Python: Data Analysis

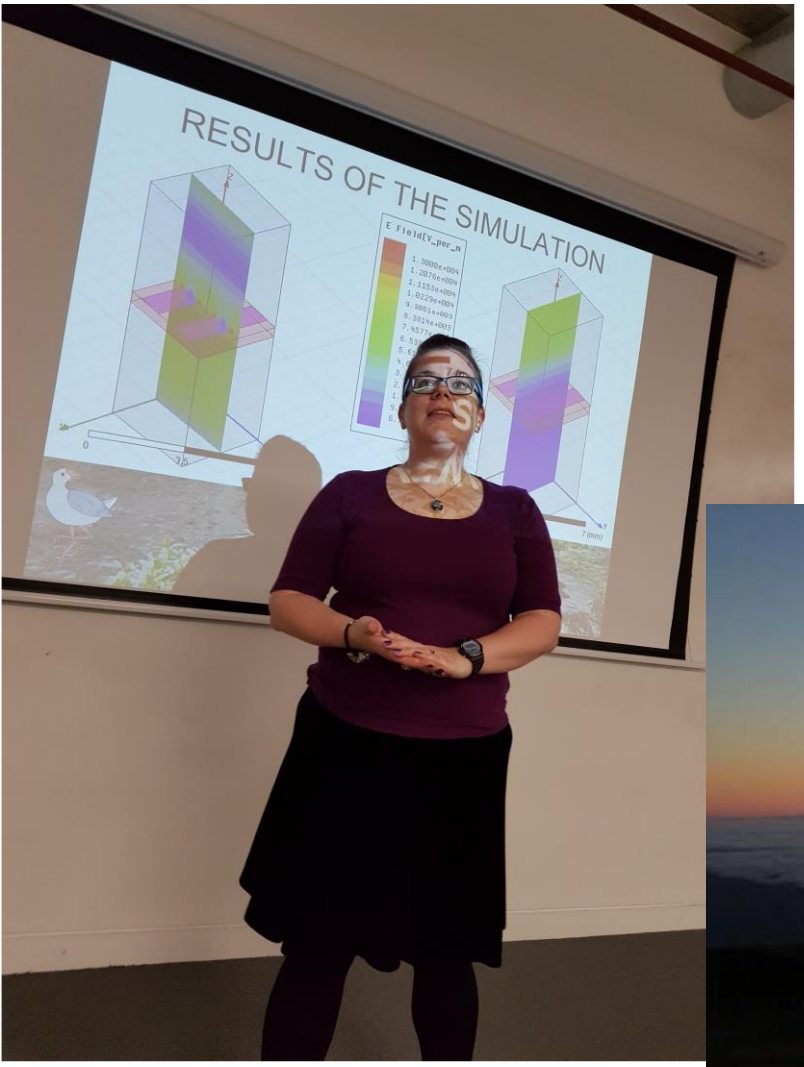
Dr Tamara Clelford

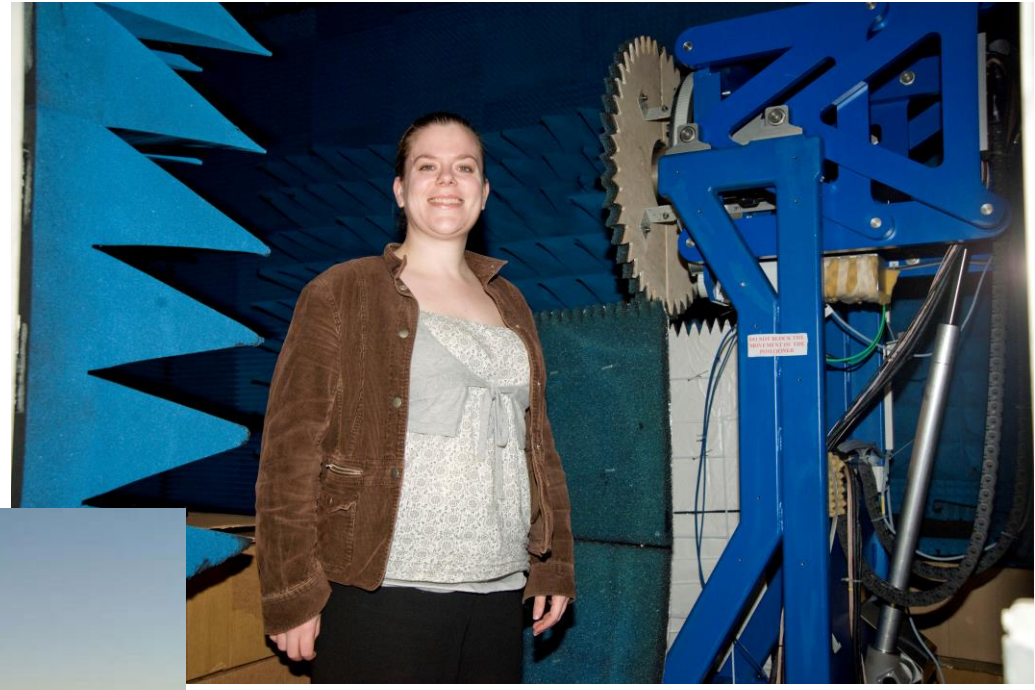
To download a copy of the slides go to
https://tamaraclelford.co.uk/iop.html

