tes_lab_analysis_solution

August 14, 2022

1 Analysis of TES Lab Data

In the TES lab you overbiased a module of detectors, then took and an IV curve by decreasing the voltage bias until the detector dropped into its superconducting transition. In this notebook you will extract the saturation power $P_{\rm sat}$ of each detector from these data. The $P_{\rm sat}$ is an important design target of TES bolometers. By analyzing $P_{\rm sat}$ measurements across a range of bath temperatures, you will fit a simple model of the TES thermal circuit, which will allow you to predict properties like $P_{\rm sat}$ and the detector noise under a range of operating conditions.

1.1 Import useful modules

We will need some standard python tools to work with the data you took:

- `pickle` - for loading data from pydfmux
- `numpy` - for manipulating data structures
- `matplotlib` - for plotting
- `from scipy.optimize import curve_fit` - for fitting $P_{\rm sat}(T_{\rm bath})$ data
- `os` - for path manipulations

[3]:
```
import pickle
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import os
```

1.2 Load data

Now, let's load the data, editing the path and filename below to point to your "drop_bolos" data (a.k.a. the IV curves).

[10]:
```
data_path = 'pydfmux_output/summer2022/20220811/20220811_163941_drop_bolos/data'
fname = 'IceBoard_0028.Mezz_1.ReadoutModule_1_OUTPUT.pkl'
with open(os.path.join(data_path, fname), 'rb') as f:
    data = pickle.load(f)
```

The data are structured as nested dictionary. Using the `.keys()` function, traverse through the dictionary to see what information is available. For example:

[11]:
```
data.keys()
```

[11]:
```
dict_keys(['RIV_log', 'subtargets', 'target', 'reduced_target', 'pstring',
'results_summary', 'current', 'pre_drop', 'oscillating_chans', 'dropped_chans',
'post_drop', 'outcome'])
```

[12]:
```
data['RIV_log'].keys()
```

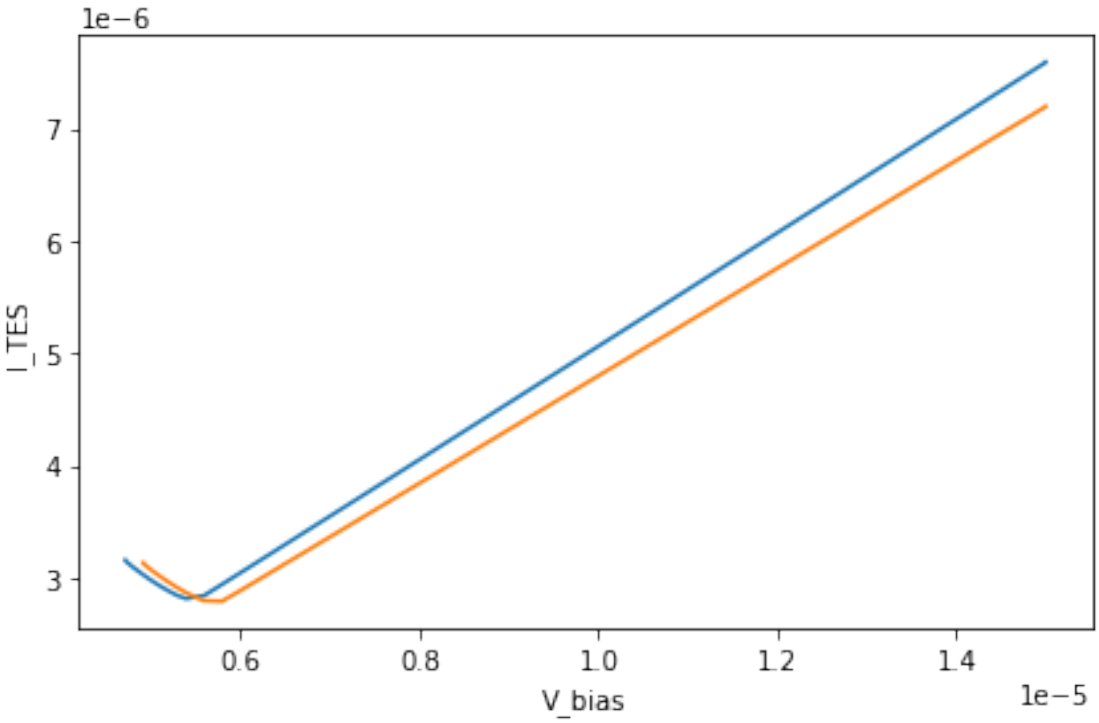
[12]:
```
dict_keys([1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 59, 61, 62])
```

