A \cdot d $$
```

- Calculation

```
In [2]:
rho = 10 Ohm * 1 mm # Material resistivity
t = 10 um # Sheet thickness
L = 150 um # Structure length
W = 10 um # Structure width
R = rho * L / W
```

- Output in Markdown

```
The capacity is C = 0.0556 pF
```

Another Example

Farfield of a Circular Aperture

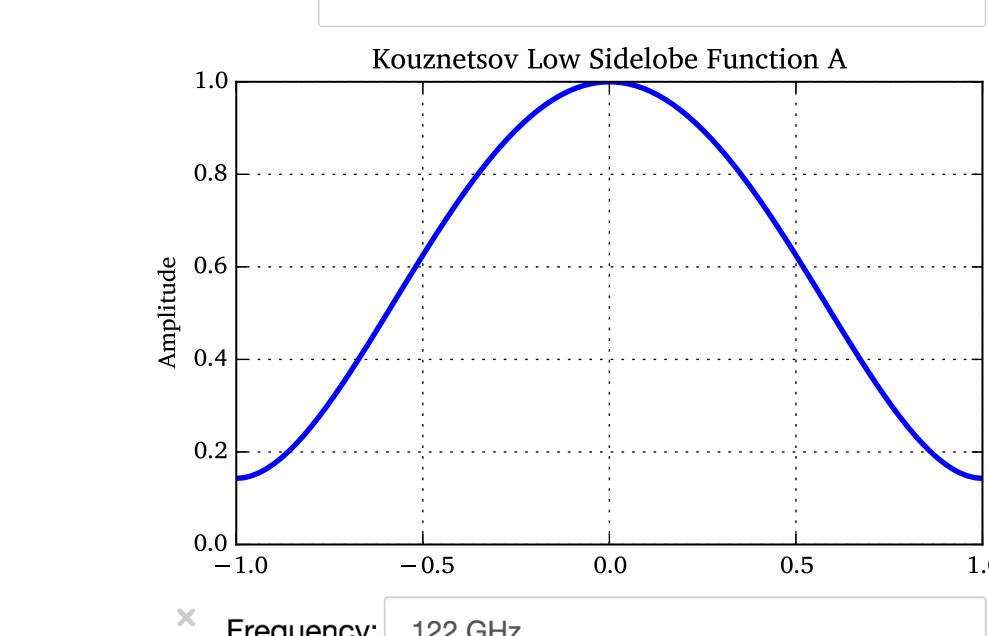
After S.Drabowich, "Modern Antennas"

Tapering Functions

Uniform: $u(x) = 1$
Modern Antennas: $u(x) = 1 - x^2$
Kouznetsov Low SideLobe Function A: $u(x) = 1/7 + 6/7 \cdot (1 - x^2)^2$
Kouznetsov Low SideLobe Function B: $u(x) = 3/29 + 18/29 \cdot (1 - x^2)^2 + 8/29 \cdot (1 - x^2)^3$

Select taper:

Taper from Modern Antennas
Kouznetsov Low SideLobe Function A
Kouznetsov Low SideLobe Function B
Uniform



Frequency: 122 GHz
Diameter: 30 mm

Illumination efficiency: $\eta = 75$

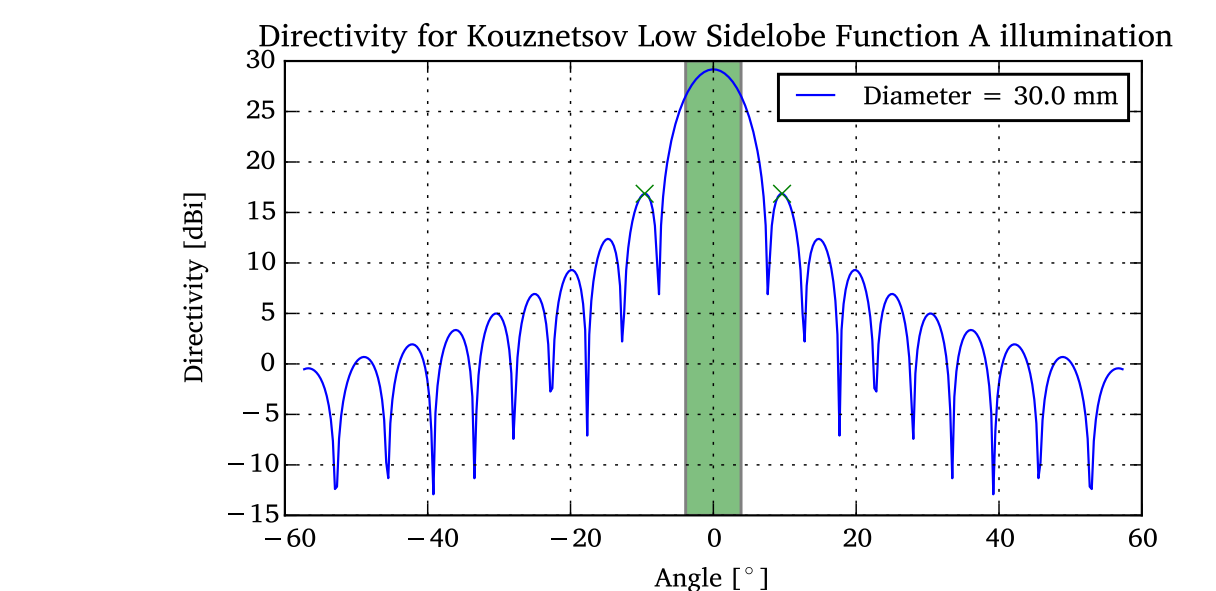
Maximum directivity for uniform illumination: 31.7 dBi

Maximum directivity for given illumination (Kouznetsov Low SideLobe Function A): 29.2 dBi

First zero at $\theta = 7.6^\circ$

First sidelobe at $\theta = 9.6^\circ$ with 12.3 dB attenuation

Half-power beamwidth: -7.8°



Under the Hood

- Widgets

