ECE 5224 HW1 Hopkinsgl

1a. $d = 2\,mil$, $l = 10\,mm$ @ $100°C$, $\rho = 2.2e^{-8}\ \Omega\cdot m$, and $\alpha = 3.4e^{-3}/°C$

$R_o = \frac{\rho \cdot l}{A_c}$ $\quad l = 10mm = \underline{0.01m}$ $\quad rad = \frac{2mil}{2} = \frac{0.002in}{2} = 0.001in$

$= 1mil$

$= 2.54E\text{-}5m$

$A_c = \pi r^2 = \pi(2.54E\text{-}5m)^2 = 2.0268E\text{-}9\,m^2$

$R_o = \frac{2.2E\text{-}8\,\Omega \cancel{m} \cdot 0.01\cancel{m}}{2.0268E\text{-}9\,\cancel{m^2}} = \underline{1.085E\text{-}1\,\Omega}$

$R_1 = R_o \cdot [1 + \alpha(T_1 - T_o)]$

$= 1.085E\text{-}1\Omega \cdot [1 + 3.4E\text{-}3(100°C - 25°C)]$

$= \boxed{1.36E\text{-}1\,\Omega}$

1b. When temperatures increase, atoms start to move at a more rapid rate which causes the current to have difficult time flowing aka resistance. This is usually just seen as thermal resistance (as seen in the slides).

2a. $f = 500MHz$ $\quad R_{AC} = \frac{\rho \cdot l}{A_{eff}}$, $A_{eff} = \pi(d\delta - \delta^2)$, $\delta = \sqrt{(\rho/(\pi f \mu)}$

$\delta = \sqrt{\rho/\pi f \mu} = \sqrt{2.2E\text{-}8\,\Omega m / (\pi(500MHz)(4\pi E\text{-}7 H/m))}$

$= \underline{3.34E\text{-}6m} \ll 25\mu m$ ✓

$A_{eff} = \pi(d\delta - \delta^2)$

$= \pi((50E\text{-}6m)(3.34E\text{-}6m) - (3.34E\text{-}6m)^2) = \underline{4.896E\text{-}10m^2}$

$R_{AC} = \frac{\rho \cdot l}{A_{eff}} = \frac{2.2E\text{-}8\,\Omega \cancel{m} \cdot 0.01\cancel{m}}{4.896E\text{-}10\,\cancel{m^2}} = \boxed{4.49E\text{-}1\,\Omega}$

2b. According to lecture 3, slid 20-21, this phenomena is known as the skin effect. Due to the high frequencies opposing electric field is made causing electrons to be forced to the surface. However eddy currents cancels the new central flow and causes the flow to be reinforced in the "skin".

2c. You could choose a conductor with a higher resistivity. Or you could maybe change the conductor from a circular design to a rectangular one for more perimeter in the cross sectional area.

3. $f = 500\,MHz$, $t = 1\,mil = 2.54E{-5}\,m$, $w = 10\,mils = 2.54E{-4}\,m$, and $l = 10\,mm = .01\,m$

$\delta = \sqrt{\rho/(\pi f u)} = \sqrt{2.2E{-8}\,\Omega m/(\pi(500E{-6})(4\pi E{-7}\,H/m))}$
$= 3.34E{-6}\,m$

$A_{c} = t \cdot w = (2.54E{-5}\,m)(2.54E{-4}) = \underline{6.45E{-9}\,m^2}$

$R_{25c} = \dfrac{\rho_{25c} \cdot l}{A_c} = \dfrac{2.2E{-8}\,\Omega \cancel{m} \cdot .01\cancel{m}}{6.45E{-9}\,\cancel{m^2}} = \boxed{3.41E{-2}\,\Omega}^{DC}$

for A_C: $2\delta = 2 \cdot 3.34E{-6} = 6.68E{-6}\,m \ll w$ and t

$\Rightarrow A_{eff} = wt - (w - 2\delta)(t - 2\delta)$
$= 6.45E{-9}\,m^2 - (2.54E{-4}\,m - 6.68E{-6}\,m)(2.54E{-5} - 6.68E{-6}\,m)$
$= 6.45E{-9}\,m^2 - (2.473E{-4}\,m)(1.872E{-5}\,m)$
$= 1.82E{-9}\,m^2$