1.3 Plot I vs. V

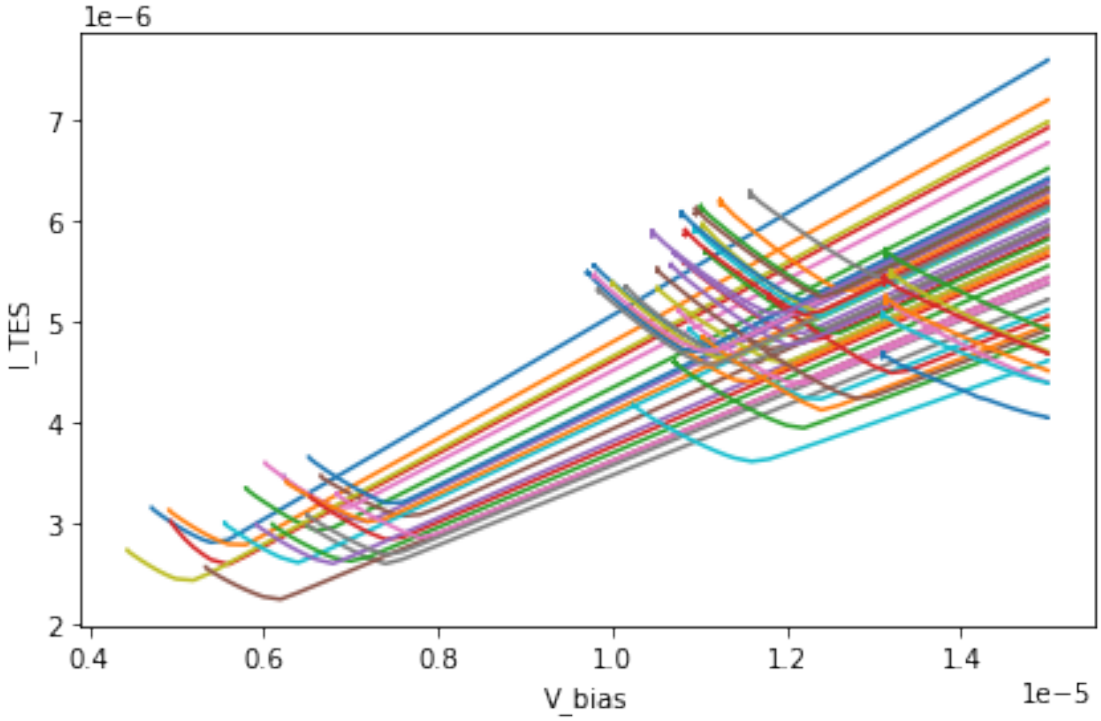
In the data dictionary, find the field with current and voltage for a detector. Plot these two variables. How do you interpret the different sections of the IV curve?

Repeat this exercise for many detectors. Do all the detectors look similar?

[13]:
```
plt.plot(data['RIV_log'][1]['V'], data['RIV_log'][1]['I'])
plt.plot(data['RIV_log'][2]['V'], data['RIV_log'][2]['I'])
plt.xlabel('V_bias')
plt.ylabel('I_TES')
plt.tight_layout()
```



