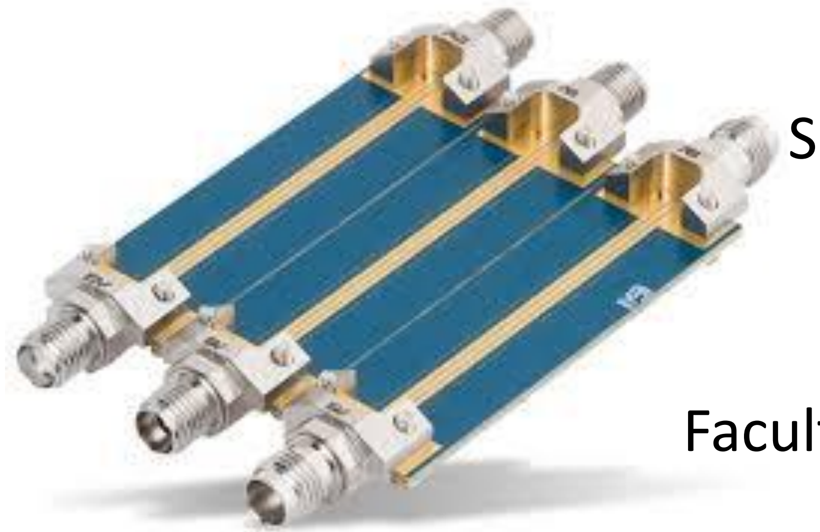
De-embedding frequency response of SMA connectors on printed circuit boards using TRL method



Presenter: Milad Seyedi

Supervisor: Professor Nasser Masoumi

17 Mordad 1401

Faculty of Electrical and Computer Engineering

University of Tehran

outline

- ❑Scikit-rf
- ❑ references

Scikit-rf

Network
The central object in skrf is a N-port microwave Network object.

```
ring_slot = rf.Network('data/ring slot.s2p')
```

A short description of the network will be printed out if entered onto the command line

```
ring_slot
2-Port Network: 'ring slot',  75.0-110.0 GHz, 201 pts, z0=[50.+0.j 50.+0.j]
```

Scikit-rf

The basic attributes of a microwave Network are provided by the following properties,

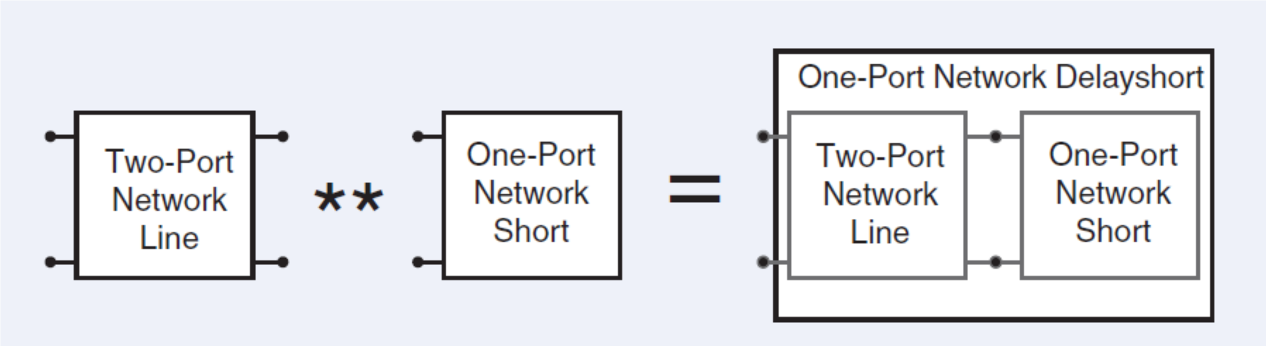
- **Network.s** : Scattering Parameter matrix.
- **Network.z0** : Port Impedance matrix.
- **Network.frequency** : Frequency Object.

Scikit-rf

Cascading and De-embedding

Cascading and de-embeding 2-port Networks can also be done though operators. Cascading is done through the power operator, **.

