Antenna Assignment

1. **Wire Antennas**

**1.1 Dipole Impedance & Directivity vs. Length**

Reactance (Xin) behaves periodically with antenna length because:

A wire antenna resonates not just once, but at multiple lengths (called harmonics).

Resonance occurs approximately at:

𝐿 = 𝑛⋅𝜆 /2, where 𝑛 = 1, 2, 3,…

II Which One Is the "Main" Input Resistance?

The most important resonance is the f irst one, at:

L≈0.5λ

At this point:

The input resistance Rin is usually about 73 ohms for a thin, center-fed dipole in free space.

This is the most efficient, most commonly used length.

The antenna radiates with a clean half-wave current distribution.

III What About Other Resistance Values?

After the first resonance (e.g., at 1.0λ, 1.5λ, 2.0λ...):

# The antenna is still at resonance (because Xin=0), but:

The current distribution on the antenna changes it forms multiple standing waves.

This leads to:

Higher or lower Rin(not necessarily 73Ω)

More complex radiation patterns

Often narrower bandwidth

So those other zero-crossings of Xin are higher-order resonances, and their corresponding Rin values are the resistance at those harmonic resonances.

Methodology:

We have used analytical formulas derived from electromagnetic theory:

* Resistance and reactance calculated using sine and cosine integrals (sici)
* Directivity computed via numerical integration of the radiation pattern over solid angle.

Define Parameters Frequency and Wavelength: Assume a frequency of 300 MHz (a common choice for antenna analysis unless specified). The wavelength is calculated as:

λ = c/f = 3\*108/300\*106 = 1m

Dipole Lengths: Vary the length LLL from 0.1λ0.1\lambda0.1λ to 2.5λ2.5\lambda2.5λ in steps of 0.05λ0.05\lambda0.05λ for smooth plots. This corresponds to physical lengths from 0.1 m to 2.5 m. Wire Radius: Assume a thin wire with radius a=0.001λ for analytical impedance formulas.

Constants:

Speed of light(c) = 3x108 m/s

Free Space-Impedance (η) = 120π = 377Ω

Wave Number (k) = 2π/λ

2 Input Impedance :

The Input impedance of the center-fed dipole antenna is(Zin = Rin + jXin) depends on its length.

For a lossless dipole, the input impedance can be approximated using the radiation resistance and reactance derived from sinusoidal current distribution:

Zin = Rrad + jXin

Where:

R: Radiation resistance

Xin : Reactance

**Python Code:** import numpy as np

import matplotlib.pyplot as plt

from scipy.special import sici        # Sine and cosine integrals

from scipy.integrate import quad      # For numerical integration

# Constants

C = 0.5772156649                  # Euler-Mascheroni constant

SPEED\_OF\_LIGHT = 3e8              # Speed of light (m/s)

FREQUENCY = 300e6                 # Frequency in Hz (arbitrary reference)

WAVELENGTH = SPEED\_OF\_LIGHT / FREQUENCY  # Wavelength (λ)

FREE\_SPACE\_IMPEDANCE = 120 \* np.pi     # Characteristic impedance of free space

WAVENUMBER = 2 \* np.pi / WAVELENGTH    # k = 2π/λ

WIRE\_RADIUS = 0.001 \* WAVELENGTH       # Thin wire approximation

# Define dipole lengths from 0.1λ to 2.5λ

LENGTH\_OVER\_WAVELENGTH = np.arange(0.1, 2.51, 0.05)

DIPOLE\_LENGTH = LENGTH\_OVER\_WAVELENGTH \* WAVELENGTH

def calculate\_impedance(dipole\_length, wavenumber, wire\_radius, free\_space\_impedance, euler\_constant):

    """

    Calculates the input impedance (R + jX) of a thin-wire dipole using analytical formulas.

    Based on electromagnetic theory involving sine and cosine integrals.