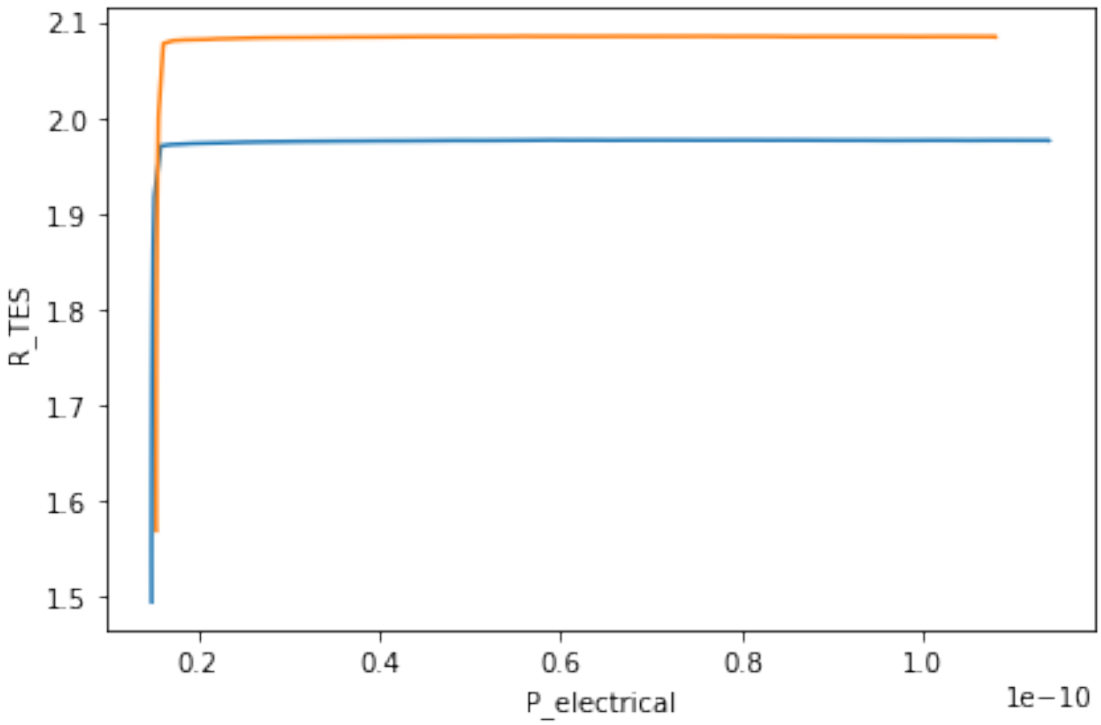
[14]:
```
for detector in data['RIV_log']:
    plt.plot(data['RIV_log'][detector]['V'], data['RIV_log'][detector]['I'])
plt.xlabel('V_bias')
plt.ylabel('I_TES')
plt.tight_layout()
```



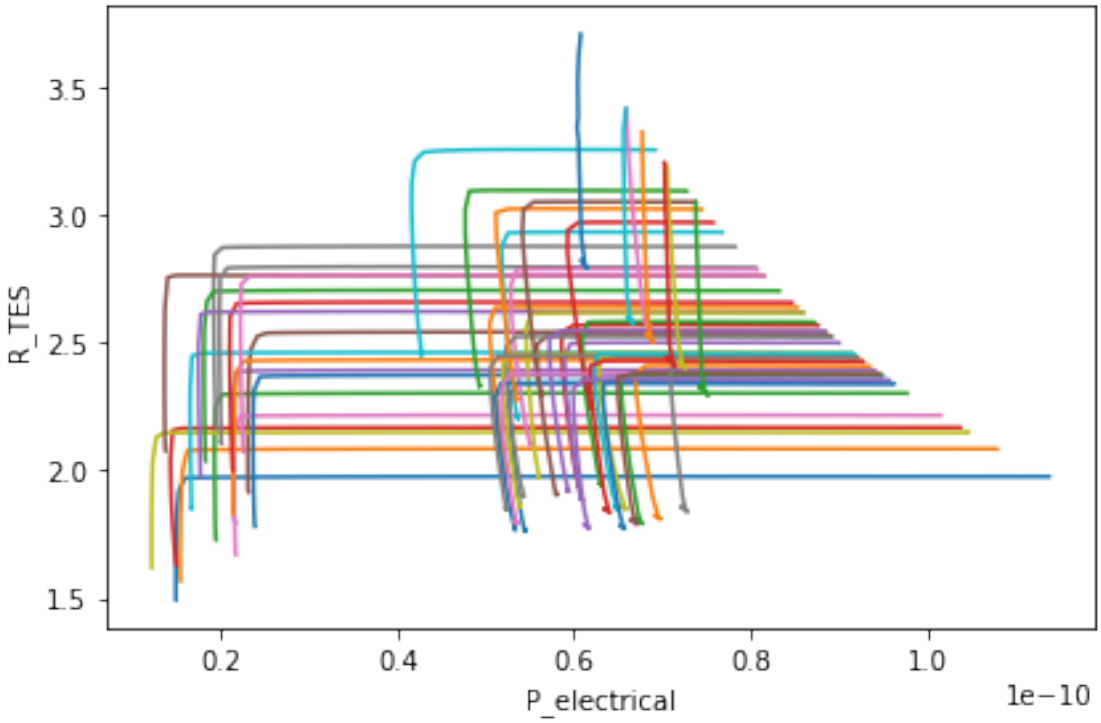
1.4 Plot P vs. R

Calculate the Joule power across a detector and its resistance, using I and V that you found above. P and R are also calculated automatically. How does the AC bias of the FDM architecture affect your calculation of P?