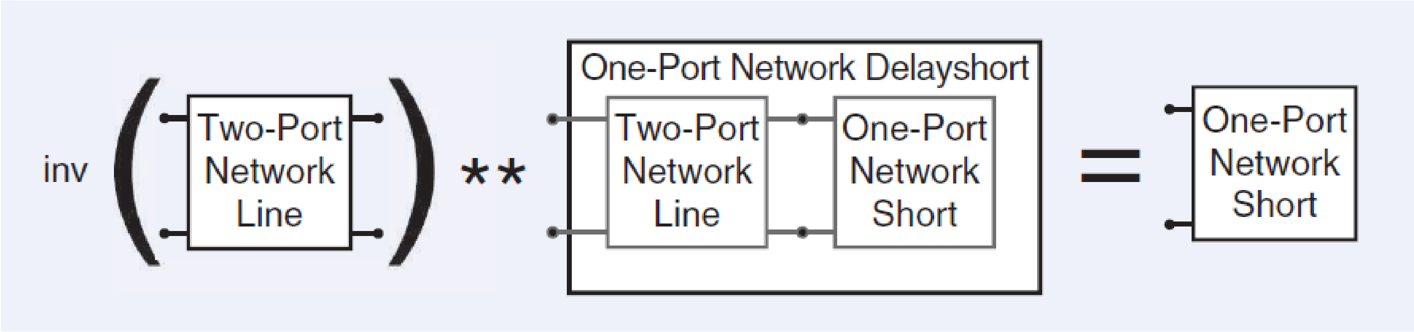
```
short = rf.data.wr2p2_short
line = rf.data.wr2p2_line

delayshort = line ** short
```



Scikit-rf

De-embedding can be accomplished by cascading the inverse of a network. The inverse of a network is accessed through the property Network.inv.

```
short = line.inv ** delayshort
```



Scikit-rf

Finding minimum (or maximum) of a Network quantity

Being a Numpy array, finding the minimum (or maximum) value of a the magnitude of the parameter:

```
import numpy as np
rs = rf.data.ring_slot  # another 2-port example

print(rs.s_mag[:,1,0].min())  # or .max() for maximum. Watch out that Python indexing starts at 0!
0.5101255034355594
```

```
f_match = rs.f[np.argmin(rs.s_mag[:,0,0])]  # frequency for min(|S11|)
print(f_match)
85850000000.0
```

Scikit-rf

Plotting Methods

Plotting functions are implemented as methods of the Network class.

Network.plot_s_re

Network.plot_s_im

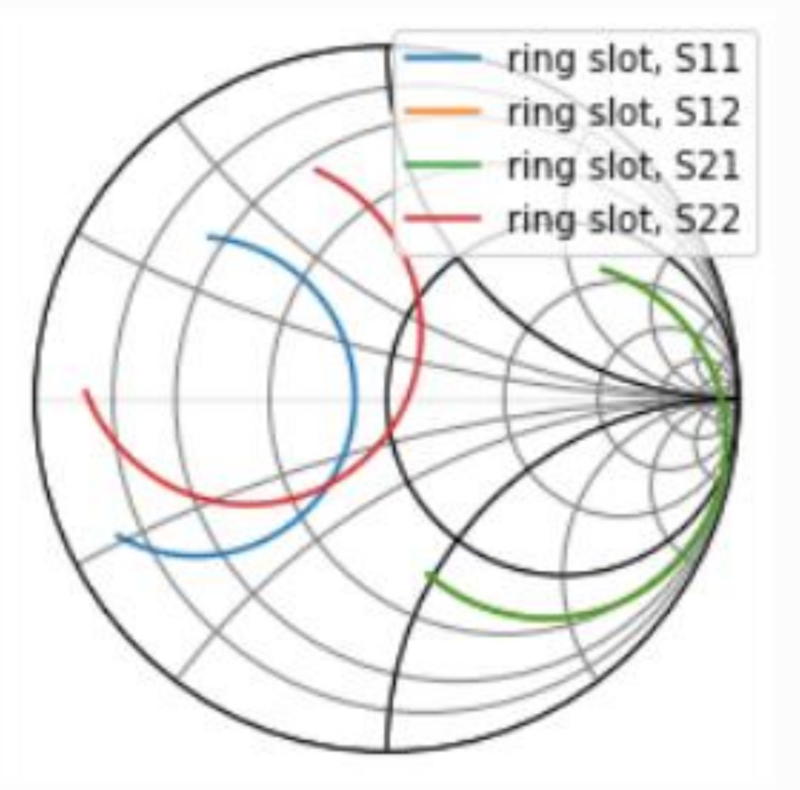
Network.plot_s_mag

Network.plot_s_db

Scikit-rf

Smith Chart

```
from skrf import Network

ring_slot = Network('data/ring slot.s2p')
ring_slot.plot_s_smith()
```



