IRS SIMULATOR

Guide: Dr.Sidharta Sarma

NOVEMBER 20, 2022

N Divya Lakshmi –S22007
Kashyap Raprthi-T22101

Stage 1: Single IRS component on ground between Transmitter & Receiver

A. THE CONVENTIONAL TWO-RAY SYSTEM MODEL

The two-rays ground-reflection model is a multipath radio propagation model which predicts the path losses between a transmitting antenna and a receiving antenna when they are in line of sight (LOS)

Generally, the two antennas each have different height.

Two ray model considers both the direct path and a ground reflected propagated path between transmitter and receiver

- The proposed 2-bit RIS element operates in the S band with a centre frequency of 2.4 GHz.
- The element spacing is 50 mm.
- Assuming that the transmit power of x(t) is Pt, the received power Pr can be expressed, from (1), in terms of Pt as follows:
- We have elements with tuneable reflection coefficient (Input to the code)

```
Flag=0
def Exception_Checker(List,Variable):
    global Flag
    try:
        if  Variable<=List[1] and Variable>=List[0]:
            pass
        else:
            Flag=4
            raise Invalid_Input

    except Invalid_Input:
        print("Enter the value in specified range")


def Quit(Flag):
    if Flag==4:
        exit()
```

Custom exception for invalid inputs

Checking Parameter range

Function to stop execution when invalid inputs are given

```
# Inputs for various parameters
try:
    D=float(input("Enter the Distance between Tx & Rx Antenna (between 5m - 50m) : "))
    Exception_Checker([5,50],D)
    Quit(Flag)

    h_t=float(input("Enter the height of Tx Antenna (between 5m - 10m) : "))
    Exception_Checker([5,10],h_t)
    Quit(Flag)

    h_r=float(input("Enter the height of Rx Antenna (between 5m - 8m) : "))
    Exception_Checker([5,8],h_r)
    Quit(Flag)

    f=float(input("Enter the frequency of signal (between 1GHz - 5Ghz) : "))
    Exception_Checker([1,5],f)
    Quit(Flag)

    pos=float(input(f"Enter the position of IRS (between {D/4} - {3*D/4}) : "))
    Exception_Checker([D/4,3*D/4],pos)
    Quit(Flag)
```

Parameter	Range
D (m)	5 to 50
ht (m)	5 to 10
hr (m)	5 to 8
f (GHz)	1 to 5
c (m/sec)	3 x (10^8)
f (Hz)	2.4 x (10^9)
λ (m)	0.125

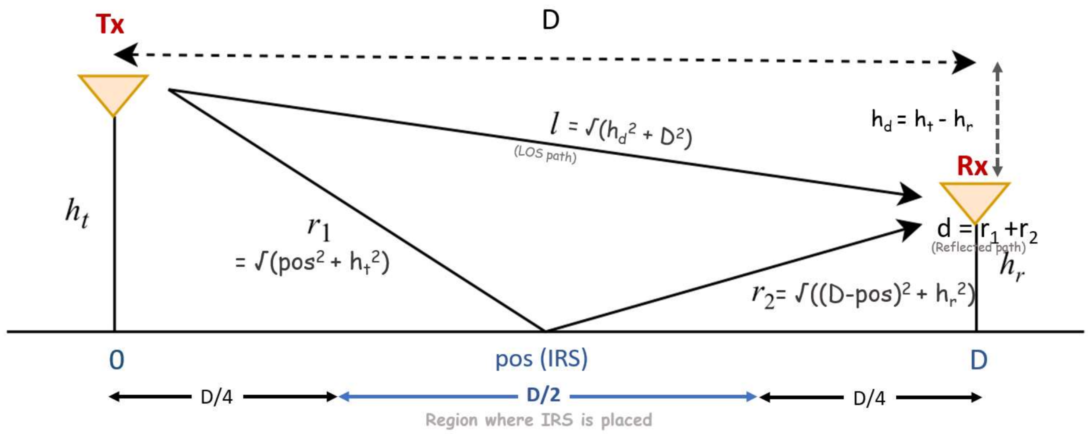


FIGURE 1. **Two-ray propagation model with a LOS ray and a ground-reflected ray**

```
class IRS:

    @staticmethod
    def Distance(pos,h_t=10,h_r=5):
        r1=np.sqrt((pos**2)+(h_t**2))
        r2=np.sqrt(((D-pos)**2)+(h_r**2))
        d=r1+r2
        return d

    def Gamma(pos):
        R=complex(np.cos((2*np.pi*(pos-l))/Lambda),np.sin((2*np.pi*(pos-l))/Lambda))
        return abs(R)
```