[15]:
```
plt.plot(data['RIV_log'][1]['P'], data['RIV_log'][1]['R'])
plt.plot(data['RIV_log'][2]['P'], data['RIV_log'][2]['R'])
plt.xlabel('P_electrical')
plt.ylabel('R_TES')
plt.tight_layout()
```



[16]:
```
for detector in data['RIV_log']:
    plt.plot(data['RIV_log'][detector]['P'], data['RIV_log'][detector]['R'])
plt.xlabel('P_electrical')
plt.ylabel('R_TES')
plt.tight_layout()
```

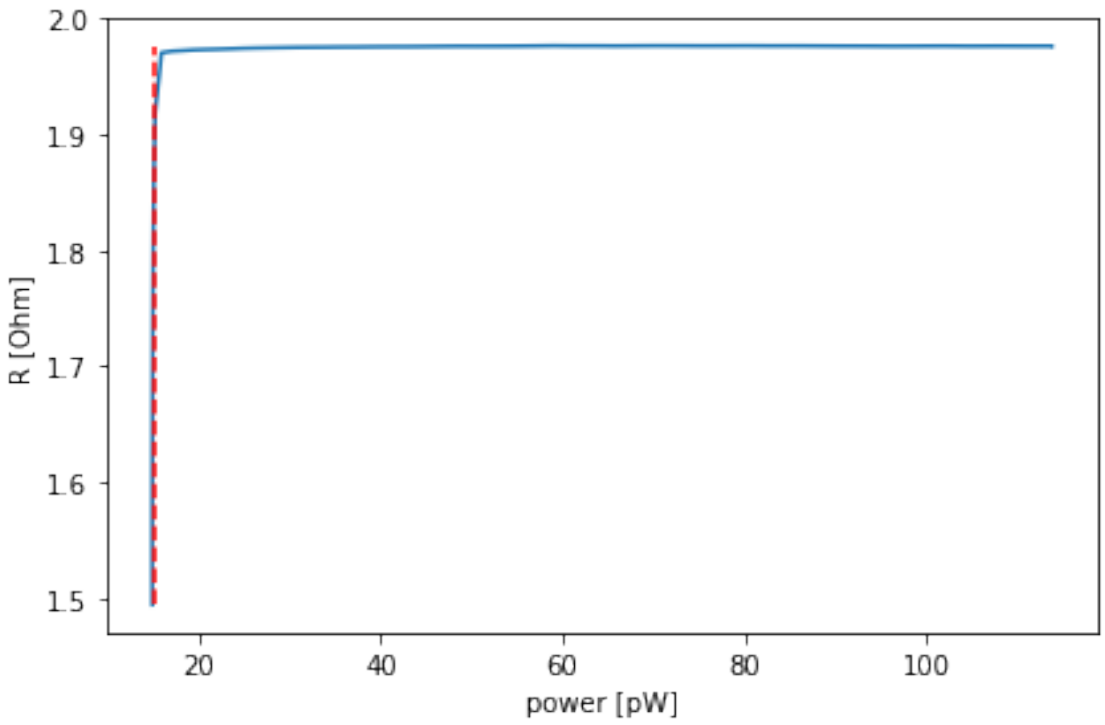


1.5 Calculate P_{sat}

The saturation power is defined as the electrical power at which the TES resistance drops below its normal value. Can you write a simple algorithm for automatically calculating P_{sat}? Do this for one detector and then loop over all of them?