Scikit-rf

Calibration

Calibrations are performed through a Calibration class. In General, Calibration objects require two arguments:

- a list of **measured** Network's
- a list of **ideal** Network's

The Network elements in each list must all be similar (same number of ports, frequency info, etc) and must be aligned to each other.

Scikit-rf

SOLT Example

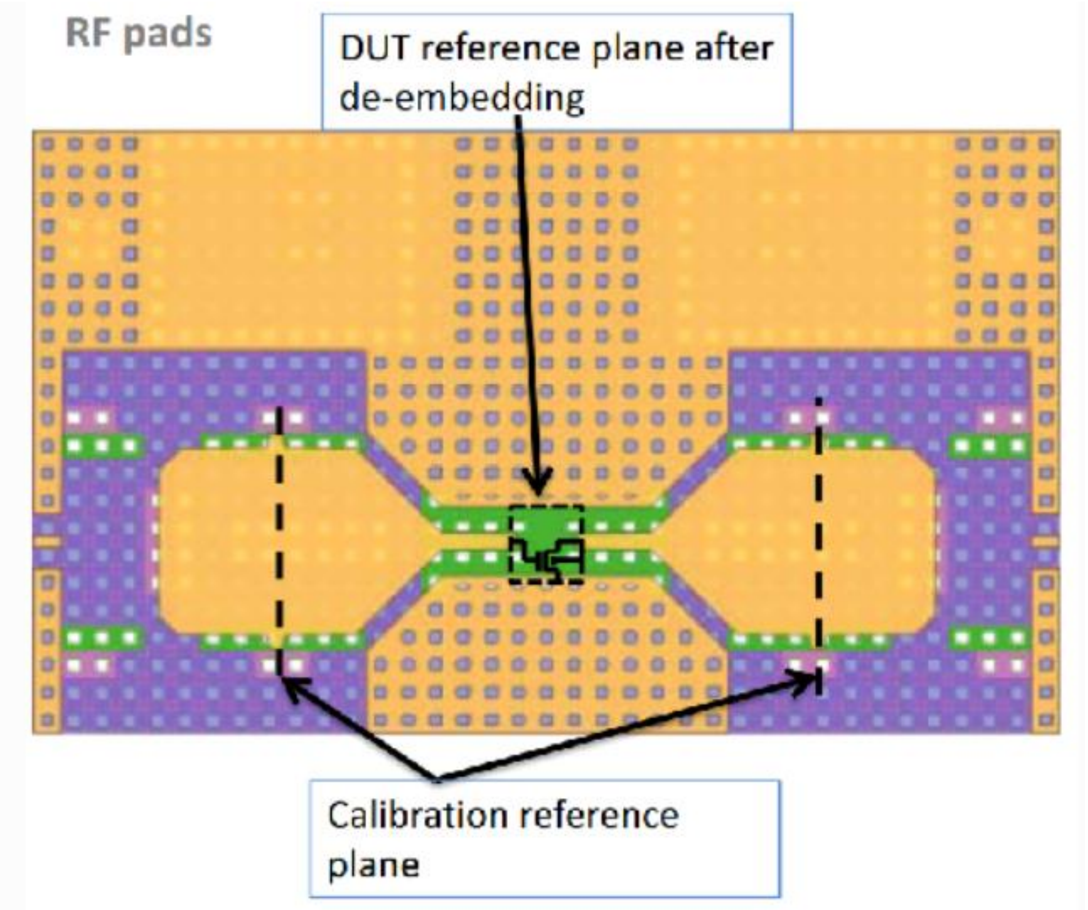
```
import skrf as rf
from skrf.calibration import SOLT

my_ideals = [
    rf.Network('ideal/short, short.s2p'),
    rf.Network('ideal/open, open.s2p'),
    rf.Network('ideal/load, load.s2p'),
    rf.Network('ideal/thru.s2p'),
    ]

# a list of Network types, holding 'measured' responses
my_measured = [
    rf.Network('measured/short, short.s2p'),
    rf.Network('measured/open, open.s2p'),
    rf.Network('measured/load, load.s2p'),
    rf.Network('measured/thru.s2p'),
    ]
```

```
## create a SOLT instance
cal = SOLT(
    ideals = my_ideals,
    measured = my_measured,
    rf.Network('measured/load, load.s2p'),
)


cal.run()

# apply it to a dut
dut = rf.Network('my_dut.s2p')
dut_caled = cal.apply_cal(dut)


dut_caled.plot_s_db()
dut_caled.write_touchstone()
```