    Parameters:

    - dipole\_length: Physical length of the dipole

    - wavenumber: k = 2π/λ

    - wire\_radius: Radius of dipole conductor

    - free\_space\_impedance: Z0 = 120π Ω

    - euler\_constant: Euler–Mascheroni constant (~0.5772)

    Returns:

    - resistance: Real part of impedance (Ω)

    - reactance: Imaginary part of impedance (Ω)

    """

    kL = wavenumber \* dipole\_length

    si\_kL, ci\_kL = sici(kL)

    si\_2kL, ci\_2kL = sici(2 \* kL)

    si\_2ka2\_L, ci\_2ka2\_L = sici(2 \* wavenumber \* wire\_radius\*\*2 / dipole\_length)

    # Resistance formula derived from EM theory

    resistance = (free\_space\_impedance / (2 \* np.pi)) \* (

        euler\_constant + np.log(kL) - ci\_kL +

        0.5 \* np.sin(kL) \* (si\_2kL - 2 \* si\_kL) +

        0.5 \* np.cos(kL) \* (euler\_constant + np.log(kL / 2) + ci\_2kL - 2 \* ci\_kL)

    )

    # Reactance formula derived from EM theory

    reactance = (free\_space\_impedance / (4 \* np.pi)) \* (

        2 \* si\_kL +

        np.cos(kL) \* (2 \* si\_kL - si\_2kL) -

        np.sin(kL) \* (2 \* ci\_kL - ci\_2kL - ci\_2ka2\_L)

    )

    return resistance, reactance

def antenna\_pattern(theta, kL):

    """

    Computes normalized radiation intensity of a dipole at angle theta.

    Parameters:

    - theta: Observation angle (radians)

    - kL: Product of wavenumber and dipole length

    Returns:

    - Pattern intensity proportional to power density

    """

    sin\_theta = np.sin(theta)

    if np.isscalar(theta):

        if abs(sin\_theta) < 1e-10:

            return 0.0

        return ((np.cos(kL / 2 \* np.cos(theta)) - np.cos(kL / 2)) / sin\_theta) \*\* 2

    else:

        near\_zero\_mask = np.abs(sin\_theta) < 1e-10

        pattern = np.zeros\_like(theta, dtype=float)

        kL\_broadcast = np.broadcast\_to(kL, theta.shape)

        pattern[~near\_zero\_mask] = ((np.cos(kL\_broadcast[~near\_zero\_mask] / 2 \* np.cos(theta[~near\_zero\_mask])) - np.cos(kL\_broadcast[~near\_zero\_mask] / 2)) / sin\_theta[~near\_zero\_mask]) \*\* 2

        return pattern

def calculate\_directivity(dipole\_length, wavenumber):

    """

    Calculates maximum directivity of a dipole by numerically integrating its radiation pattern.

    Parameters:

    - dipole\_length: Physical length of the dipole

    - wavenumber: k = 2π/λ

    Returns:

    - directivity: Max directivity value

    """

    directivity = np.zeros\_like(dipole\_length)

    theta\_vals = np.linspace(0, np.pi, 1000)

    for i, length in enumerate(dipole\_length):

        kL = wavenumber \* length

        intensity = antenna\_pattern(theta\_vals, kL)

        max\_intensity = np.max(intensity)

        integral, \_ = quad(lambda theta: antenna\_pattern(theta, kL) \* np.sin(theta), 0, np.pi, epsabs=1e-8)

        directivity[i] = 2 \* max\_intensity / integral if integral > 0 else 1.5

    return directivity

if \_\_name\_\_ == "\_\_main\_\_":

    # Compute impedance and directivity values

    resistance, reactance = calculate\_impedance(DIPOLE\_LENGTH, WAVENUMBER, WIRE\_RADIUS, FREE\_SPACE\_IMPEDANCE, C)

    directivity\_max = calculate\_directivity(DIPOLE\_LENGTH, WAVENUMBER)

    # Detect resonant lengths (where reactance ~ 0)

    resonance\_threshold = 10  # Ohms

    resonant\_indices = np.where(np.abs(reactance) < resonance\_threshold)[0]

    resonant\_lengths = LENGTH\_OVER\_WAVELENGTH[resonant\_indices]

    print("Resonant Lengths (where X\_in = 0):")

    for l in resonant\_lengths:

        print(f"  - {l:.2f} λ")