$R_{AC} = \dfrac{\rho \cdot l}{A_{eff}} = \dfrac{2.2E{-8}\,\Omega \cancel{m} \cdot .01\cancel{m}}{1.82E{-9}\,\cancel{m^2}} = \boxed{1.21E{-1}\,\Omega}^{AC}$

HW1_Prob_4

February 10, 2025

1 Problem 4

1.1 Self Inductance Formula

$$L_{\text{self}} = \frac{\mu}{2\pi} l \left[\ln\left(\frac{l}{r} + \sqrt{1 + \frac{l^2}{r^2}} \right) - \sqrt{1 + \frac{r^2}{l^2}} + \frac{r}{l} + \frac{1}{4} \right]$$

- l = length in meters
- r = radius in meters
- $\mu = 4(\pi)$ x 10E-7 H/m

Necessary packages

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Making the formula into a function:

[34]:
```
def selfInduct(r: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    # The actual formula
    lSelf = 0

    if (l > r):
        # Convert from mm to cm
        r * 1e-1
        l * 1e-1

        lSelf = (.002 * l) * (np.log((2 * l) / r) - (3 / 4)) # returns uH/cm
        lSelf *= 100 # puts in back in nH/mm
    else:
        # Convert from mm to m
        r *= 1e-3
        l *= 1e-3

        outside = ((mu / (2 * np.pi)) * l)
        inside1 = np.log((l / r) + np.sqrt(1 + ((l ** 2) / (r ** 2))))
        inside2 = np.sqrt(1 + ((r ** 2) / (l ** 2)))
```

```
        lSelf = outside * (inside1 - inside2 + (r / l) + (1 / 4))
        # returns in henries / m, so make it into nano henries / mm
        lSelf *= 1e6

    return lSelf # Returns H/m
```

Making a range of lengths:

[35]:
```
# Arranging an array from 2mm to 20mm (.002m to .02m)
lengths = np.linspace(2, 20, 15) # This is in mm

print(lengths)
```

```
[ 2.          3.28571429  4.57142857  5.85714286  7.14285714  8.42857143
  9.71428571 11.         12.28571429 13.57142857 14.85714286 16.14285714
 17.42857143 18.71428571 20.        ]
```

Calculate the inductances:

[36]:
```
# 5 mil diamter = 0.000127m
diam = 0.000127 * 1e3 # put it in mm
r = diam / 2

# Generate a list of inductances to plot
lSelfs = []
for len in lengths:
    induct = selfInduct(r, len)
    lSelfs.append(induct) # These are in Henries

print("Self inductances in nH/mm:\n", lSelfs)
```

```
Self inductances in nH/mm:
 [1.3572038936813529, 2.5559220649113503, 3.858000738183177, 5.233385809228277,
6.6656791571661005, 8.144511458889621, 9.662722498970505, 11.215067218171974,
12.797536407782296, 14.40696523251209, 16.04079153108193, 17.696898435227215,
19.37350745192024, 21.06910326686843, 22.78237930878971]
```

Calculate the inductances using the rule of thumb:

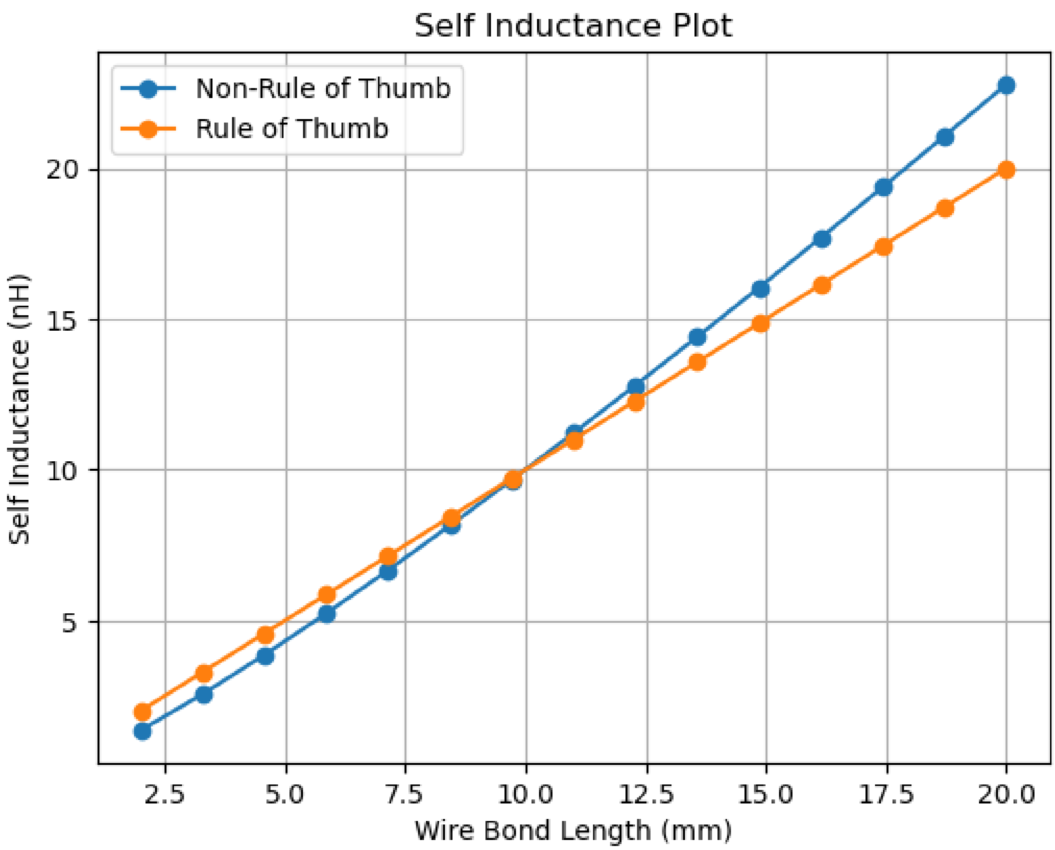
[37]:
```
# Rule of thumb is ~1 nH/mm (1uH/m)
lSelfs_rot = (lengths) * 1 # put them in nH/mm
print(lSelfs_rot) # this is in nH/mm|
```

```
[ 2.          3.28571429  4.57142857  5.85714286  7.14285714  8.42857143
  9.71428571 11.         12.28571429 13.57142857 14.85714286 16.14285714
 17.42857143 18.71428571 20.        ]
```

Plot the values:

[38]:
```
fig, ax = plt.subplots()

ax.plot(lengths, lSelfs, marker='o', linestyle='-', label='Non-Rule of Thumb')
ax.plot(lengths, lSelfs_rot, marker='o', linestyle='-', label='Rule of Thumb')

ax.set(xlabel='Wire Bond Length (mm)', ylabel='Self Inductance (nH)',
       title="Self Inductance Plot")
ax.grid()
ax.legend()
```

[38]:
```
<matplotlib.legend.Legend at 0x1e6c876b1c0>
```



It seems to the stay close together until around 10mm. That's when then non-rule of thumb method over takes the other.

HW1_Prob_5

February 10, 2025

1 Problem 5

1.1 Self Inductance Formula

$$L_{\text{self}} = \frac{\mu}{2\pi} l \left[\ln \left(\frac{l}{r} + \sqrt{1 + \frac{l^2}{r^2}} \right) - \sqrt{1 + \frac{r^2}{l^2}} + \frac{r}{l} + \frac{1}{4} \right]$$

- l = length in meters
- r = radius in meters
- μ = 4(π) x 10E-7 H/m

Necessary packages

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Making the formula into a function:

[36]:
```
def selfInduct(r: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    # The actual formula
    lSelf = 0

    if (l > r):
        # Convert from mm to cm
        r * 1e-1
        l * 1e-1

        lSelf = (.002 * l) * (np.log((2 * l) / r) - (3 / 4)) # returns uH/cm
        lSelf *= 100 # puts in back in nH/mm
    else:
        # Convert from mm to m
        r *= 1e-3
        l *= 1e-3

        outside = ((mu / (2 * np.pi)) * l)
        inside1 = np.log((l / r) + np.sqrt(1 + ((l ** 2) / (r ** 2))))
        inside2 = np.sqrt(1 + ((r ** 2) / (l ** 2)))
```

```
        lSelf = outside * (inside1 - inside2 + (r / l) + (1 / 4))
        # returns in henries / m, so make it into nano henries / mm
        lSelf *= 1e6

    return lSelf # Returns H/m
```

Make a range of radii:

[45]:
```
# Arranging an array from .5 mils to 5 mils (0.0000127m to 0.000127m)
radii = np.linspace(0.5, 5, 15) # This is in mils
# Convert to m then to mm
radii *= (2.54e-5)
radii *= (1e3)

print(radii)
```

```
[0.0127     0.02086429 0.02902857 0.03719286 0.04535714 0.05352143
 0.06168571 0.06985    0.07801429 0.08617857 0.09434286 0.10250714
 0.11067143 0.11883571 0.127     ]
```

Calculate the inductances:

[46]:
```
len = 10 # This is in mm

# Generate a list of inductances to plot
lSelfs = []

for rad in radii:
    induct = selfInduct(rad, len)
    lSelfs.append(induct) # These are in Henries

print("Self inductances in nH/mm:\n", lSelfs)
```

```
Self inductances in nH/mm:
 [13.223771118143162, 12.230897345515382, 11.57041397177423, 11.074741643965066,
10.67783976651739, 10.346810889562244, 10.062870367021471, 9.814274933666315,
9.593191184866667, 9.3941319941726, 9.213103979090937, 9.047110139949,
8.893843687907168, 8.75149113097138, 8.618600932155074]
```

Calculate the inductances using the rule of thumb:

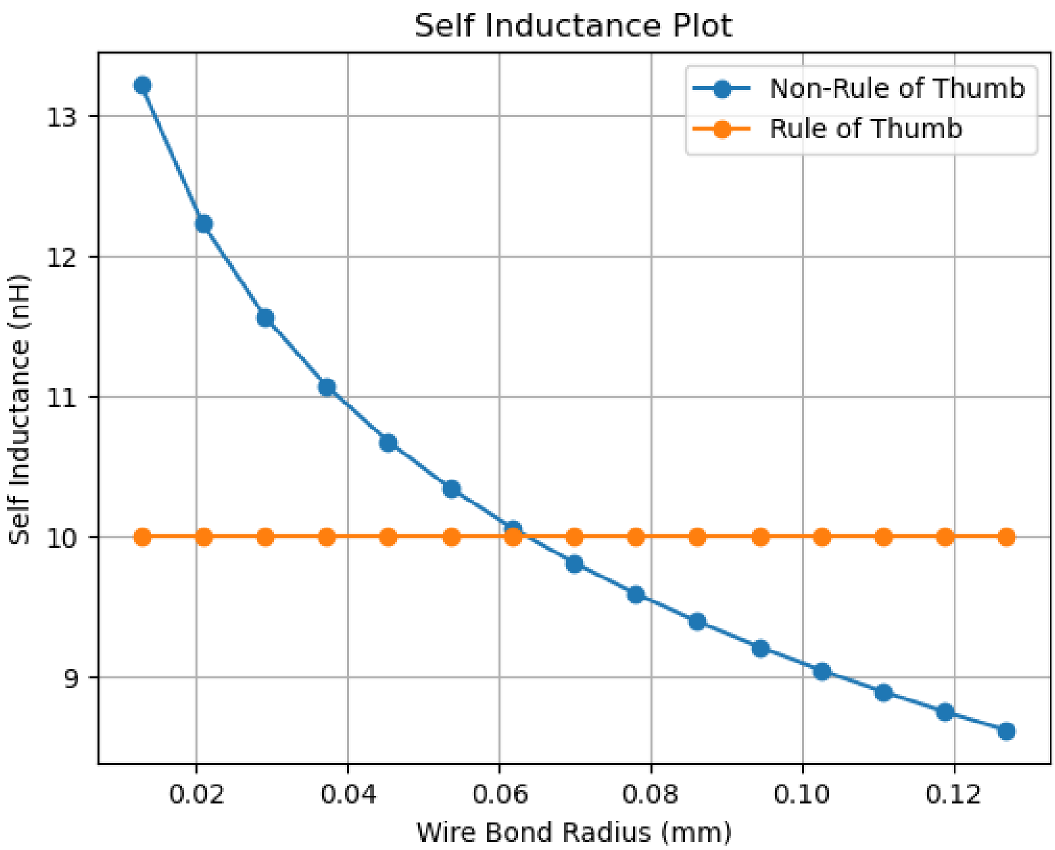
[47]:
```
# Rule of thumb is ~1 nH/mm (1uH/m)
lSelfs_rot = np.full(15, 10) # Length doesn't change so its always the same.
print(lSelfs_rot)
```

```
[10 10 10 10 10 10 10 10 10 10 10 10 10 10 10]
```

