measure_power

May 23, 2022

[2]:
```
import sys
sys.path.append("gpib_instrument_control/")
import lmx2594
import gpib_instrument_control.hp_3478a
import gpib_instrument_control.pm_1038

meter = gpib_instrument_control.hp_3478a.Hp3478A()
pm = gpib_instrument_control.pm_1038.Pm1038(None, meter,  None,␣
 ↪"pm11-0674_correctionFactors.mat")

lmx = lmx2594.Lmx2594('/dev/ttyUSB0', 320e6)

lmx.enableLockDetect(True)
print("Is locked?", lmx.isLocked())
print(pm.readChannelB(50e6))
```

```
Created LMX object wit fosc 320.0
Resetting LMX
Applying config
Is locked:  True
Is locked? True
-7.598000000000001
```

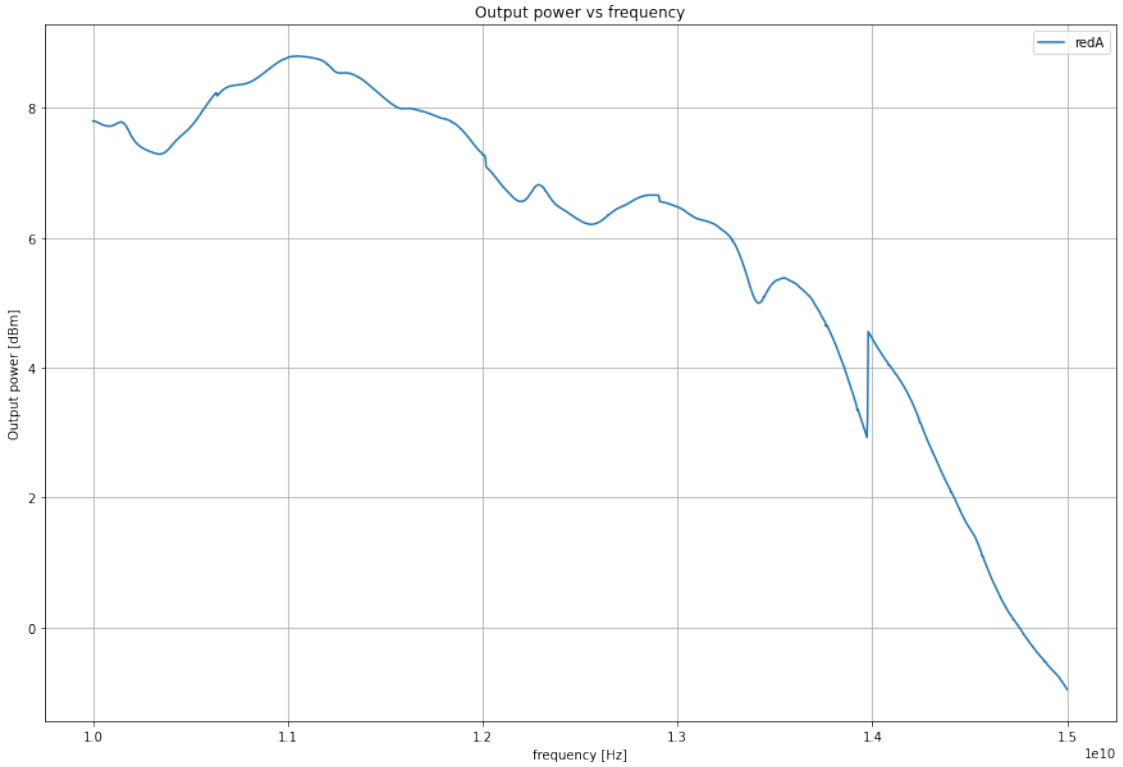
[16]:
```
import numpy as np
import matplotlib.pyplot as plt
import time
%matplotlib inline
plt.rcParams['figure.figsize'] = [15, 10]

def powerSweep(frange):
    ampl = []
    lmx.setFrequency(frange[0])
    time.sleep(1)
    for f in frange:
        lmx.setFrequency(f)
        ampl.append(pm.readChannelB(f))
    return ampl
```

```
#frange = np.linspace(1e9, 15e9, 30)
#plt.plot(frange, ampl)
```

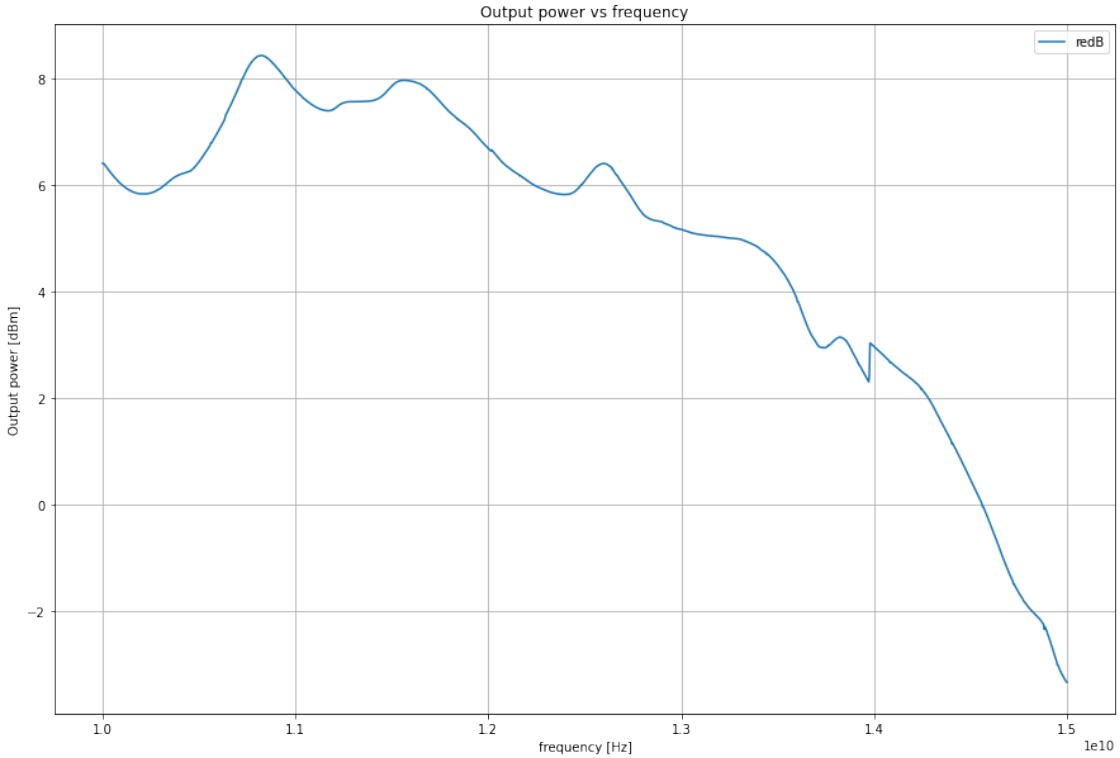
[18]:
```