Scikit-rf

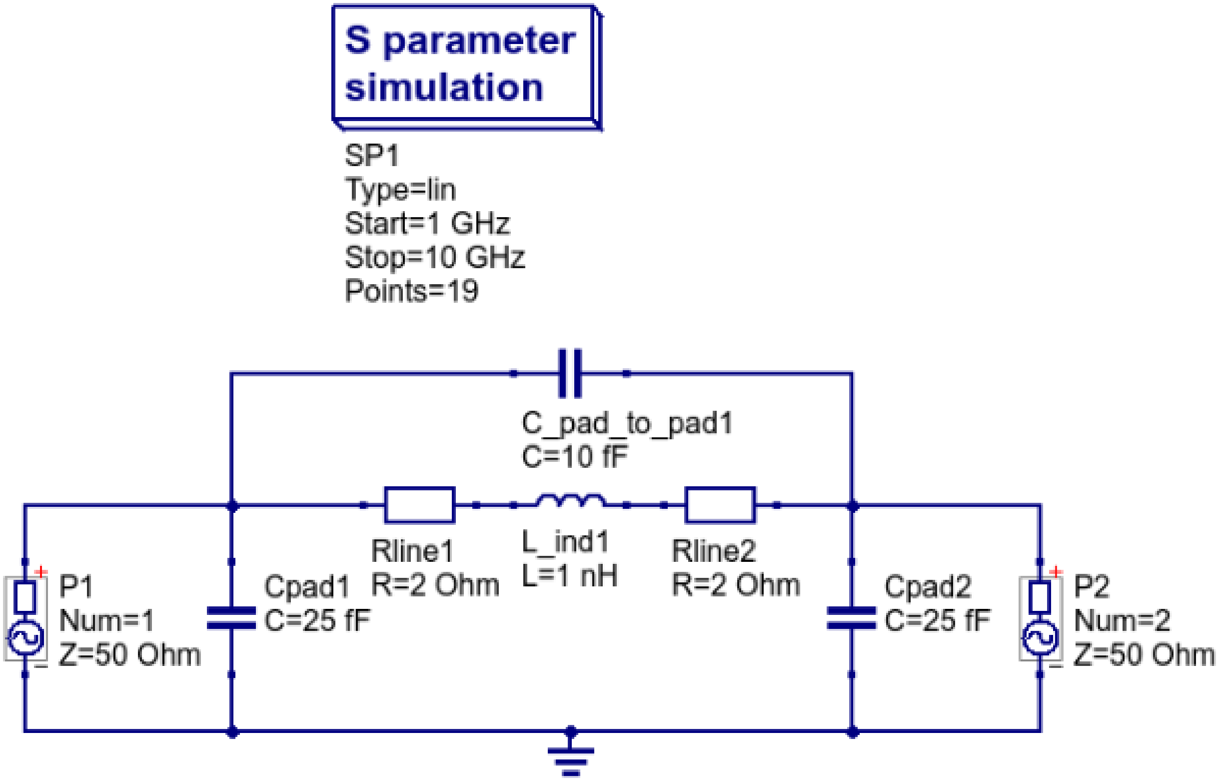
De-embedding

De-embedding refers to the removal of extraneous effects that a test fixture can have on the measurement of a device under test (DUT).