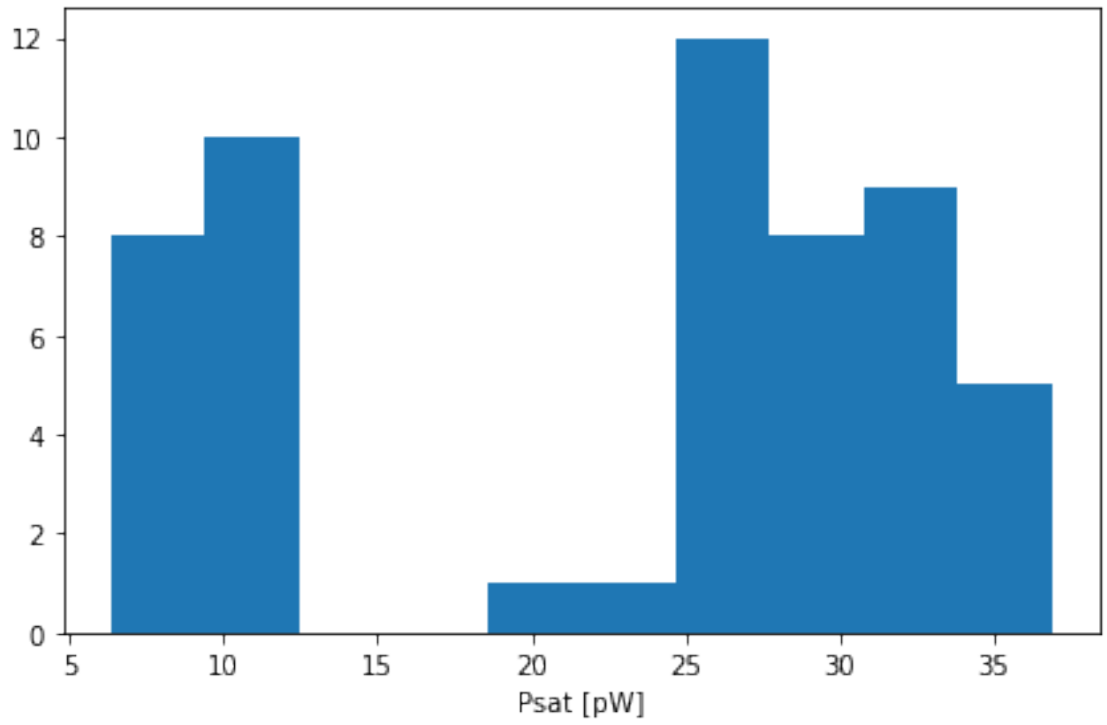
[18]:
```
detector=1
Rnormal = np.mean(data['RIV_log'][detector]['R'][0:5])
P = np.array(data['RIV_log'][detector]['P'])
R = np.array(data['RIV_log'][detector]['R'])
Psat = np.min(P[R>(0.95*Rnormal)])
```

[19]:
```
plt.plot(P*1e12, R)
plt.plot([Psat*1e12, Psat*1e12], [np.min(R), np.max(R)], 'r--')
plt.xlabel('power [pW]')
plt.ylabel('R [Ohm]')
plt.tight_layout()
```



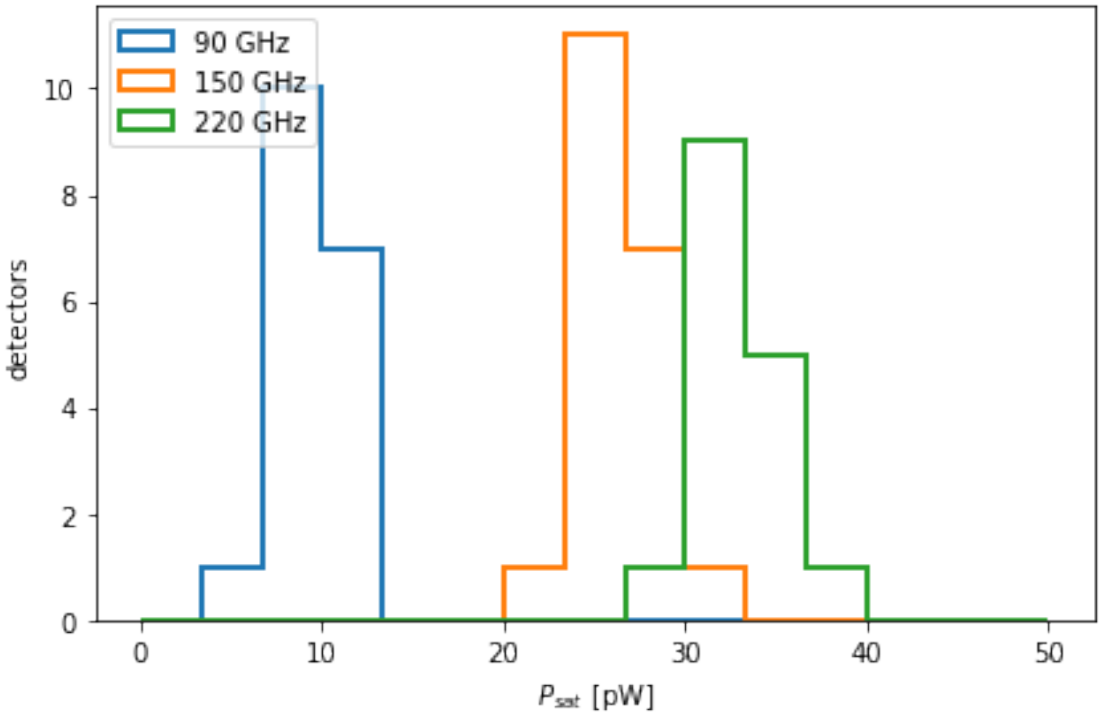
[20]:
```
Psats = {}
for detector in data['RIV_log']:
    Rnormal = np.mean(data['RIV_log'][detector]['R'][0:5])
    P = np.array(data['RIV_log'][detector]['P'])
    R = np.array(data['RIV_log'][detector]['R'])
    Psat = np.min(P[R>(0.95*Rnormal)]) / 2
    Psats[detector] = Psat
```