DATA ANALYSIS IN PYTHON

- How to read in and write out csv files using pandas
- How to plot a graph using Matplotlib
- How to write functions in Python
- How to write a simple simulation in Python
- Ideas on data analysis in Python

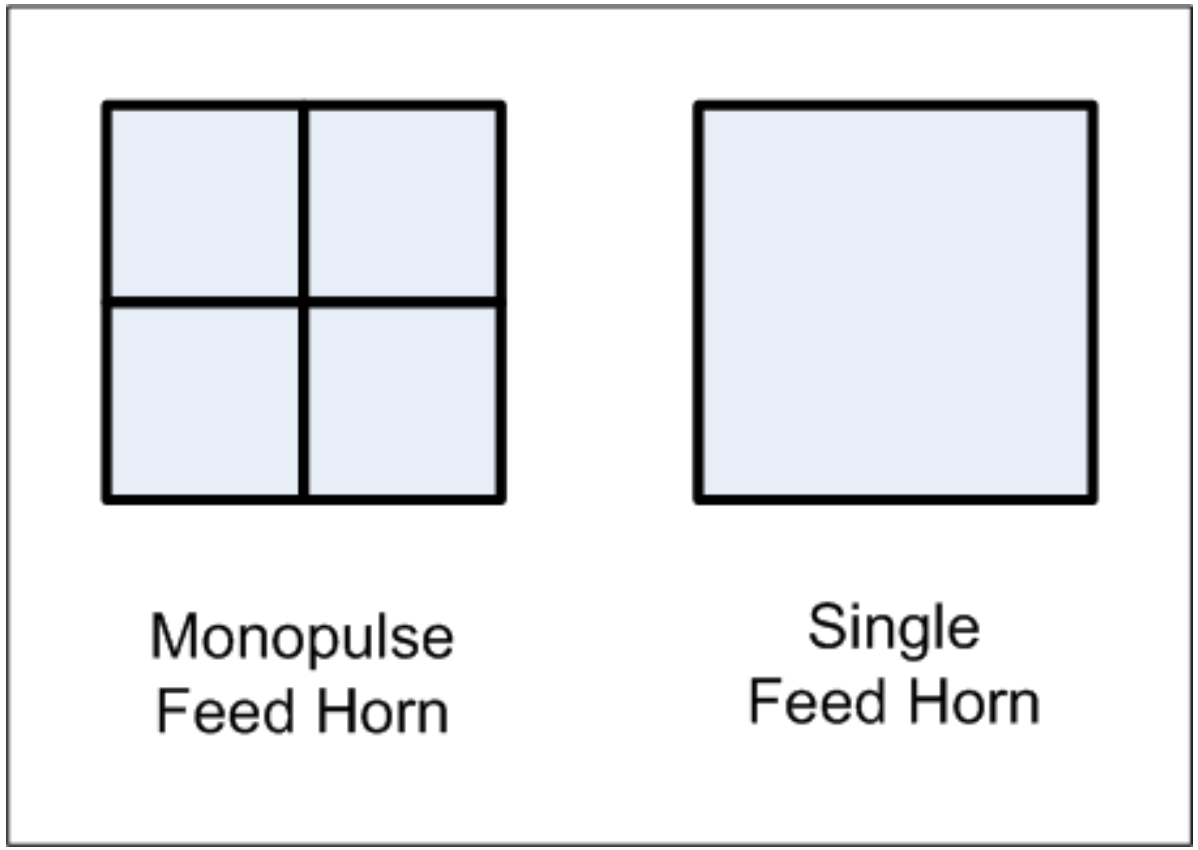
ABOUT ME



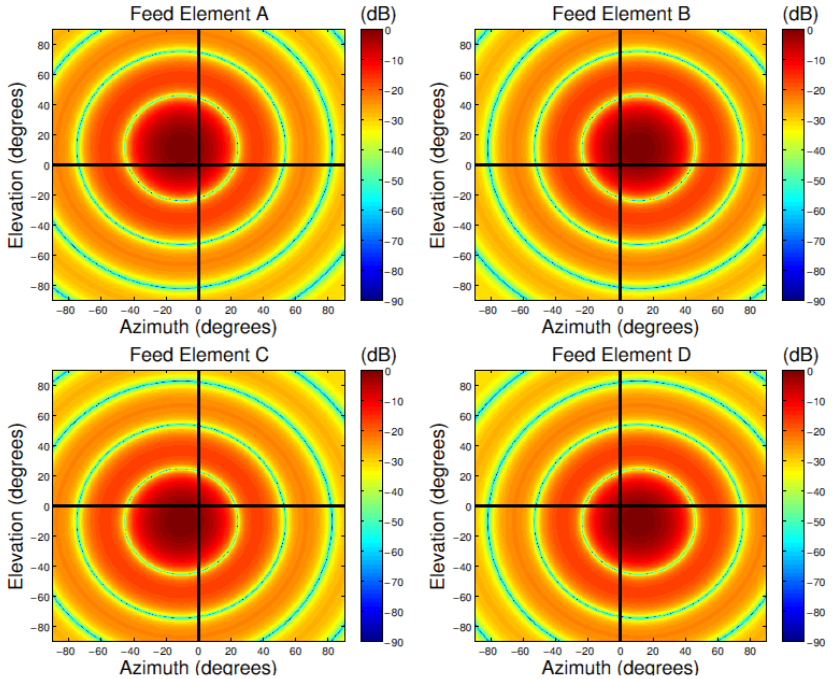


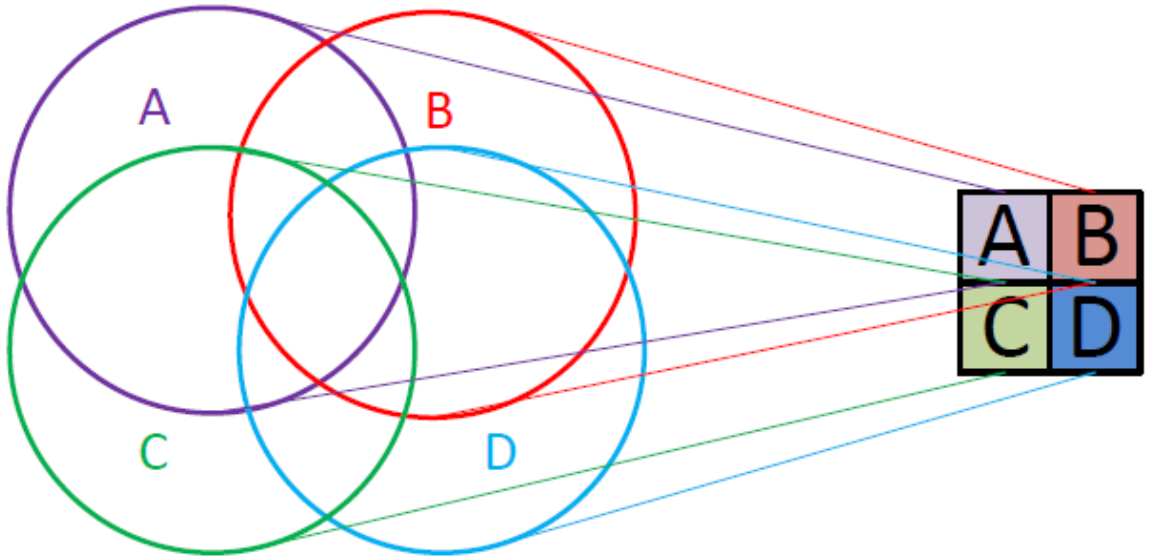
https://www.hawaii.edu/news/2015/10/21/third-maunakea-observatory-set-for-decommissioning/