Plot the values:

[48]:
```
fig, ax = plt.subplots()

ax.plot(radii, lSelfs, marker='o', linestyle='-', label='Non-Rule of Thumb')
ax.plot(radii, lSelfs_rot, marker='o', linestyle='-', label='Rule of Thumb')

ax.set(xlabel='Wire Bond Radius (mm)', ylabel='Self Inductance (nH)',
       title="Self Inductance Plot")
ax.grid()
ax.legend()
```

[48]: <matplotlib.legend.Legend at 0x25a65aaabf0>



The rule of thumb line never changes due to the constant length unlike the non-rule of thumb line with its changing radius. In contrast to the previous problem, the line is a downward curve instead of a slightly curved and upward line b/c of modification in having a varying length vs a varying radius.

HW1_Prob_6

February 10, 2025

1 Problem 6

Necessary packages

[23]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

Make the formula for self mutual inductance:

[19]:
```python
def selfInduct(r: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    # The actual formula
    lSelf = 0.0

    if (l > r):
        # Convert from mm to cm
        r * 1e-1
        l * 1e-1

        lSelf = (.002 * l) * (np.log((2 * l) / r) - (3 / 4)) # returns uH/cm
        lSelf *= 100 # puts in back in nH/mm
    else:
        print("warning")

    return lSelf

def mutualInduct(s: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    mSelf = 0.0

    # Convert from mm to cm
    s * 1e-1
    l * 1e-1
```

```
    mSelf = (.002 * l) * (np.log((2 * l) / s) - 1) # returns uH/cm
    mSelf *= 100 # puts in back in nH/mm


    return mSelf

def totalInduct(s: float, l: float, r: float):
    totalWMut = 0.0
    totalWOMut = 0.0

    totalWMut = (2 * (selfInduct(r, l))) - (2 * mutualInduct(s, l))
    totalWOMut = (2 * (selfInduct(r, l)))

    return totalWMut, totalWOMut
```

Create an array of spacings

[20]:
```
# Arranging an array from 1mm to 10mm (.001m to .01m)
spacing = np.linspace(1, 10, 15) # This is in mm

print(spacing)
```

```
[ 1.          1.64285714  2.28571429  2.92857143  3.57142857  4.21428571
  4.85714286  5.5         6.14285714  6.78571429  7.42857143  8.07142857
  8.71428571  9.35714286 10.        ]
```