[21]:
```
_ = plt.hist(np.array(list(Psats.values()))*1e12)
plt.xlabel('Psat [pW]')
plt.tight_layout()
```



In FDM, each multiplexing module can have detectors with very different saturation powers. SPT-3G used trichroic detectors with 95, 150, and 220 GHz detectors in every pixel, and detectors from all three bands in each multiplexing module. Can you tell which detectors belong to which band based on the saturation powers? Why?

[23]:
```
for band in [90, 150, 220]:
    Psats_inband = [Psats[det]*1e12 for det in Psats
                    if data['subtargets'][det]['observing_band'] == band]
    _ = plt.hist(Psats_inband,
                 histtype='step',
                 linewidth=2,
                 bins=np.linspace(0,50,16),
                 label='{} GHz'.format(band))
plt.legend(loc='upper left')
plt.xlabel('$P_{sat}$ [pW]')
plt.ylabel('detectors')
plt.tight_layout()
```



1.6 Fit a thermal model to TES data

The saturation power can be described by the following equation:

$$P_{\text{sat}} = k(T_c^n - T_{\text{bath}}^n),$$

where k is a constant, T_c is the TES critical temperature, T_{bath} is the temperature of the thermal bath (the cryostat stage where the detectors are mounted), and n is a constant that is usually about 3. Perhaps obviously, T_{bath} is the only quantity that can actually be varied in this equation, so by measuring P_{sat} across a range of bath temperatures, we can fit the model and extract values for k, T_c, and n.

Why do we even care about estimating these TES parameters? There are many reasons related to detector optimization, but I will give two illustrative examples: * P_{sat} engineering - If the saturation power of a detector were to be too low in a lab test, one would naturally want to know why. Two possibilities are that the T_c is below the target, or that k below the target. If T_c were below the target, one would want to adjust the TES composition to raise the critical temperature. On the other hand, k is related to the conductivity through the suspended legs of the bolometer, so if k were below its target, one would want to adjust the geometry of the bolometer. * Phonon noise - The phonon noise of a TES detector is given by

$$\text{NEP}_{\text{ph}} = \sqrt{4k_B T^2 G},$$

where G is the thermal conductivity between the TES and the thermal bath (i.e. the conductivity of the bolometer legs). We can compute this by taking the derivative of P_{sat} with respect to T_c

$$G = knT_c^{n-1}.$$

Thus, another use of the TES model is that it lets us estimate the phonon NEP of the detector, which can be compared against noise measurements of the device.

Enough theory for now. Let's load up some archival data and try to fit these parameters.