MULTI-CHANNEL ANTENNAS

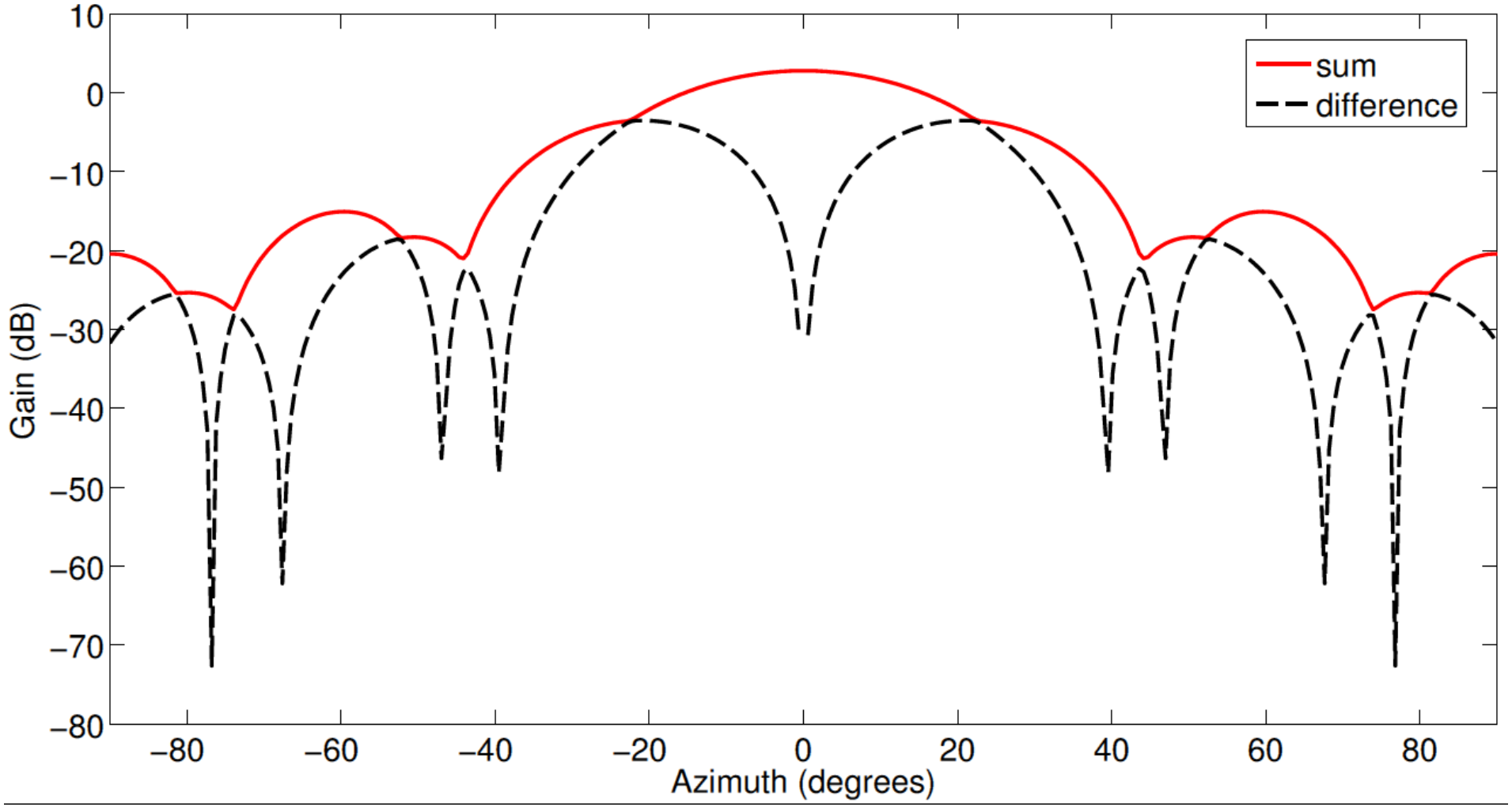


MULTI-CHANNEL ANTENNAS

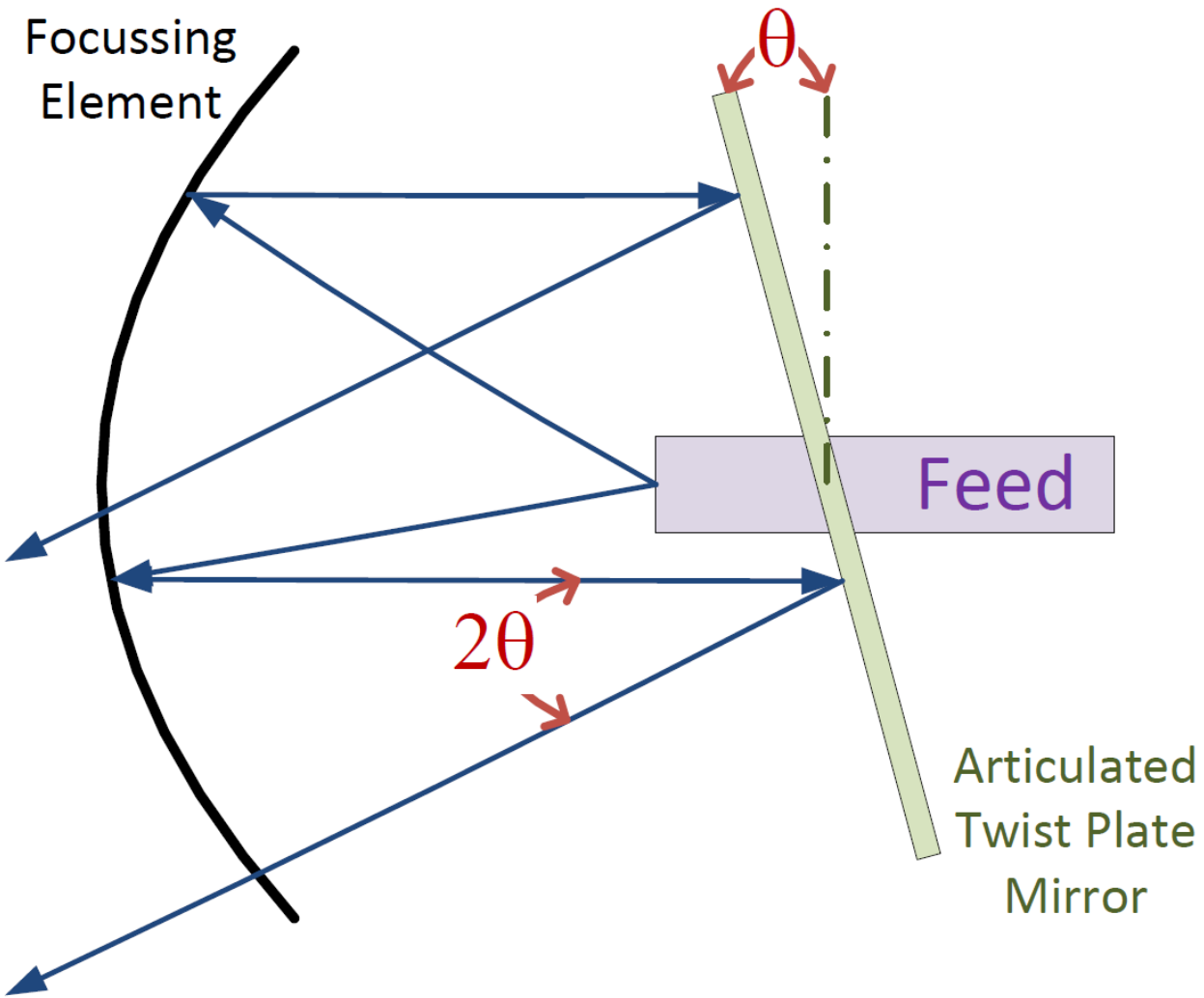




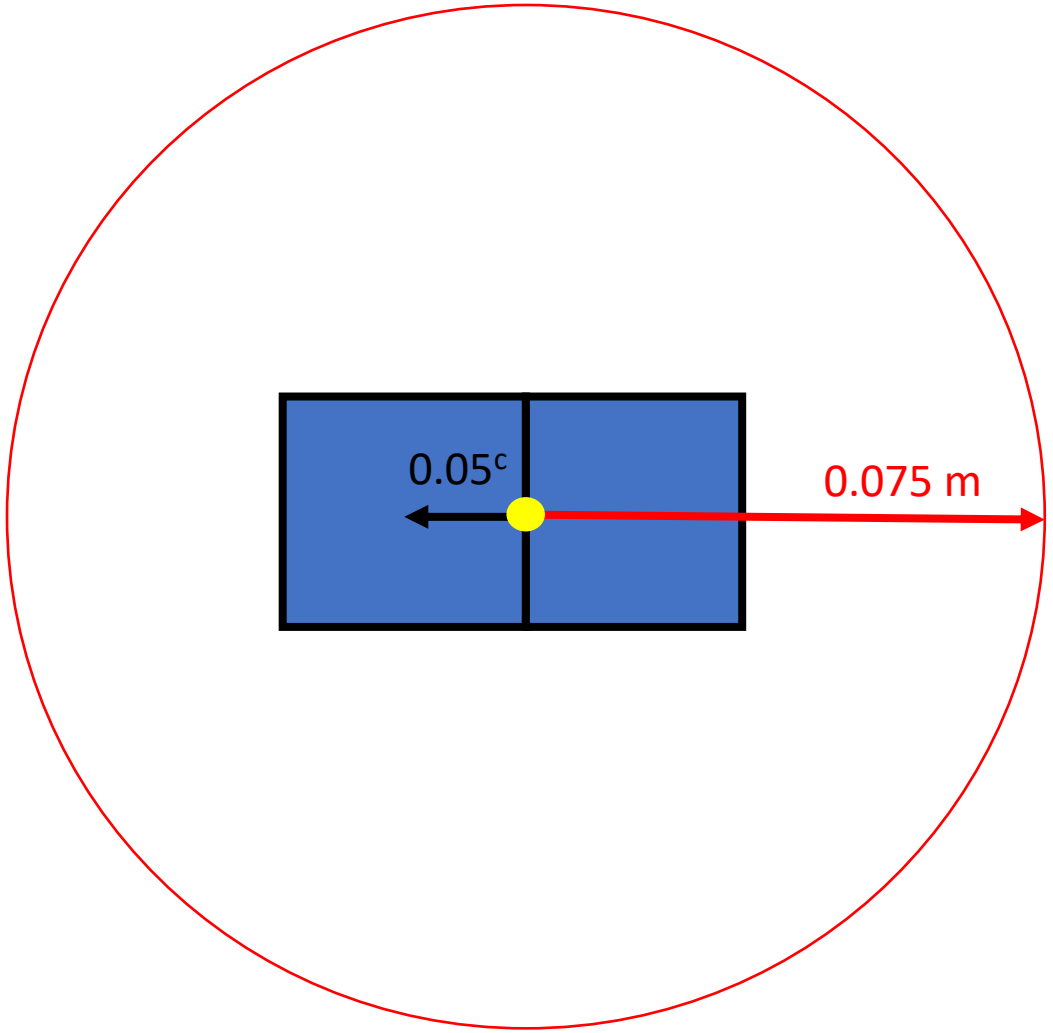
SUM AND DIFFERENCE SIGNAL