Scikit-rf

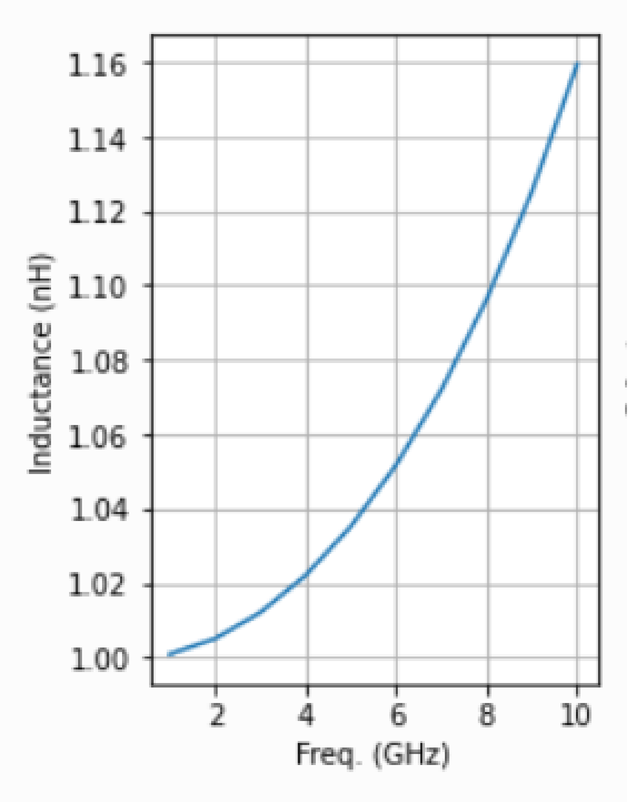
De-embedding with Scikit-RF



Scikit-rf

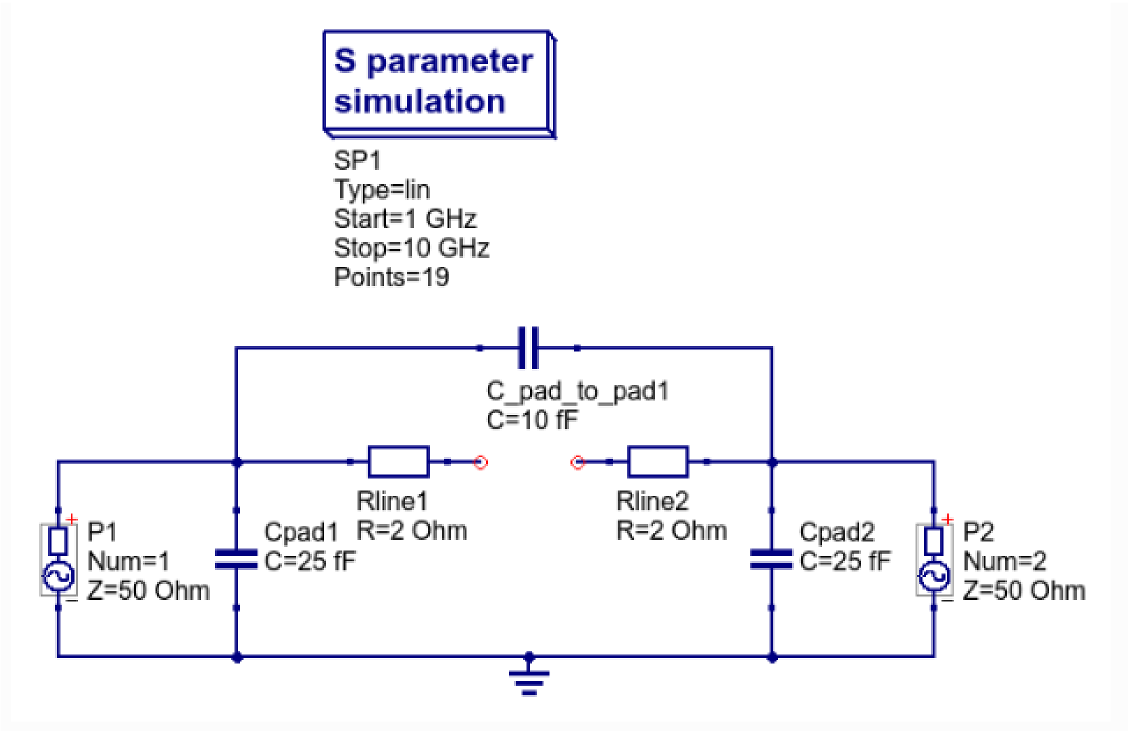
De-embedding with Scikit-RF

```
import skrf as rf
import numpy as np
import matplotlib.pyplot as plt

raw_ind = rf.Network('data/ind.s2p')

Lraw_nH = 1e9 * np.imag(1/raw_ind.y[:,0,0])/2/np.pi/raw_ind.f
```