```
In [11]: from IPython.html import widgets
from IPython.display import display
from IPython.html.widgets import interact, interactive, fixed

In [12]: # Define tapering functions
from __future__ import division
tapers = {"Uniform": lambda x: 0.5*x+1,
"Kouznetsov Low SideLobe Function A": lambda x: 1-x**2,
"Kouznetsov Low SideLobe Function B": lambda x: 3/29+18/29*(1-x**2)**2+8/29*(1-x**2)**3,
}
taper_widget = widgets.Select(description='Select taper:',
options=list(tapers.keys()))

In [13]: # Plot tapering function
import matplotlib.pyplot as plt
def select_tapering(taper):
    taper_function = tapers[taper]
    plt.figure(figsize=(10,10))
    plt.plot(taper_function, lw=2)
    plt.grid()
    plt.title(taper)
    plt.ylabel('Amplitude')
    interact(select_tapering, taper=taper_widget)
```

- Python in markdown

```
In [40]: # Calculate HPBW, first zero and first sidelobe
import numpy as np
minimas = signal.argmin(abs(H(0:N/2)))
maximas = signal.argmax(abs(H(0:N/2)))
theta_min = -theta[minimas]*180/pi
theta_max = theta[maximas]*180/pi
sll = -d_max - d[maximas]*180/pi
idx = where(h > 0.5)[0]
hpbw = 2*theta[idx]*180/pi

First zero at $theta_0 = {{ '%2.1f' % theta_min }}-br>
First sidelobe at $theta_s = {{ '%2.1f' with %2.1f dB attenuation' % (theta_max, sll)}}-br>
Half-power beamwidth: {{ '%2.1f' % hpbw }}
```

Wish List

- Hierarchy, allow to collapse parts of a notebook
- Keep notebook and data together ('projects')
- Versioning, visual diffing
- Templates based on existing notebooks
- Easy Export into something 'looking good'