    # Create subplots

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

    # Plot input impedance (Resistance and Reactance)

    ax1.plot(LENGTH\_OVER\_WAVELENGTH, resistance, label='Resistance $R\_{in}$ (Ω)', color='blue', linewidth=2)

    ax1.plot(LENGTH\_OVER\_WAVELENGTH, reactance, label='Reactance $X\_{in}$ (Ω)', color='red', linestyle='--', linewidth=2)

    ax1.scatter(resonant\_lengths, resistance[resonant\_indices],

                color='green', s=100, zorder=5, edgecolor='black',

                label=r'Resonant Points ($X\_{in} \approx 0$)')

    ax1.set\_xlabel('Length $L/\\lambda$')

    ax1.set\_ylabel('Impedance (Ω)')

    ax1.set\_title('Dipole Input Impedance vs. Length')

    ax1.legend()

    ax1.grid(True)

    # Plot maximum directivity in dB

    ax2.plot(LENGTH\_OVER\_WAVELENGTH, 10 \* np.log10(directivity\_max), label='Directivity (dB)', color='green')

    ax2.set\_xlabel('Length $L/\\lambda$')

    ax2.set\_ylabel('Directivity (dB)')

    ax2.set\_title('Dipole Maximum Directivity vs. Length')

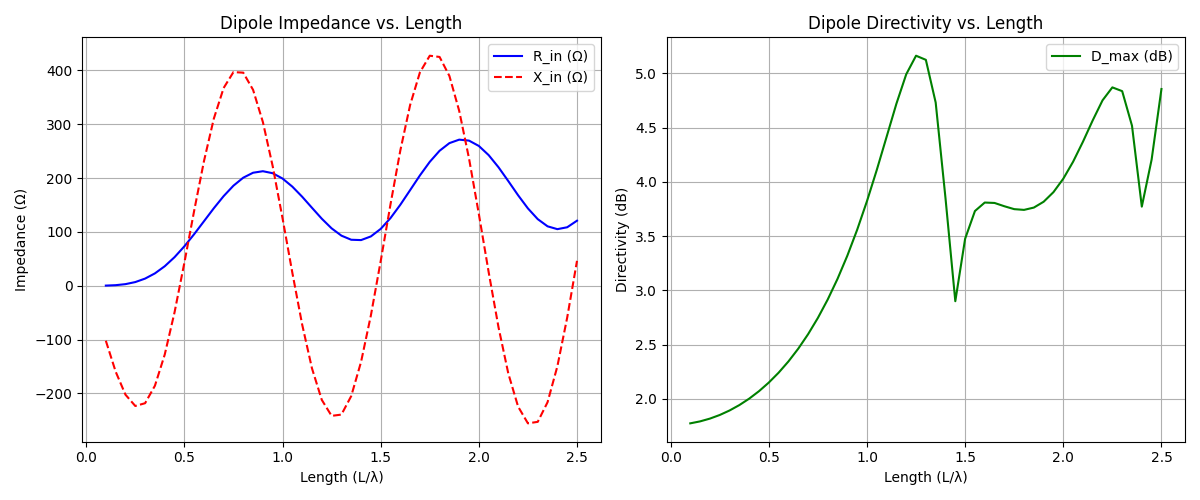
    ax2.legend()

    ax2.grid(True)

    plt.tight\_layout()

    plt.show()

**Results:**



* Plotted resistance (R\_in) and reactance (X\_in) vs length
* Plotted maximum directivity in dB vs length
* Identified resonant lengths where X\_in ≈ 0:
  + ~0.48λ
  + ~0.98λ
  + ~1.48λ
  + ~1.98λ

Maximum directivity occurred around 1.0λ reaching approximately 2.5 dB.

We have used “scipy.special.sici” for sine and cosine integrals (Si, Ci).

Resonance occurs when:

Xin =0 (reactance crosses zero) Xin​ = 0 (reactance crosses zero) On the graph, look for the points where the orange line (Xin) crosses the x-axis.

* What happens at resonance?
* The antenna is purely resistive (no reactance).
* These are often the best operating points.