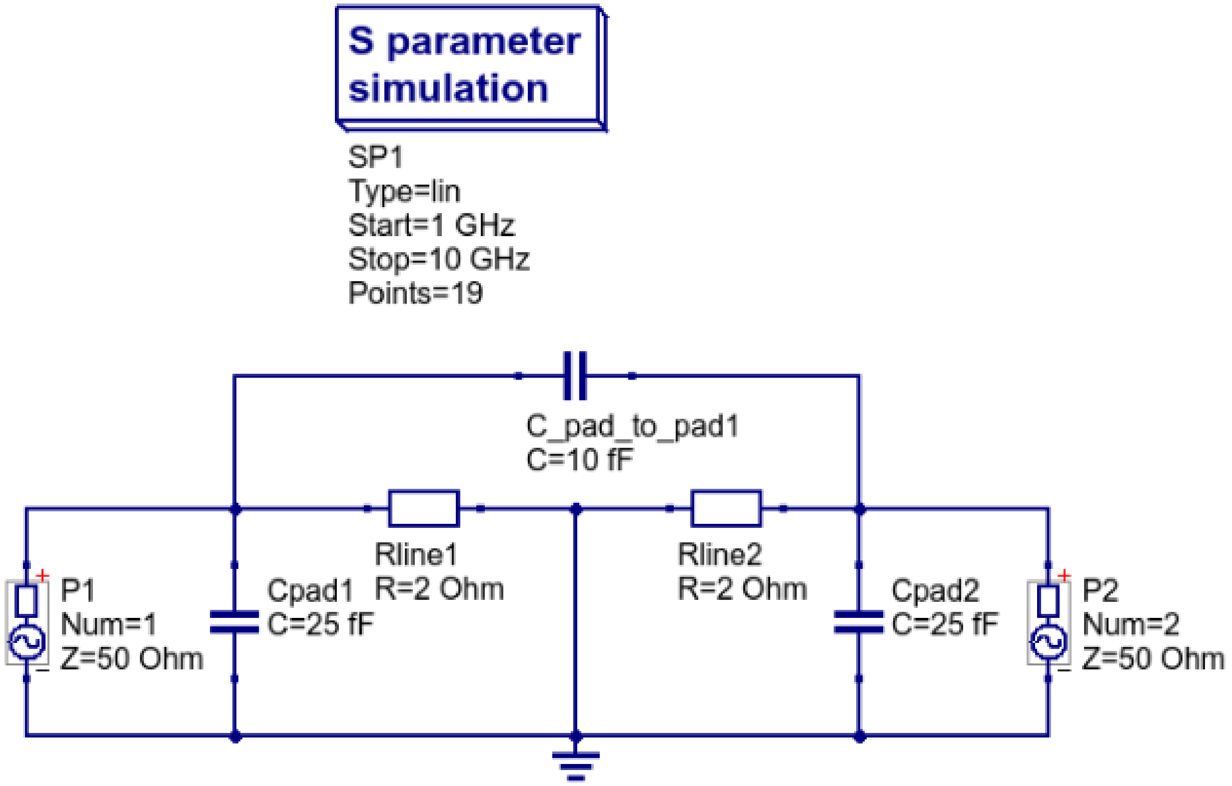
Scikit-rf

open dummy



Scikit-rf

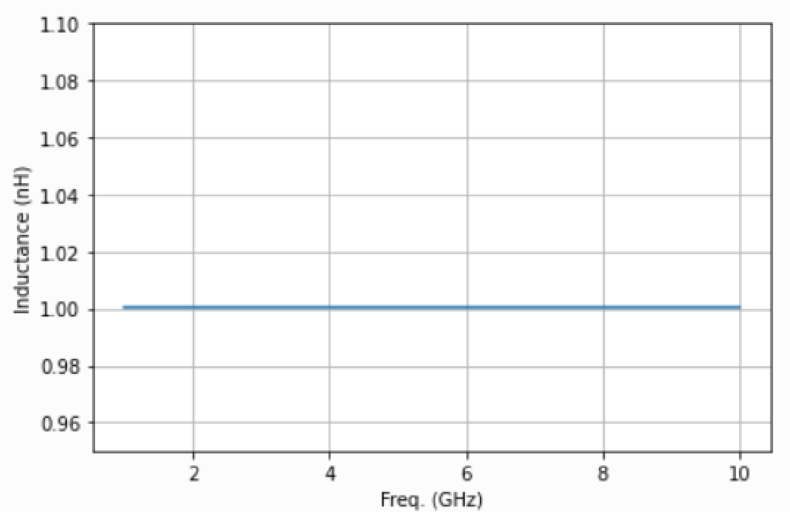
short dummy



Scikit-rf

De-embedding with Scikit-RF

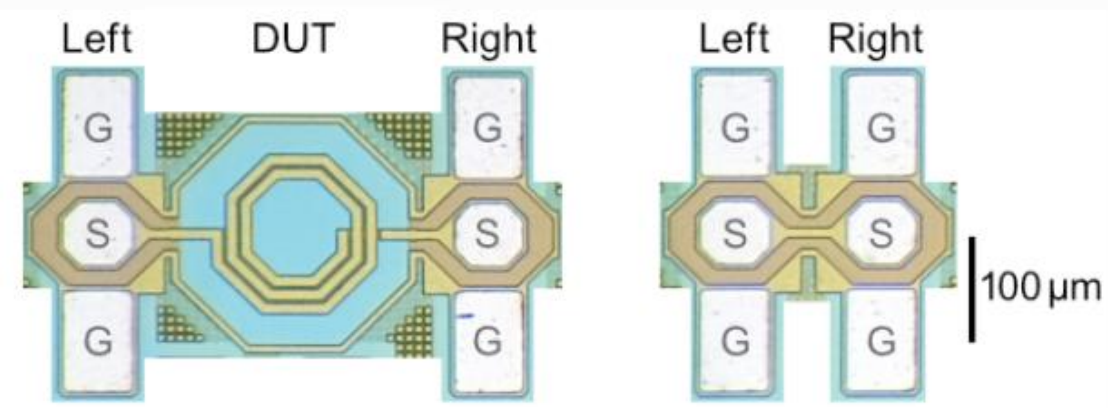
```
open_nw = rf.Network('data/open.s2p')
short_nw = rf.Network('data/short.s2p’)

from skrf.calibration import OpenShort

dm = OpenShort(dummy_open=open_nw, dummy_short=short_nw,
name='tutorial')
actual_ind = dm.deembed(raw_ind)

Lactual_nH = 1e9 * np.imag(1/actual_ind.y[:,0,0])/2/np.pi/actual_ind.f
```



Scikit-rf

Through-only de-embedding

At higher frequencies above 10 GHz, **the fringe capacitance of the open dummy and the parasitic inductance of the short dummy** cause errors. For high frequencies, several methods using a through (thru) dummy have been proposed