$\boxed{r_1 = \sqrt{(pos^2 + h_t^2)}}$

$\boxed{r_2 = \sqrt{((D-pos)^2 + h_r^2)}}$

$\boxed{R = e^{j\Delta\phi}}$ $\boxed{\Delta\phi = \frac{2\pi(r_1+r_2-l)}{\lambda}}$ $\boxed{\lambda = c / f}$

At IRS, we input:

1. Position of IRS (Coordinates)
   Function returns total distance (d) traversed by the wave
2. Reflection Coefficient (Γ)
3. Reflection coefficient is a complex number whose magnitude takes values between -1 and +1. We can represent Gamma as:

$$\Gamma = a.R = a.(e^{j\,\Delta\Phi}) = a\,(\cos\Delta\Phi + j\sin\Delta\Phi)$$

Analytical Method to find AF & Phase difference of received signal

```python
@staticmethod
def AF_Phase(Distance):
    a=(Lambda/(4*np.pi))*(1/l)*np.cos((2*np.pi*l)/Lambda)
    b=(-1)*(Lambda/(4*np.pi))*(1/l)*np.sin((2*np.pi*l)/Lambda)
    x=0
    for i in range(len(Distance)):
        M = (Lambda/(4*np.pi))*(IRS.Gamma(Distance[i])/Distance[i])
        phase=((-2)*np.pi*(l))/Lambda
        a += (M*np.cos(phase))
        b += (M*np.sin(phase))
        x += (2*np.pi*(Distance[i])/Lambda)

    AF=(abs(complex(a,b)))**2

    Phase=x-((2*np.pi*l)/Lambda)
    while(True):
        if Phase <= -360:
            Phase+=360
        elif Phase >= 360:
            Phase-=360
        elif -360<=Phase and Phase<=360:
            break

    return AF,Phase
```

$x(t) = M_1e^{j\theta 1} + M_2e^{j\theta 2}$

$x(t) = (M_1\cos\theta_1 + M_2\cos\theta_2) + j(M_1\sin\theta_1 + M_2\sin\theta_2)$

$\sqrt{(a^2 + b^2)}\ \angle\tan^{-1}(b/a)$

$0 < \alpha < 1$
$\pi < \Phi < \pi$
$\Rightarrow |R| < 1$

```python
@staticmethod
def Attenuation_Factor(Distance):
    x=0
    y=0
    for i in range(len(Distance)):
        x+=(1/Distance[i])
        y+=Distance[i]
    AF=((Lambda/(4*np.pi))**2)*(((1/l)+x)**2)
    Phase=2*np.pi*(y-l)/Lambda

    while(True):
        if Phase <= -360:
            Phase+=360
        elif Phase >= 360:
            Phase-=360
        elif -360<=Phase and Phase<=360:
            break
    return AF,Phase
```

Attenuation Factor using Derived Formula

Results:

$$P_r = P_t\left(\frac{\lambda}{4\pi}\right)^2 \underbrace{\left|\frac{1}{l} + \sum_{i=1}^{N}\frac{R_i \times e^{-j\Delta\phi_i}}{r_{1,i} + r_{2,i}}\right|^2}_{\text{Attenuation Factor (AF)}}$$