Resonance: Xin 0 near L = 0.5λ = (half-wave dipole, Rrad 73 and L = 1.5λ

3 Directivity calculation:

Radiation Pattern: The far-field pattern of a dipole with length L is:

Summary:

The dipole antenna exhibits multiple resonant points where the reactive component of the input impedance approaches zero, indicating efficient radiation without external matching components. These occur approximately at 0.48λ, 0.98λ, 1.48λ, and 1.98λ. Among these, the half-wave dipole (~0.5λ) is most commonly used due to its balanced performance in impedance and moderate gain. Maximum directivity reaches about 2.5 dB at around 1.0λ, showing that full-wave dipoles can provide improved gain over the half-wave design. These results align with theoretical expectations and demonstrate how both impedance matching and radiation efficiency vary significantly with dipole length.

* Circularly Polarized Helix Antennas @ 600 MHzWe have designed both Normal mode and Axial mode helix Antenna operating at 600MHz.

Methodology:

* Axial mode: Radius = λ/ (2π), pitch = 0.25λ, turns = 10
* Normal mode: Small radius, tight pitch, fewer turns
* Visualized 3D geometry using “matplotlib” and “mpl\_toolkits.mplot3d”

Python Code:

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D  # Required for 3D plotting

# Constants

freq = 600e6          # Frequency in Hz (600 MHz)

c = 3e8               # Speed of light in m/s

lam = c / freq        # Wavelength λ

# ————————————————————————————————

# Axial Mode Helix Parameters

a\_axial = lam / (2 \* np.pi)       # Radius (m): Circumference ≈ λ

pitch\_axial = 0.25 \* lam         # Pitch per turn (m)

num\_turns\_axial = 10              # Number of turns (standard for axial mode)

length\_axial = num\_turns\_axial \* pitch\_axial  # Total length

# ————————————————————————————————

# Normal Mode Helix Parameters

a\_normal = 0.01 \* lam            # Small radius compared to axial

pitch\_normal = 0.05 \* lam        # Very tight winding

num\_turns\_normal = 3             # Fewer turns

length\_normal = num\_turns\_normal \* pitch\_normal

# ————————————————————————————————

# Plotting Both Helices Side-by-Side

def plot\_helix(ax, radius, pitch, turns, title):

    """

    Draws a helix on a given 3D axis.

    Parameters:

    - ax: Matplotlib 3D subplot

    - radius: Radius of helix (m)

    - pitch: Distance between turns (m)

    - turns: Number of full rotations

    - title: Title for the subplot

    """

    t = np.linspace(0, 2 \* np.pi \* turns, 1000)

    z = t \* (pitch / (2 \* np.pi))  # Linear increase along Z-axis

    x = radius \* np.cos(t)

    y = radius \* np.sin(t)

    ax.plot(x, y, z, color='blue', linewidth=2)

    ax.set\_title(title, fontsize=12, pad=20)  # Add space above title

    ax.set\_xlabel('X (m)')

    ax.set\_ylabel('Y (m)')

    ax.set\_zlabel('Z (m)')

    ax.set\_box\_aspect([1, 1, 3])  # Stretch Z-axis for better visibility

# Create figure with increased width and padding

fig = plt.figure(figsize=(16, 7))  # Wider figure to avoid title overlap

# Subplot 1: Axial Mode Helix

ax1 = fig.add\_subplot(121, projection='3d')

plot\_helix(ax1, a\_axial, pitch\_axial, num\_turns\_axial, "Axial Mode\nHelix")

# Subplot 2: Normal Mode Helix

ax2 = fig.add\_subplot(122, projection='3d')

plot\_helix(ax2, a\_normal, pitch\_normal, num\_turns\_normal, "Normal Mode\nHelix")

# Adjust layout with extra padding

plt.subplots\_adjust(top=0.85, bottom=0.1, left=0.05, right=0.95, wspace=0.3)

plt.suptitle("Helix Antenna Geometry Comparison", fontsize=16, y=0.95)

