Guide for using code base PLS_IRS

This document provides a quick walk through in how to run the code used to generate the results reported in the paper *"Balancing Secrecy and Throughput via Network Layer Control for Intelligent Reflective Surface Aided Wireless Environments".* Note that functions and script not listed in this paper are described within the MATLAB code, including all function definitions, inputs and outputs.

mainLOSIRS.m

This file allows the operator to randomly generate a scenario (or a preset scenario of their choosing) to observe IRS and AP interaction with the environment. To load a preset scenario, either modify exampleScenario.m as shown in Fig. 1 or duplicate the exampleScenario.m file with a different name and then change the parameters.

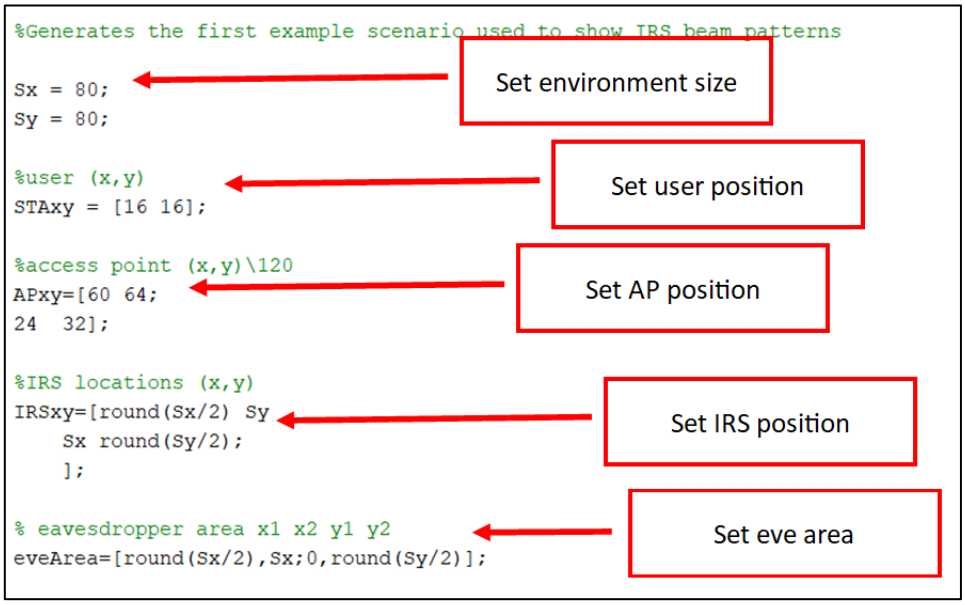


Figure 1: exampleScenario.m parameters

Note that you may only set one user position, otherwise an error will be thrown. The IRS positions must be so either the x dimension is 0 or Sx, or, the y dimension is 0 or Sy. This places IRS on the environment border.

The following Fig. 2 outlines the adjustable parameters in mainLOSIRS.m. Fig. 2a shows the physical network operating parameters that can be adjusted based on the type of network, the maximum powers allowed by the AP and also the secrecy biasing term σ. Fig. 2b shows the IRS parameters, Fig. 2c shows the environment initialisation which places the nodes in the environment. Fig. 2d allows the operator to adjust the resolution at which the images are generated, as well as the resolution of the eavesdropper region used to calculate EASOR.

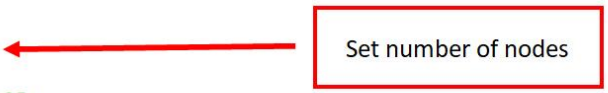
```
%% network operating parameters

%Carrier frequency [Hz]
f=2.4*10^9;
%speed of light [constant]
c=3*10^8;
%wavelength [m]
lambda = c/f;

%transmit power [watts]
ptmax = 1;

%maximum jammer power [watts]
pjmax = 1;

%transmit gain and receiver gain [Linear.]
Gk=1;
GSTA=1;

%noise floor [Watt]
N0 = 10^-12;

%bandwidth [Hz]
W = 20*10^6;

%security bias
sigma=1;
```