frange = np.linspace(10e9, 15e9, 1500)
amplRedA = powerSweep(frange)
plt.plot(frange, amplRedA, label='redA')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[18]:
```
Text(0.5, 0, 'frequency [Hz]')
```



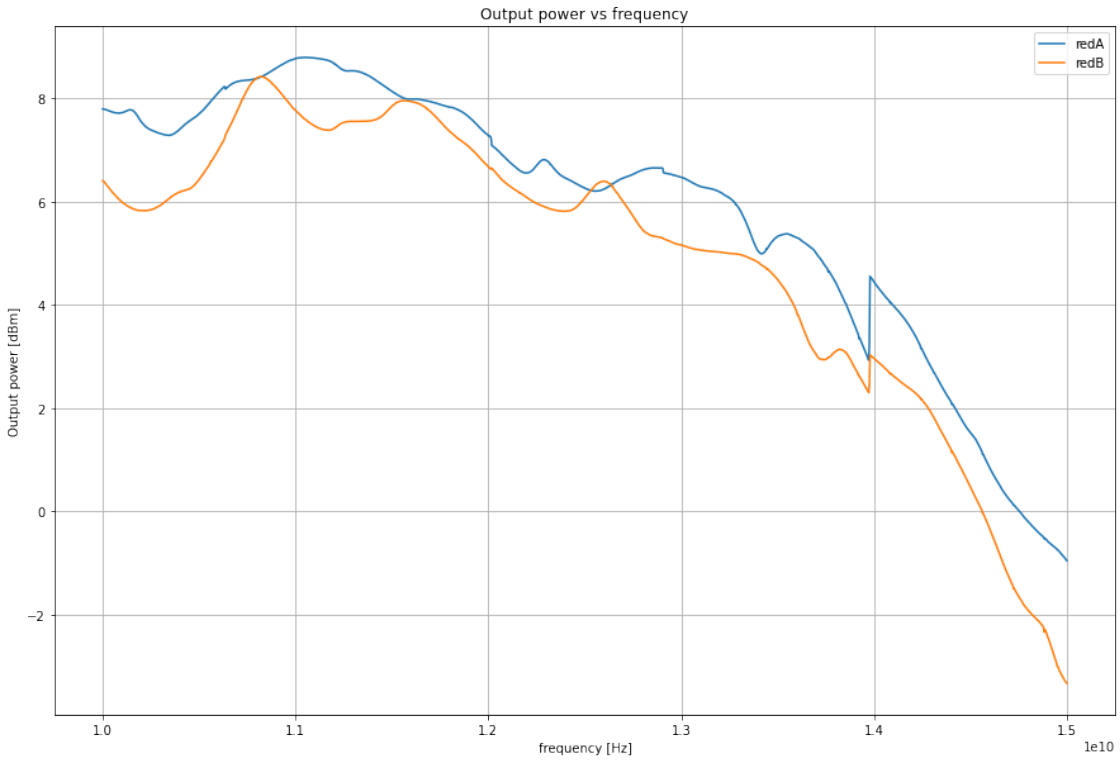
[20]:
```
frange = np.linspace(10e9, 15e9, 1500)
amplRedB = powerSweep(frange)
plt.plot(frange, amplRedB, label='redB')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[20]: Text(0.5, 0, 'frequency [Hz]')



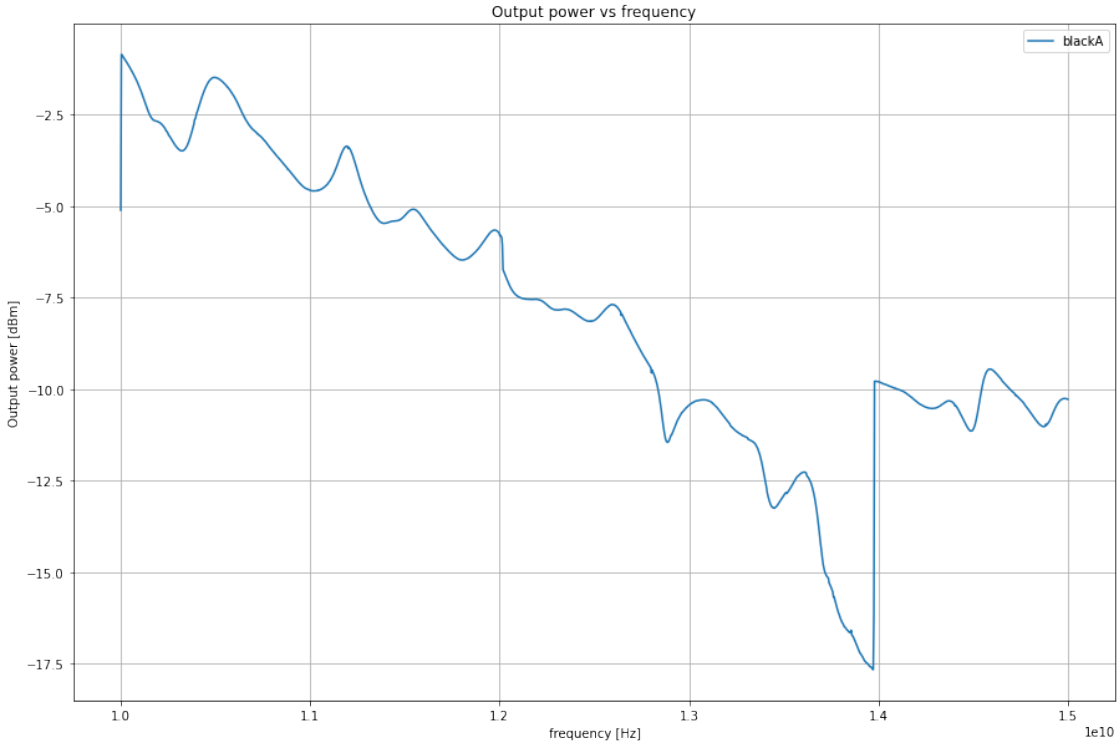
[21]:
```
plt.plot(frange, amplRedA, label='redA')