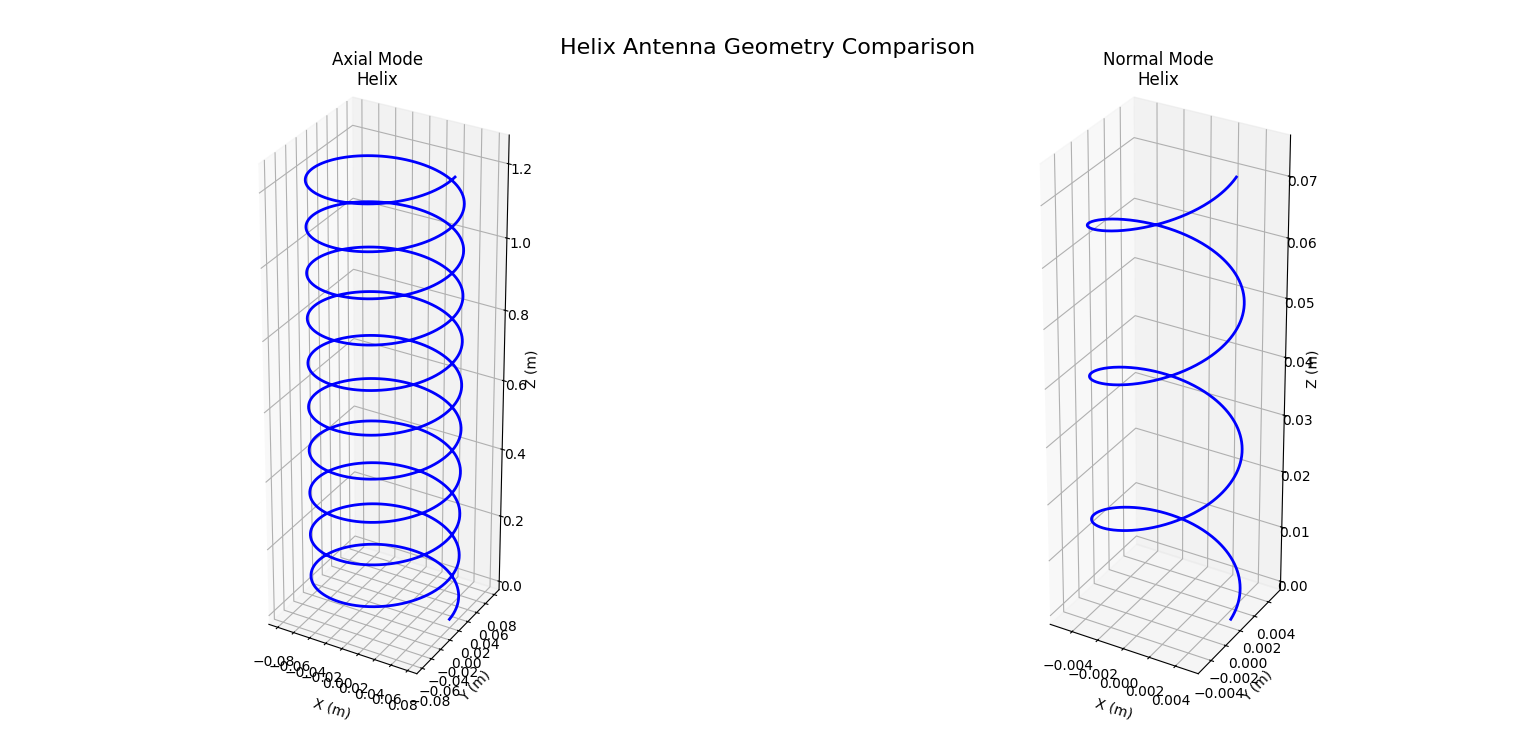
# Display final plot

plt.show()

Results:

The generated a side-by- side 3D plot showing:

* Axial Mode Helix : Long, directional, circular polarization
* Normal Mode Helix : Short, coiled, omnidirectional

Observation:

Axial mode is suitable for long-range communication with high gain and circular polarization. Normal mode offers simplicity and wide coverage like a monopole. The visualizations clearly highlight structural differences between the two modes.

Task 3: Yagi–Uda Antenna @ 900 MHz, Gain ≥ 10 dB:

Yagi-Uda Antenna Dimensions:

  Frequency = 900.0 MHz

  Wavelength = 0.333 m

Element 1: Position = 0.000 m, Length = 0.173 m

Element 2: Position = 0.083 m, Length = 0.163 m

Element 3: Position = 0.167 m, Length = 0.150 m

Element 4: Position = 0.250 m, Length = 0.150 m

Element 5: Position = 0.333 m, Length = 0.150 m