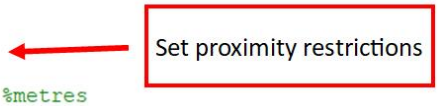
a) network operating parameters

```
%% IRS parameters

%IRS dimensions [element]
Na=20;
Nb=20;

%element size [m]
Le=lambda/2;
```

b) Intelligent reflective surface parameters

```
%% environment set up

%load the example scenario
exampleScenario;

%{
%this code for a random environment
    Sx=80;
    Sy=80;
    numIRS=2;
    numAP=2;
    percentEve=25;

    %proximity conditions
    minDistanceAP = 10; %metres
    minDistanceIRS=10; %metres
    minDistanceAPB=10; %metres
    minDistanceUB=farfieldrelax; %metres

    %[eveArea, STAxy, APxy, IRSxy,orientationIRS] = generate_random_scen(Sx
%}
```





c) environment initialization

```
%% resolution

%RESOLUTION of images
photoRes=1; %data points calculated / metre. increase for higher resolution
Nx = Sx*photoRes;
Ny = round(Nx*Sy/Sx);
dx = Sx/Nx;
xa = (0:Nx)*dx;
dy = Sy/Ny;
ya = (0:Ny)*dy;
eveRes=1; %for EASOR calculation, increase for higher resolution
```

d) resolution adjustment

Figure 2: mainLOSIRS.m parameters

The command window will print the key network layer configurations for the environment.

```
Transmit AP 2
With IRS associations
     2
     1

transmit Powers
     1
     1

User capacity Mbit/s
    0.4951

EASOR percentage
     9
```

Figure 3: command output of mainLOSIRS.m

The colour map figures generated by colour_maps.m are:

1. SINR for each location in the environment
2. Transmitted signal pathloss for each IRS
3. Signal pathloss from transmit AP
4. Jammer pathloss from all jamming AP
5. Node-based visualisation of generated scenario
6. Phase shifts of each IRS
7. Combined jamming signals pathloss for each IRS

visualize_random_scenario.m

This code allows generation, saving, loading of random scenarios for quick visualisation as shown in Fig. 4. The operator needs to define the number of nodes, environment sizes and proximity restrictions.