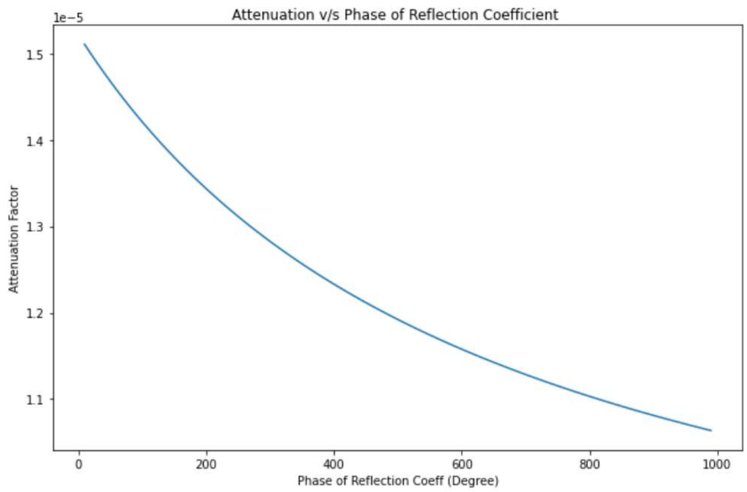
$$R_i = e^{j\Delta\phi_i}$$

$$\Delta\phi_i = \frac{2\pi\,(r_{1,i} + r_{2,i} - l)}{\lambda}$$

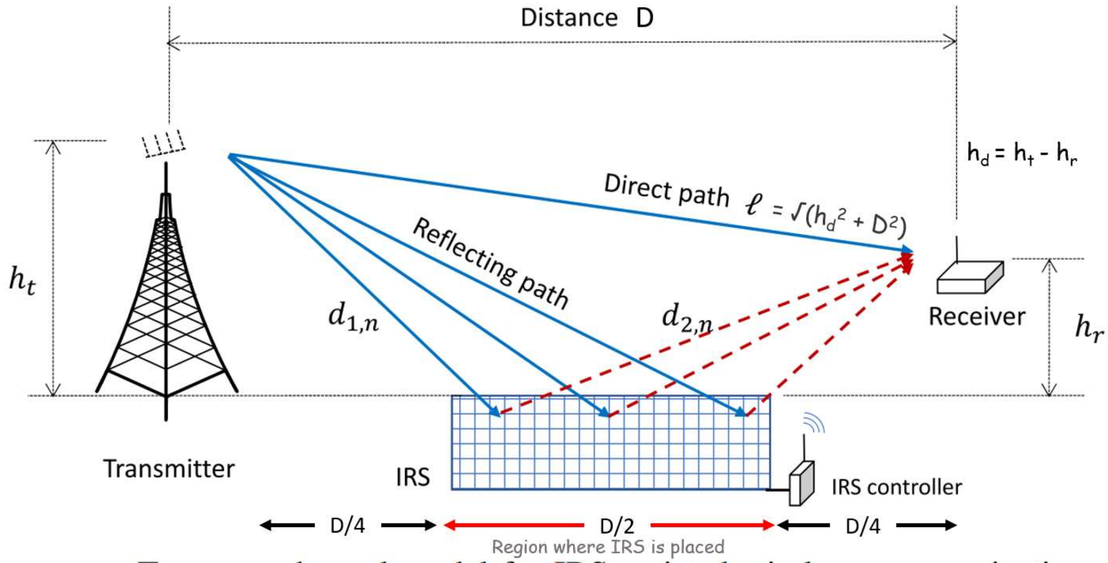
```
Stage1= IRS()               #object instantiation
Distance=np.array([Stage1.Distance(pos)])
AF,Phase=Stage1.AF_Phase(Distance)
print("Total Distance travelled by reflected wave :",Distance)
print(f"Attenuation Factor & Phase difference(degree) of received signal are {AF} & {Phase} respectively")

AF_Derived,Phase_Derived=Stage1.Attenuation_Factor(Distance)
print("Attenuation Factor from derived formula :",AF_Derived)
print("Phase Difference(degree) from derived formula :",Phase_Derived)
```

```
Enter the Distance between Tx & Rx Antenna (between 5m - 50m) : 12
Enter the height of Tx Antenna (between 5m - 10m) : 10
Enter the height of Rx Antenna (between 5m - 8m) : 5
Enter the frequency of signal (between 1GHz - 5Ghz) : 2.4
Enter the position of IRS (between 3.0 - 9.0) : 5
Total Distance travelled by reflected wave : [19.78266515]
Attenuation Factor & Phase difference(degree) of received signal are 1.6078017934380174e-06 & 340.93393634027666 respectively
Attenuation Factor from derived formula : 1.6078017934380172e-06
Phase Difference(degree) from derived formula : 340.9339363402767
```