[35]:
```
# load the data
data_path = ''
fname = 'gt_example_data.pkl'
with open(os.path.join(data_path, fname), 'rb') as f:
    gt_data = pickle.load(f)
```

[37]:
```
gt_data['w206/226.90.x']
```

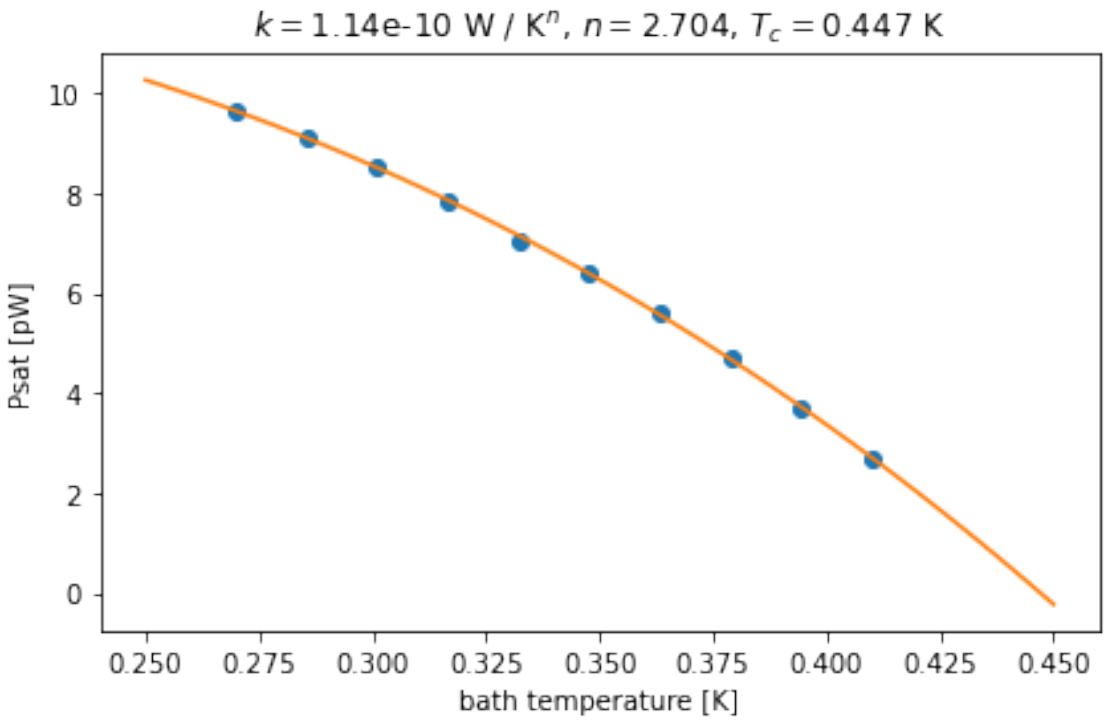
[37]:
```
{'k_IV': 1.1430586554776375e-10,
 'Psat280_IV': 9.307472282290058e-12,
 'n_IV': 2.7035336687177915,
 'k_err_IV': 8.474839532225526e-12,
 'n_err_IV': 0.10692089374933386,
 'Tc_err_IV': 0.0011887242654429549,
 'G_IV': 7.84154216298936e-11,
 'Tc_IV': 0.4470689377583631,
 'PsatVtemp': {'Psat': array([9.66710749e-12, 8.52371893e-12, 7.85528775e-12,
7.07074415e-12,
        3.69017479e-12, 9.10250286e-12, 2.69363905e-12, 6.41759169e-12,
        4.71776303e-12, 5.60301149e-12]),
  'T': array([0.26997 , 0.30111 , 0.316705, 0.33226 , 0.394485, 0.285565,
        0.41006 , 0.34779 , 0.37888 , 0.363375])}}
```

[45]:
```
# define model to fit
def psat_model(Tbath, k, n, Tc):
    return k * (Tc**n - Tbath**n)
```

[46]:
```
popt, _ = curve_fit(psat_model, temp, Psat, [1e-10, 3, 0.5])
```

[80]:
```
Psat = gt_data['w206/226.90.x']['PsatVtemp']['Psat']
temp = gt_data['w206/226.90.x']['PsatVtemp']['T']
plt.plot(temp, Psat*1e12, 'o')

temp_plot = np.linspace(0.25, 0.45)
plt.plot(temp_plot, 1e12*psat_model(temp_plot, *popt))
plt.xlabel('bath temperature [K]')
plt.ylabel('Psat [pW]')
plt.title('$k = ${:.2e} W / K$^n$, $n = ${:.3f}, $T_c = ${:.3f} K'.
 ↪format(*popt))
plt.tight_layout()
```



[47]:
```
popt
```

[47]:
```
array([1.14305871e-10, 2.70353374e+00, 4.47068937e-01])
```

[ ]:
```

```