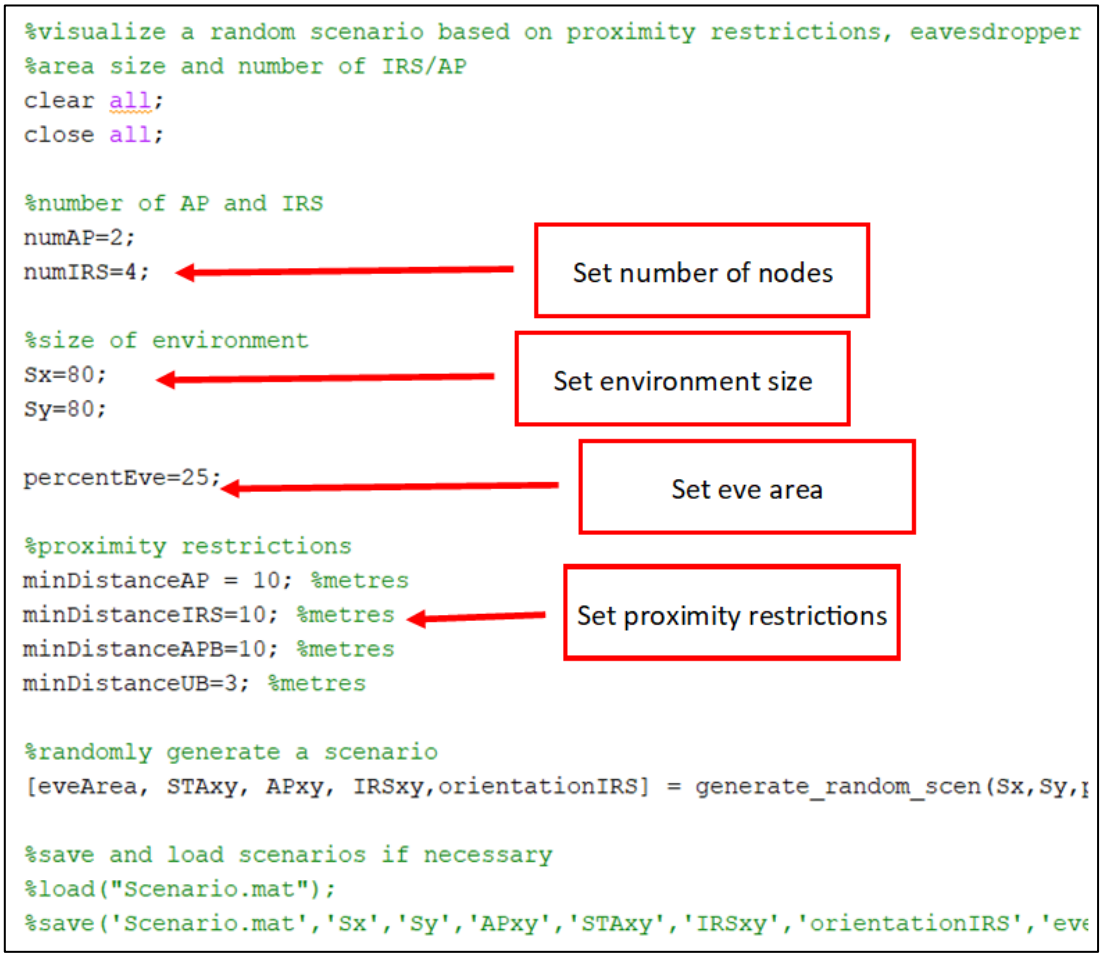


Figure 4: Visualise a randomly generated scenario

plotScenarioMaps.m

This code is used to analyse defined scenarios. It generates colour maps for AP connections, IRS connections, FJ power, user throughput and EASOR across the scenario. It also calculates the average EASOR and user throughput, as well as the minimum and maximum bounds on each. The network initialisation is effectively identical to the rest of the described code.

```
%% enable the parts of the algorithm you want

%CHANGE THESE
IRSon=true;
FJon=true;
```

Figure 5: Manually enable algorithms

As shown in Fig. 5, to switch on parts of the algorithm these values can be swapped between true and false. This data is collated and compared to baselines in **barCharts.m**.

plotScenarioSigma.m

This code does the same as plotScenarioMaps but plots the average user throughput and EASOR for changing values of secrecy bias σ.

montecarlo_SX.m

Perform monte carlo simulation of environment with changing IRS and environment size S_x.

montecarlo_SIGMA.m

Perform monte carlo simulation of environment with changing IRS and secrecy bias σ.

montecarlo_EVEAREA.m

Perform monte carlo simulation of environment with changing IRS and eavesdropper area size $\xi S_x S_y$.

montecarlo_BUDGET.m

Perform monte carlo simulation of environment with changing IRS under element budget $N_a N_b$.

montecarlo_APIRS.m

Perform monte carlo simulation of environment with changing IRS and AP.

Future proofing:

The IRS orientation can be manually configured if desired and could be placed anywhere in the environment. This requires adjustment of the code in exampleScenario.m such that the IRS orientations are not automatically generated. The IRS have a 'dark side' such that the environment area on the far side from the AP relative to the IRS is not illuminated by the signal dissipated by the IRS. This code is shown in Fig. 5.

```
elseif orientationIRS(ixIRS)=="vertical"
    %check the 'dark side' of IRS along x axis

    if(IRSx(ixIRS)<APx)
        xlim1=IRSx(ixIRS)/dx+1;
        xlim2=Nx;
    else
        xlim1=1;
        xlim2=IRSx(ixIRS)/dx;
    end
```

Figure 6: IRS illumination restriction

Additional data files:

Scenario 1,2 and 3 are labelled accordingly for use in the same directory. All results collected for the monte carlo simulations are stored in the file *data*.