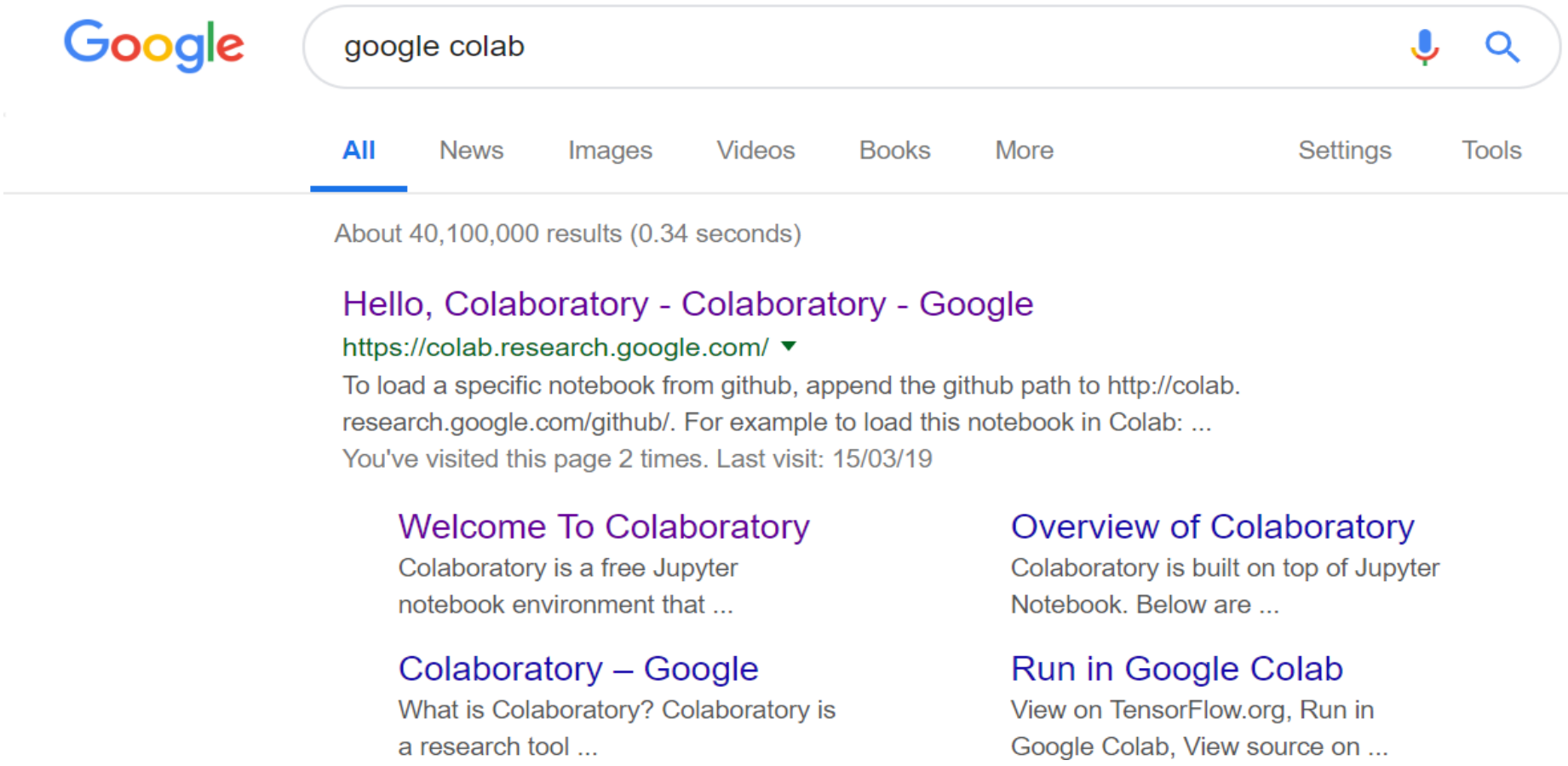
TWIST REFLECTOR ANTENNA



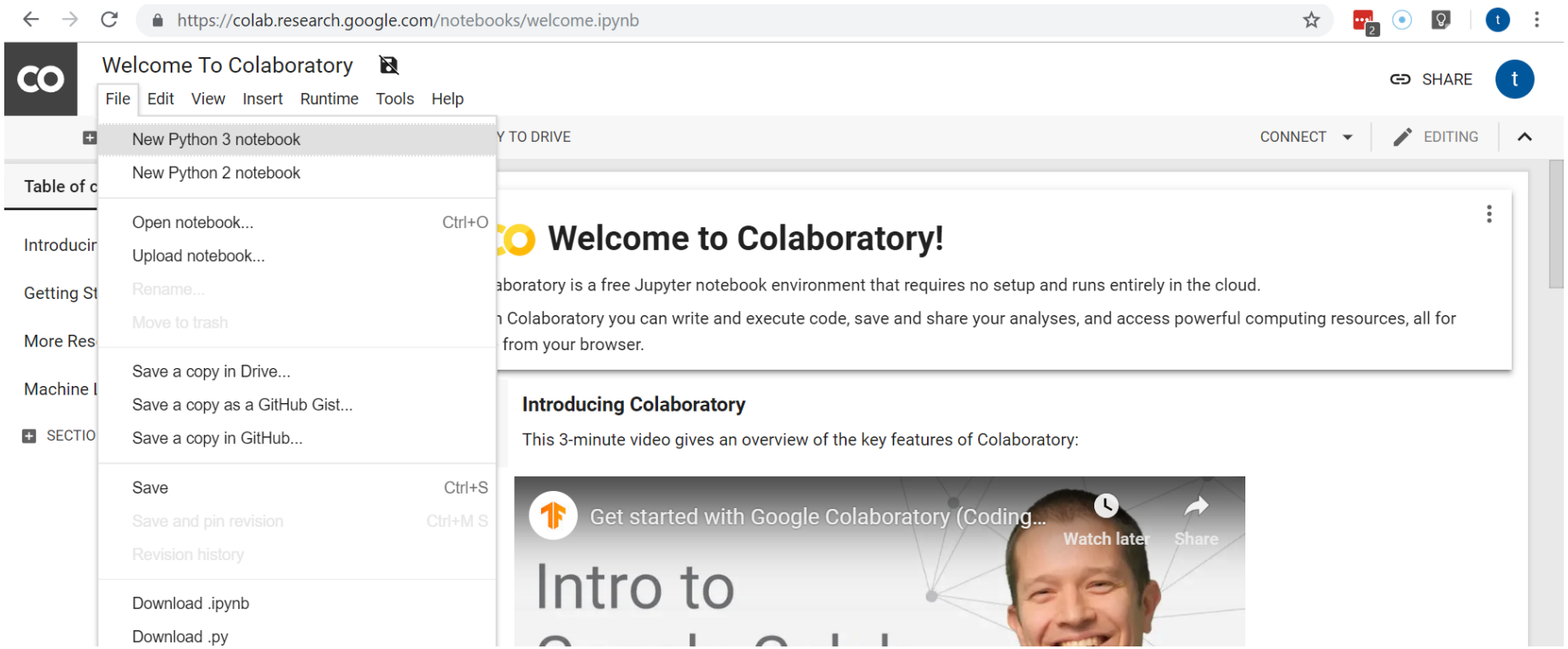
TODAY'S MONOPULSE ANTENNA SYSTEM



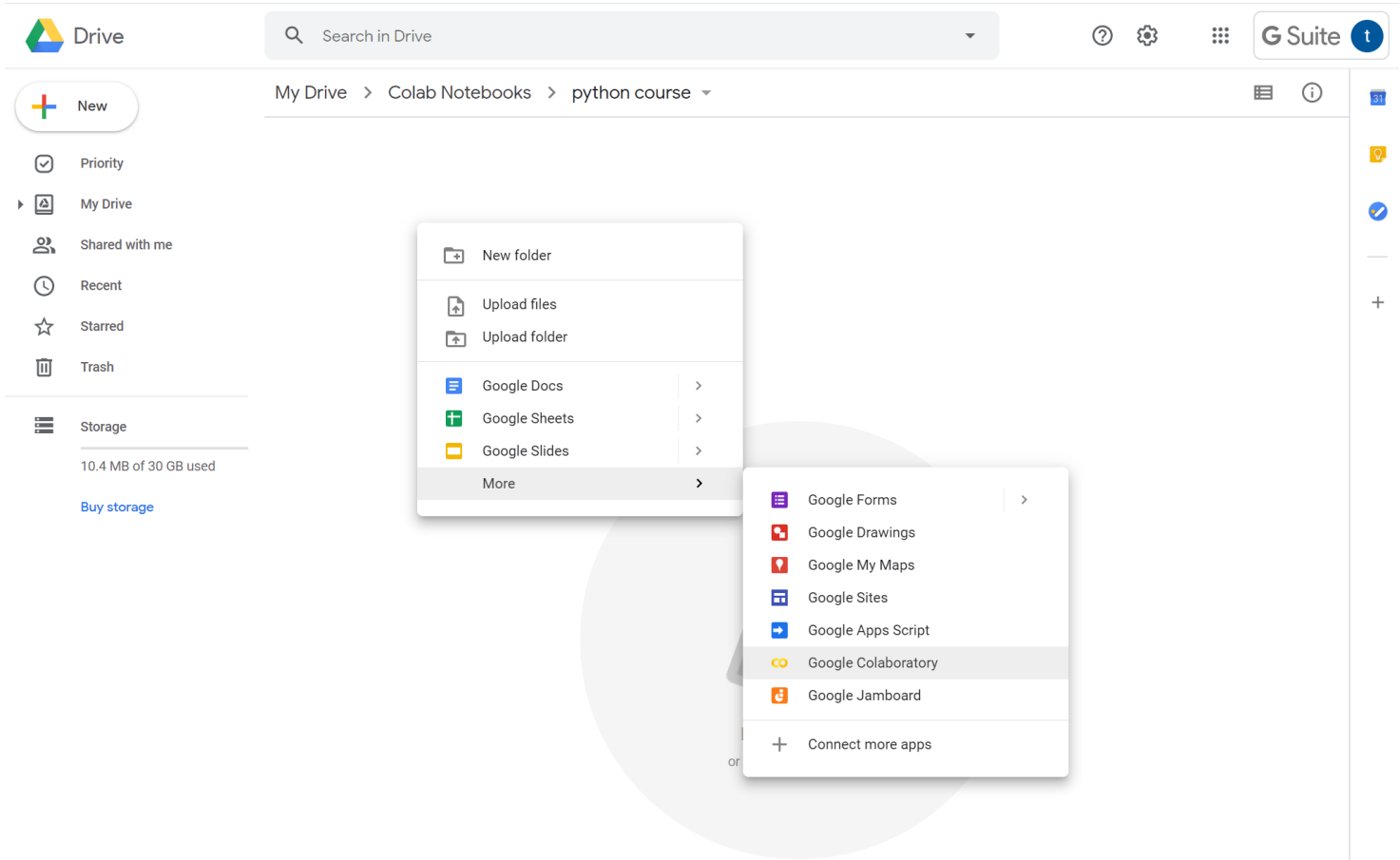
GO TO GOOGLE COLAB
https://colab.research.google.com



OPEN A NEW NOTEBOOK



CREATE A NEW ‘GOOGLE COLAB’ FOR TODAY



READ DATA IN

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from google.colab import files
uploaded = files.upload()
```

```
df = pd.read_csv('measured_data.csv')
```

LOOK AT DATA

```
df.head()
```

```
# select a column
df.sum_m_db
# add two columns
df.sum_m_db + df.diff_m_db
```

```
# can create fully bespoke graphs
plt.plot(df.angle_deg, df.sum_m_db, 'g:', label = 'measured sum')
plt.plot(df.angle_deg, df.diff_m_db, 'b', label = 'measured diff')
plt.xlabel('Angle (degrees)')
plt.ylabel('Gain (dB)')
plt.legend(loc = 'upper right')
```