Calculate the inductances

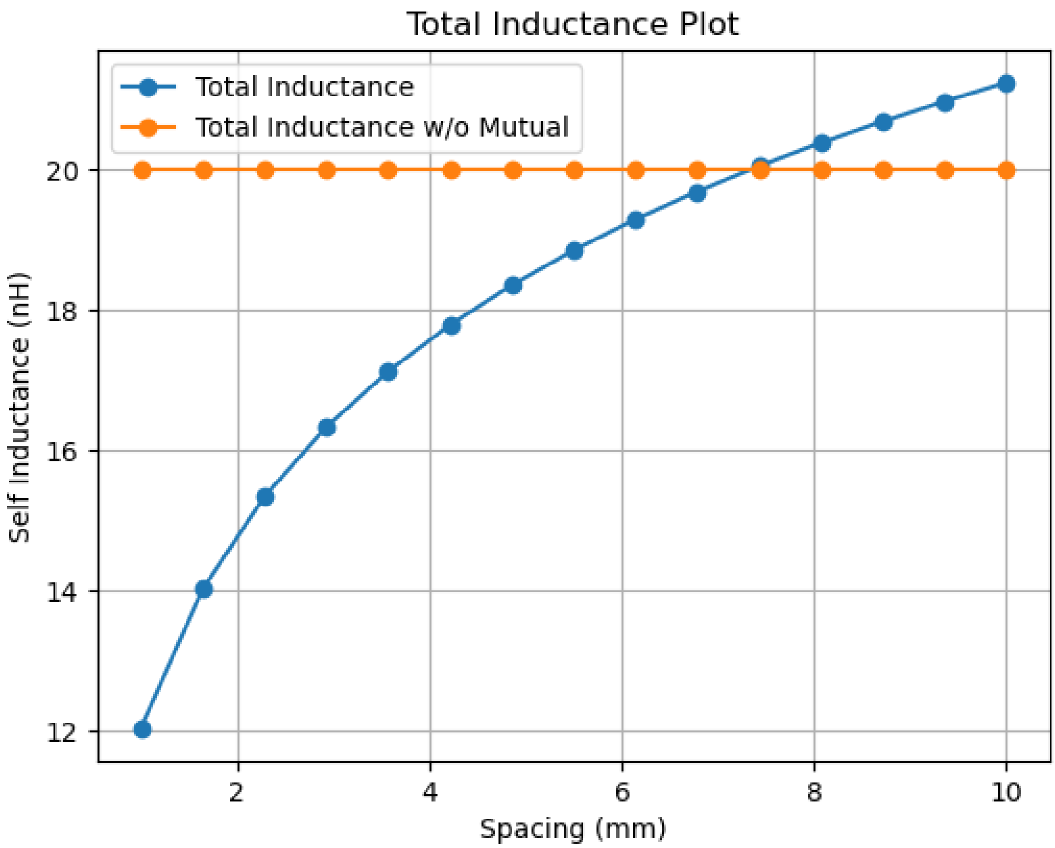
[21]:
```
totalWMuts = []
totalWOMuts = []

len = 10 # this in in mm
# diam is 5 mils (0.127 mm), so rad is 0.0635 mm
rad = 0.0635

for s in spacing:
    x1, x2 = totalInduct(s, len, rad)
    totalWMuts.append(x1)
    totalWOMuts.append(x2)
```

Plot

[22]:
```
fig, ax = plt.subplots()

ax.plot(spacing, totalWMuts, marker='o', linestyle='-', label='Total␣
 ↪Inductance')
ax.plot(spacing, totalWOMuts, marker='o', linestyle='-', label='Total␣
 ↪Inductance w/o Mutual')
```

```
ax.set(xlabel='Spacing (mm)', ylabel='Self Inductance (nH)',
       title="Total Inductance Plot")
ax.grid()
ax.legend()
```

[22]: <matplotlib.legend.Legend at 0x2ec8fb97250>



Since the inductance remains constant no matter the spacing, the line without mutal inductance barely changes which shows the amount of influence it has on the total inductance.

HW1_Prob_7

February 10, 2025

1 Problem 7

Necessary packages

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Create an array of spacings

[5]:
```
# Arranging an array from 1mm to 10mm (.001m to .01m)
spacing = np.linspace(1, 10, 15) # This is in mm

print(spacing)
```

```
[ 1.          1.64285714  2.28571429  2.92857143  3.57142857  4.21428571
  4.85714286  5.5         6.14285714  6.78571429  7.42857143  8.07142857
  8.71428571  9.35714286 10.        ]
```

[6]:
```
len = 10 # This is in mm

# diam is 2 mils (0.0508 mm), so rad is 0.0254 mm
rad = 0.0254

width = 20 # This is in mm
```