plt.plot(frange, amplRedB, label='redB')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[21]: Text(0.5, 0, 'frequency [Hz]')



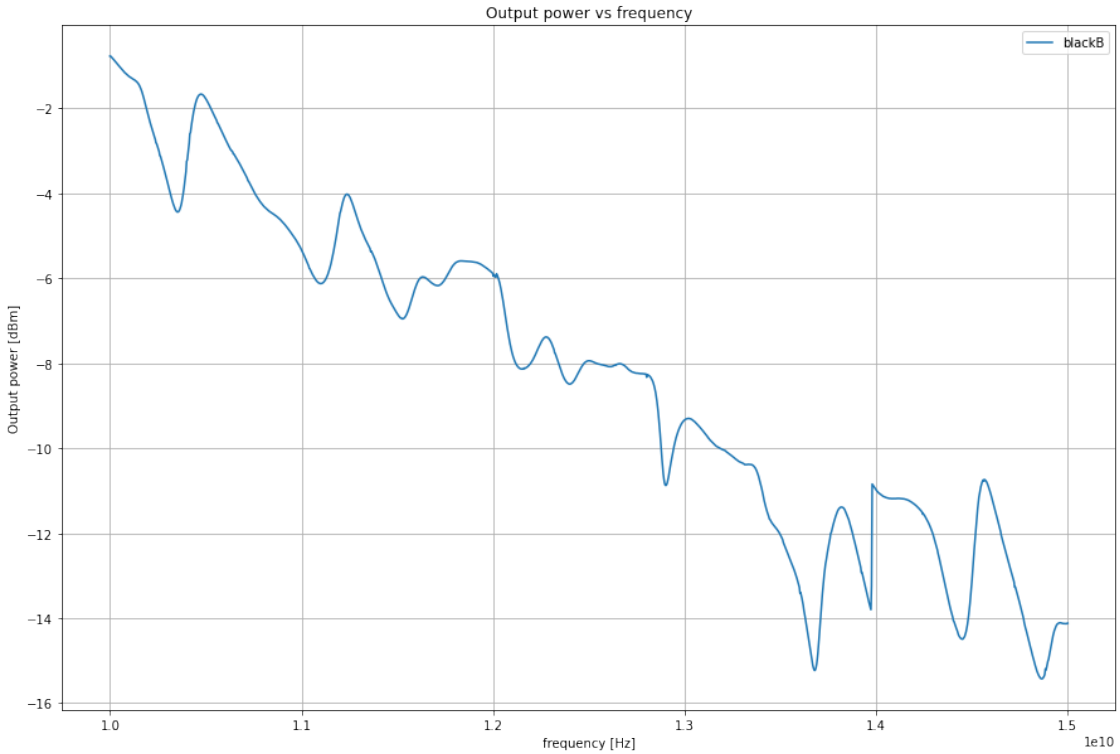
[19]:
```
#frange = np.linspace(1e9, 15e9, 30)
amplBlackA = powerSweep(frange)
plt.plot(frange, amplBlackA, label='blackA')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[19]:
```
Text(0.5, 0, 'frequency [Hz]')
```



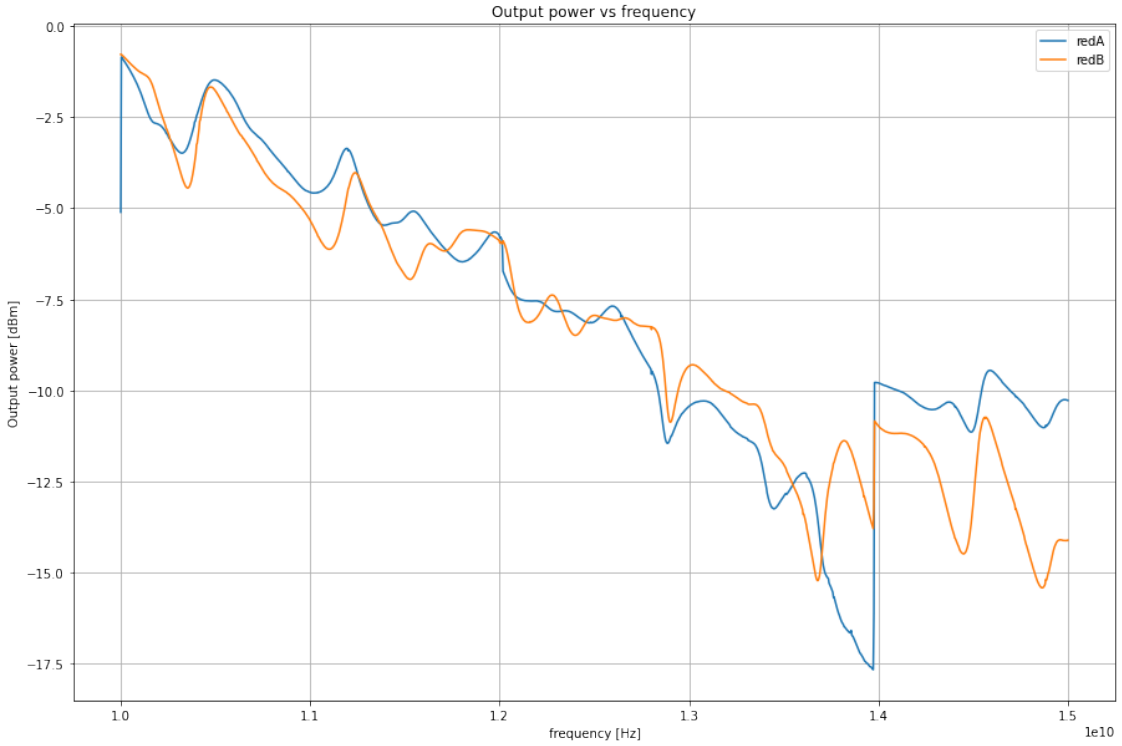
[22]:
```
amplBlackB = powerSweep(frange)
plt.plot(frange, amplBlackB, label='blackB')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[22]:
```
Text(0.5, 0, 'frequency [Hz]')
```



[23]:
```
plt.plot(frange, amplBlackA, label='redA')
plt.plot(frange, amplBlackB, label='redB')
plt.grid(True)
plt.legend()
plt.title('Output power vs frequency')
plt.ylabel('Output power [dBm]')
plt.xlabel('frequency [Hz]')
```

[23]:
```
Text(0.5, 0, 'frequency [Hz]')
```



[25]:
```
import scipy.io as sio

sm = {'f':frange,
            'redA': amplRedA,
            'redB': amplRedB,
            'redA': amplBlackA,
            'redB': amplBlackB,
           }

sio.savemat('amplitude_data.mat', sm)
```