```
# create function to label axes and put in legend as each graph
# will be the same
def label_graph():
    plt.xlabel('Angle (degrees)')
    plt.ylabel('Gain (dB)')
    plt.legend(loc = 'best')
    return
```

matplotlib.org/3.1.1/index.html

matplotlib

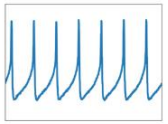
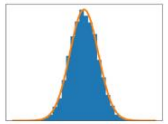
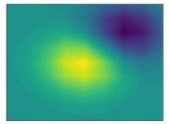
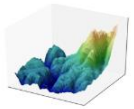
Version 3.1.2

Fork me on GitHub

Installation | Documentation | Examples | Tutorials | Contributing

home | contents » modules | index

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Documentation

This is the documentation for Matplotlib version 3.1.2.

To get started, read the User's Guide.

Latest release
3.3.0: docs | changelog

Last release for Python 2
2.2.5: docs | changelog

Development version
docs

Trying to learn how to do a particular kind of plot? Check out the examples gallery or the list of plotting commands.

Other learning resources

There are many external learning resources available including printed material, videos and tutorials.

Quick search
Go

Matplotlib 3.0 is Python 3 only.

For Python 2 support, Matplotlib 2.2.x will be continued as a LTS release and updated with bugfixes until January 1, 2020.

Support Matplotlib

GRAPHS

- Colours to choose from
  - b
  - g
  - k
  - r
  - c
  - m
  - y
  - w
  - 'purple'

LINE GRAPHS

- move the legend around, options are:
  - upper left
  - upper right
  - lower left
  - lower right
  - best
  - upper center
  - lower center
  - center left
  - center right

LINE GRAPHS

- Options for line types
  - - solid line
  - -- dashed line
  - -. dash dot line
  - : dotted line

DATA POINTS

- Marker options
  - . point
  - , pixel
  - o circle
  - v triangle_down
  - ^ triangle_up
  - < triangle_left
  - > triangle_right
  - 1 tri_down
  - 2 tri_up
  - 3 tri_left
  - 4 tri_right
  - s square
  - p pentagon
  - * star
  - h hexagon1
  - H hexagon2
  - + plus
  - x x
  - D diamond
  - d thin_diamond
  - | vline
  - _ hline

CREATE SIMULATION TO COMPARE TO MEASURED DATA

```
# defined parameters
diameter = .15
wavelength = 3e8 / 15e9
channel_offset = 0.05
```

```
# convert angle range to radians
angle_rad = np.radians(list(range(-30,31)))
print(angle_rad)
```

```
# sinc function sinu/u for diffraction pattern from an aperture
# calculate the left u - ratio of aperture to wavelength as function
# of angle
u_left = ((np.pi * diameter)/wavelength) * (angle_rad - channel_offset)
print(u_left)
```

```
left = np.sin(u_left) / u_left
print(left)
```

PLOT SIMULATION

```python
# going to need to convert to db a lot so create function
def v_to_db(v):
    db = 10*(np.log10(v**2))
    return(db)
```

```python
plt.plot(df.angle_deg, v_to_db(left), label = 'left')
label_graph()
```

```python
u_right = ((np.pi * diameter)/wavelength) * (angle_rad + channel_offset)
right = np.sin(u_right) / u_right
```

```python
plt.plot(df.angle_deg, v_to_db(left), label = 'left')
plt.plot(df.angle_deg, v_to_db(right), label = 'right')
label_graph()
```

```python
#sum_s_v = left + right
#diff_s_v = left - right
sum_s_db = v_to_db(left + right)
diff_s_db = v_to_db(left - right)
```

```python
plt.plot(df.angle_deg, sum_s_db, label = 'sum')
plt.plot(df.angle_deg, diff_s_db, label = 'difference')
label_graph()
```

DATA ANALYSIS

```
plt.plot(df.angle_deg, sum_s_db, label = 'simulated sum')
plt.plot(df.angle_deg, df.sum_m_db, label = 'measured sum')
label_graph()
# download a copy of your graph
plt.savefig('simulated_v_measured_sum.png')
```

```
from google.colab import files
files.download("simulated_v_measured_sum.png")
```

```
# need to be both np array's or pandas data frames
sum_d_db = np.array(df.sum_m_db) - sum_s_db
#diff_d_db = np.array(df.diff_m_db) - diff_s_db
print(sum_d_db)
```

```
plt.plot(df.angle_deg, sum_d_db, label = 'difference sum')
# plt.plot(df.angle_deg, diff_d_db, label = 'difference difference')
label_graph()
```

ADD NEW DATA TO EXISTING CSV FILE

```
df.head()
```

```
# add to data frame to write out
df['sum_s_db'] = sum_s_db
df['diff_s_db'] = diff_s_db
df.head()
```

WRITE OUT CSV FILE

```
df.to_csv('meas_sim_data.csv', index=False)
```

```
from google.colab import files
files.download("meas_sim_data.csv")
```

HOW CAN I LEARN MORE?

- Practice
- O’Reilly books
- Google search

QUESTIONS!

- Dr Tamara Clelford

https://tamaraclelford.co.uk/online_courses.html