[13]:
```
def selfInduct(r: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    # The actual formula
    lSelf = 0.0

    if (l > r):
        # Convert from mm to cm
        r * 1e-1
        l * 1e-1

        lSelf = (.002 * l) * (np.log((2 * l) / r) - (3 / 4)) # returns uH/cm
        lSelf *= 100 # puts in back in nH/mm
```

```
    else:
        print("warning")

    return lSelf

def mutualInduct(s: float, l: float) -> float:
    # Constant for mu
    mu = (4 * np.pi) * (10 ** -7)

    mSelf = 0.0

    # Convert from mm to cm
    s * 1e-1
    l * 1e-1

    mSelf = (.002 * l) * (np.log((2 * l) / s) - 1) # returns uH/cm
    mSelf *= 100 # puts in back in nH/mm


    return mSelf

def totalInduct(s: float, l: float, r: float):
    # Assume we are getting mm
    totalWMut = 0.0

    totalWMut = (2 * (selfInduct(r, l))) - (2 * mutualInduct(s, l))

    return totalWMut

def parPlaneInduct(s: float, l: float):
    # Convert mm to m
    s *= 1e-3
    l *= 1e-3

    mu = (4 * np.pi) * (10 ** -7)

    leff = (mu * s * l) / l # this is in nH

    return leff

def groundPlaneInduct(l: float, s: float, d: float):
    # Convert mm to m
    s *= 1e-3
    l *= 1e-3
    d *= 1e-3

    mu = (4 * np.pi) * (10 ** -7)
```

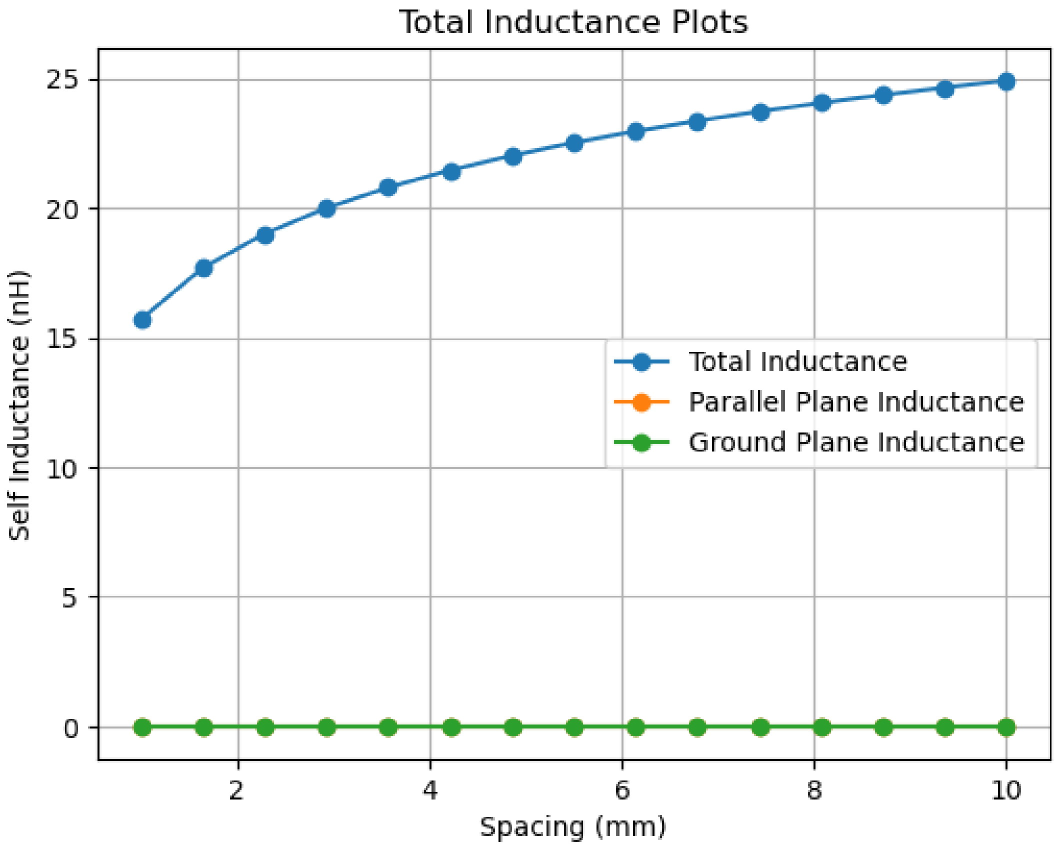
```
    leff = ((mu * l) / (2 * np.pi)) * np.arccosh((2 * s) / d)

    return leff # this is in nH
```