Attenuation Vs Phase of Reflection Coefficient:



Stage 2: Multiple IRS component on ground between Transmitter & Receiver



Two-way channel model for IRS-assisted wireless communications.

```
pos=np.arange(D/4,3*D/4,0.06)
Stage2= IRS()
Distance=np.array([])
with open('Positions_Stage2.csv',"a") as f:
    for i in pos:
        x=Stage2.Distance(i)
        Distance=np.append(Distance,x)
        f.write(str(x))
        f.write("\n")
```

Object instantiation

Writing the positions into a file

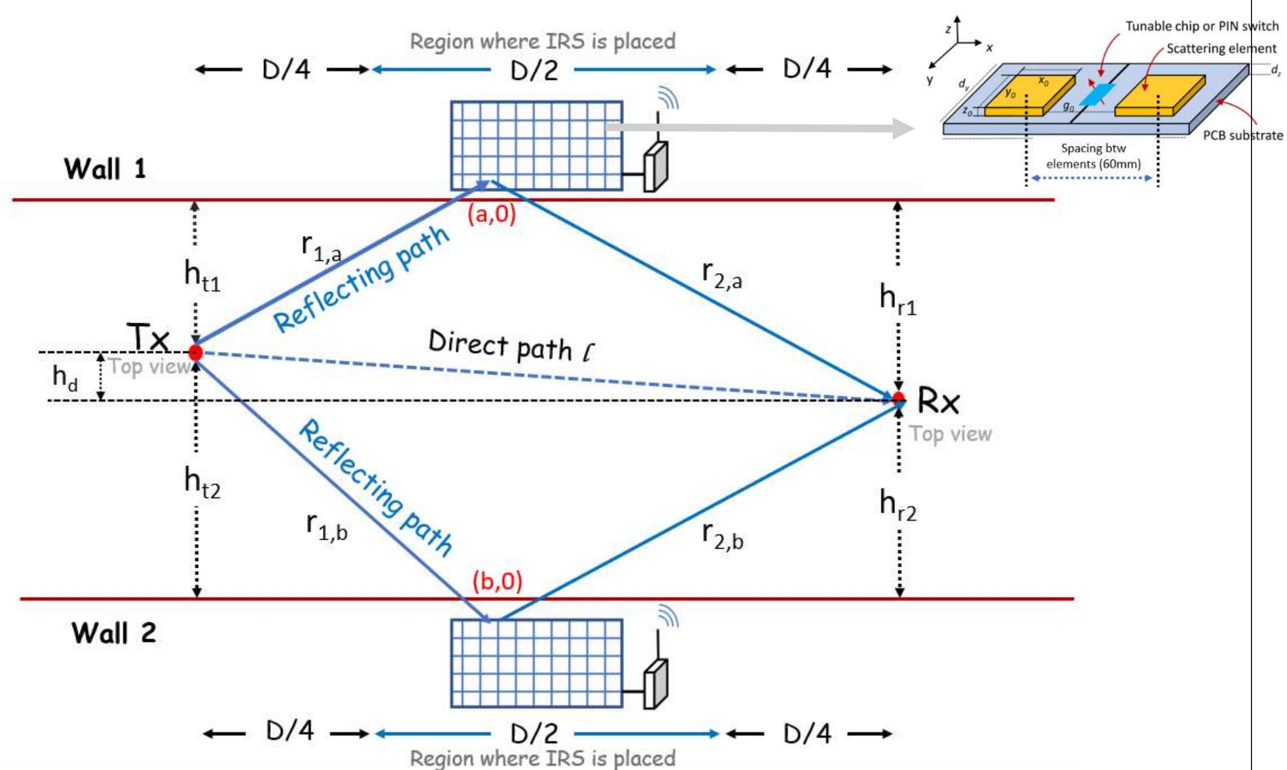
Position of IRS components of 60 mm length between D/4 distance from Tx & D/4 away from Rx