(a) Test device (b) Thru dummy

Scikit-rf

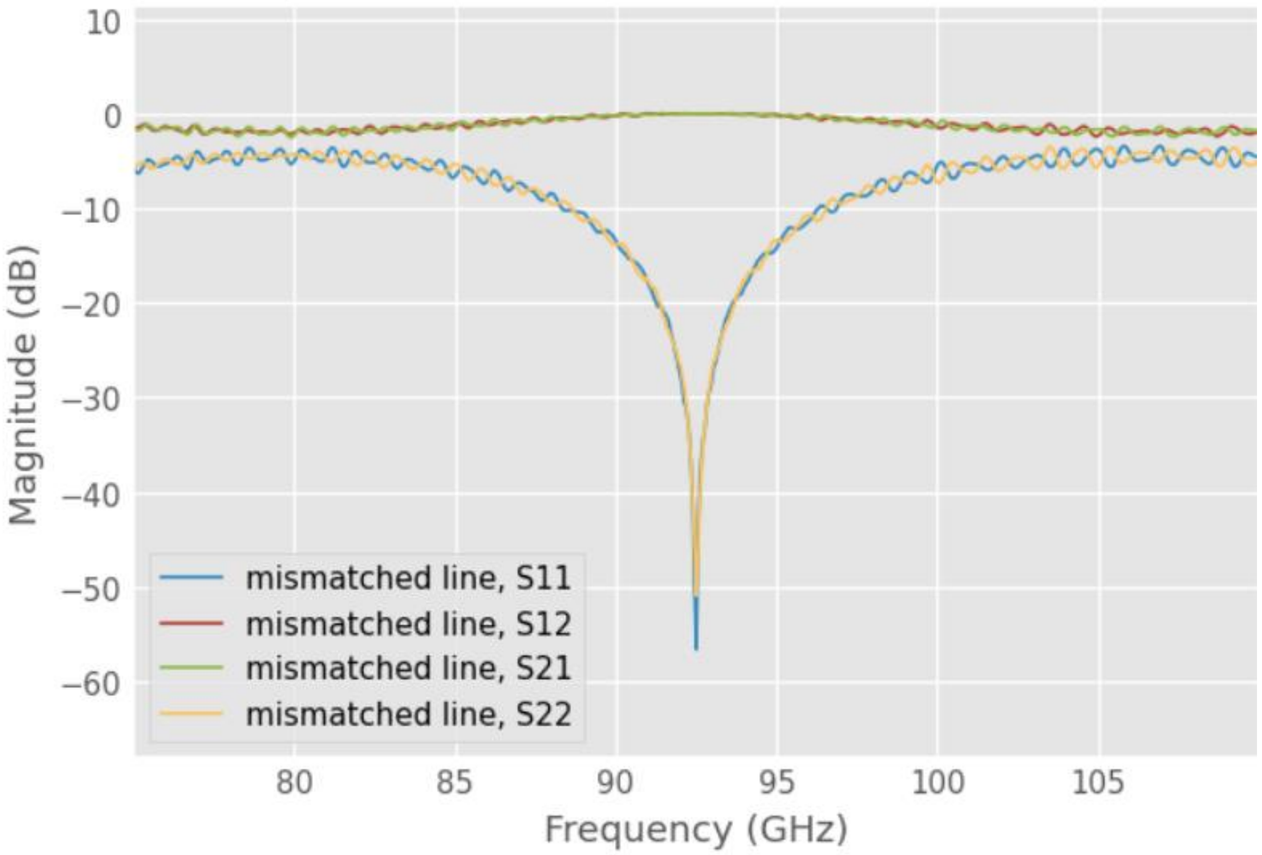
TRL

```
import skrf as rf
from skrf.calibration import TRL

T = rf.Network('trl_data/thru.s2p')
R = rf.Network('trl_data/reflect.s2p')
L = rf.Network('trl_data/line.s2p')

switch_terms = (rf.Network('trl_data/forward switch term.s1p'),
                rf.Network('trl_data/reverse switch term.s1p'))

measured = [T,R,L]
```

```
trl = TRL(measured = measured,
        switch_terms = switch_terms)

dut_raw = rf.Network('trl_data/mismatched line.s2p')
dut_corrected = trl.apply_cal(dut_raw)
dut_corrected.plot_s_db()
```

Scikit-rf

TRL



Scikit-rf

Switch Terms

The two error networks change slightly depending on which port is being excited. This is due to the internal switch within the VNA.

How do I get them ?

forward switch term == a2/b2 with source port 1

reverse switch term == a1/b1 with source port 2

references

[1] https://scikit-rf.readthedocs.io/