[14]:
```
totalInducts = []
parPlaninducts = []
groundPlaneInducts = []

for s in spacing:
    totalInducts.append(totalInduct(s, len, rad))
    parPlaninducts.append(parPlaneInduct(s, len))
    groundPlaneInducts.append(groundPlaneInduct(len, s, rad*2))

print(parPlaninducts)
```

```
[1.2566370614359174e-09, 2.0644751723590067e-09, 2.8723132832820964e-09,
3.6801513942051866e-09, 4.487989505128277e-09, 5.2958276160513665e-09,
6.1036657269744555e-09, 6.911503837897544e-09, 7.719341948820636e-09,
8.527180059743726e-09, 9.335018170666815e-09, 1.0142856281589907e-08,
1.0950694392512996e-08, 1.1758532503436083e-08, 1.2566370614359171e-08]
```

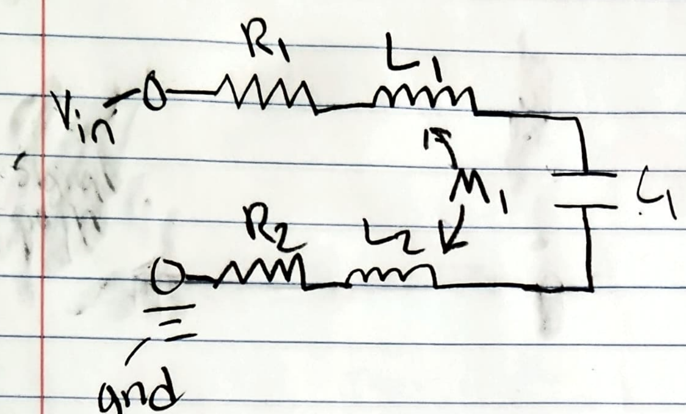
[15]:
```
fig, ax = plt.subplots()

ax.plot(spacing, totalInducts, marker='o', linestyle='-', label='Total␣
↪Inductance')
ax.plot(spacing, parPlaninducts, marker='o', linestyle='-', label='Parallel␣
↪Plane Inductance')
ax.plot(spacing, groundPlaneInducts, marker='o', linestyle='-', label='Ground␣
↪Plane Inductance')

ax.set(xlabel='Spacing (mm)', ylabel='Self Inductance (nH)',
       title="Total Inductance Plots")
ax.grid()
ax.legend()
```

[15]:
```
<matplotlib.legend.Legend at 0x2262eebad70>
```



Ground plane has the lowest inductance due to its large surface area

8. a.



capacitive load of the die

b. Totat inductance $= L_1 + L_2 - 2M_1$

formulas_used

February 10, 2025

1 Other Formulas Used (More or Less)

1.0.1 Parallel Inductance Formula

$$L_{\text{parallel}} = \frac{L_p + M}{2}$$

1.0.2 Self-Inductance Formula

$$L_{\text{self}} = 0.002l \left[\ln\left(\frac{2l}{r}\right) - \frac{3}{4} \right]$$

1.0.3 Mutual Inductance Formula

$$M = \frac{\mu l}{2\pi} \left[\ln\left(\frac{2l}{s}\right) - 1 \right]$$

1.0.4 Equivalent Inductance Formula

$$L_{\text{eq}} = L_1 + L_2 - 2M_{12}$$

1.0.5 Effective Inductance Formula

$$L_{\text{eff}} = \frac{\mu l}{2\pi} \cosh^{-1}\left(\frac{2s}{d}\right)$$

1.0.6 Effective Inductance Formula

$$L_{\text{eff}} = \frac{\mu l s}{w}$$