Stage 2: Result:

```
AF,Phase=Stage2.AF_Phase(Distance)
print(f"Attenuation Factor & Phase difference(degree) of received signal are {AF} & {Phase} respectively")

AF_Derived,Phase_Derived=Stage1.Attenuation_Factor(Distance)
print("Attenuation Factor from derived formula :",AF_Derived)
print("Phase Difference(degree) from derived formula :",Phase_Derived)
```

```
Attenuation Factor & Phase difference(degree) of received signal are 0.0026417484747550536 & 230.40160872494744 respectively
Attenuation Factor from derived formula : 0.0026417484747550553
Phase Difference(degree) from derived formula : 230.4016087249911
```

Stage 3: Two walls containing Multiple IRS components




```
# Inputs for various parameters
try:
    D=float(input("Enter the Distance between Tx & Rx Antenna (between 5m - 50m) : "))
    Exception_Checker([5,50],D)
    Quit(Flag)

    h_t1=float(input("Enter the Distance of Tx Antenna from wall 1 (between 2m - 20m) : "))
    Exception_Checker([2,20],h_t1)
    Quit(Flag)

    h_r1=float(input("Enter the Distance of Rx Antenna from wall 1 (between 2m - 20m) : "))
    Exception_Checker([2,20],h_r1)
    Quit(Flag)

    h_t2=float(input("Enter the Distance of Tx Antenna from wall 2 (between 2m - 20m) : "))
    Exception_Checker([2,20],h_t2)
    Quit(Flag)

    h_r2=float(input("Enter the Distance of Rx Antenna from wall 2 (between 2m - 20m) : "))
    Exception_Checker([2,20],h_r2)
    Quit(Flag)

    f=float(input("Enter the frequency of signal (between 1GHz - 5Ghz) : "))
    Exception_Checker([1,5],f)
    Quit(Flag)
```

Parameter	Range
D (m)	5 to 50
ht1 (m)	2 to 20
hr1 (m)	2 to 20
ht2 (m)	2 to 20
hr2 (m)	2 to 20
f (GHz)	1 to 5
λ (m)	0.125
spacing btw elements (mm)	60

```
except ValueError:
    Flag=4
    print("Enter only Integers in given range")

Quit(Flag)
# h_t2=8
# h_r2=6
pos_Wall_1=np.arange(D/4,3*D/4,0.06)
pos_Wall_2=np.arange(D/4,3*D/4,0.06)
```

Exception Handling: Throws ValueError if entered parameter values lie outside the range

a : pos_Wall_1
b : pos_Wall_2

```
Stage3= IRS()

Distance=np.array([])

with open('Distance_Traversed_Stage3.csv',"a") as f:
    for i in pos_Wall_1:
        x=Stage3.Distance(i,h_t,h_r)
        Distance=np.append(Distance,x)
        f.write(str(x))
        f.write("\n")
    for i in pos_Wall_2:
        x=Stage3.Distance(i,h_t2,h_r2)
        Distance=np.append(Distance,x)
        f.write(str(x))
        f.write("\n")
with open("Distance_Traversed_Stage3.csv","r") as f:
    Data=f.read()
    Data=Data.splitlines()
```

Object Instantiation

File creation: Storing, in a file, the distance traversed by wave for each IRS component

Stage 3: Result:

```
D=np.array([])
for i in Data:
    D=np.append(D,float(i))

# print(type(D))
AF,Phase=Stage3.AF_Phase(D)
print(f"Attenuation Factor & Phase difference(degree) of received signal are {AF} & {Phase} respectively")

AF_Derived,Phase_Derived=Stage1.Attenuation_Factor(D)
print("Attenuation Factor from derived formula :",AF_Derived)
print("Phase Difference(degree) from derived formula :",Phase_Derived)
```

distance traversed by wave for each IRS component

```
Attenuation Factor & Phase difference(degree) of received signal are 0.1351886010632874 & 151.49210320203565 respectively
Attenuation Factor from derived formula : 0.13518860106328756
Phase Difference(degree) from derived formula : 151.4921032008715
```

REFERENCES:

[1] Toward Smart Wireless Communications via Intelligent Reflecting Surfaces: A Contemporary Survey

[2] Reconfigurable Intelligent Surface-Based Wireless Communications: Antenna Design, Prototyping, and Experimental Results

[3] Wireless Communications Through Reconfigurable Intelligent Surfaces