```
1 !pip install arlpy
```

```
Collecting arlpy
  Downloading arlpy-1.8.4.tar.gz (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.2/44.2 kB 1.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.18.1 in /usr/local/lib/python3.10/dist-packages (from arlpy) (1.23.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from arlpy) (1.11.4)
Collecting utm>=0.5.0 (from arlpy)
  Downloading utm-0.7.0.tar.gz (8.7 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from arlpy) (1.5.3)
Requirement already satisfied: bokeh>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from arlpy) (3.3.1)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (3.1.2)
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (1.2.0)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (23.2)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (9.4.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (6.3.2)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.4.0->arlpy) (
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->arlpy)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->arlpy) (2023.3.po
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=2.9->bokeh>=1.4.0->ar
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=1.0.
Building wheels for collected packages: arlpy, utm
  Building wheel for arlpy (setup.py) ... done
  Created wheel for arlpy: filename=arlpy-1.8.4-py3-none-any.whl size=49566 sha256=b41b77aef4aeceb9484aac0bf9aebd8ef98af13d6f
  Stored in directory: /root/.cache/pip/wheels/1c/6d/89/8ac02e6c0c7cde3fcbf8c1c6936474b2fe2d1efc78b368c0ae
  Building wheel for utm (setup.py) ... done
  Created wheel for utm: filename=utm-0.7.0-py3-none-any.whl size=6084 sha256=70209aa130fea698733d8260044ab27a5808aa2ffb035f7
  Stored in directory: /root/.cache/pip/wheels/2f/a1/c8/543df0e8f5e824c3e92a432e32deb9cd89ae686095ee8cfcbe
Successfully built arlpy utm
Installing collected packages: utm, arlpy
Successfully installed arlpy-1.8.4 utm-0.7.0
```

```
1 Start coding or generate with AI.
```

```
1 !wget http://oalib.hlsresearch.com/AcousticsToolbox/at_2020_11_4.zip
```

```
--2023-12-10 11:47:54--  http://oalib.hlsresearch.com/AcousticsToolbox/at_2020_11_4.zip
Resolving oalib.hlsresearch.com (oalib.hlsresearch.com)... 71.6.189.79
Connecting to oalib.hlsresearch.com (oalib.hlsresearch.com)|71.6.189.79|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35156907 (34M) [application/zip]
Saving to: ‘at_2020_11_4.zip’

at_2020_11_4.zip    100%[===================>]  33.53M  55.6MB/s    in 0.6s

2023-12-10 11:47:55 (55.6 MB/s) - ‘at_2020_11_4.zip’ saved [35156907/35156907]
```

```
1 !unzip at_2020_11_4.zip
```

```
  inflating: __MACOSX/at_2020_11_4/Bellhop/._Bellhop3D.f90
  inflating: at_2020_11_4/Bellhop/influence.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._influence.f90
  inflating: at_2020_11_4/Bellhop/Bellhop3D User Guide 2016_7_25.pdf
  inflating: __MACOSX/at_2020_11_4/Bellhop/._Bellhop3D User Guide 2016_7_25.pdf
  inflating: at_2020_11_4/Bellhop/WriteRay.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._WriteRay.f90
  inflating: at_2020_11_4/Bellhop/ArrMod.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._ArrMod.f90
  inflating: at_2020_11_4/Bellhop/Step.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._Step.f90
  inflating: at_2020_11_4/Bellhop/Cone.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._Cone.f90
  inflating: at_2020_11_4/Bellhop/bdry3DMod.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._bdry3DMod.f90
  inflating: at_2020_11_4/Bellhop/bdryMod.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._bdryMod.f90
  inflating: at_2020_11_4/Bellhop/Step3DMod.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._Step3DMod.f90
  inflating: at_2020_11_4/Bellhop/RayNormals.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._RayNormals.f90
  inflating: at_2020_11_4/Bellhop/ReadEnvironmentBell.f90
  inflating: __MACOSX/at_2020_11_4/Bellhop/._ReadEnvironmentBell.f90
  inflating: at_2020_11_4/tslib/sourceMod.f90
  inflating: __MACOSX/at_2020_11_4/tslib/._sourceMod.f90
  inflating: at_2020_11_4/tslib/rfft.f90
  inflating: __MACOSX/at_2020_11_4/tslib/._rfft.f90
  inflating: at_2020_11_4/tslib/.DS_Store
  inflating: __MACOSX/at_2020_11_4/tslib/._.DS_Store
  inflating: at_2020_11_4/tslib/cans.f90
  inflating: __MACOSX/at_2020_11_4/tslib/._cans.f90
  inflating: at_2020_11_4/tslib/Makefile
  inflating: __MACOSX/at_2020_11_4/tslib/._Makefile
  inflating: at_2020_11_4/tslib/preenv.f90
  inflating: __MACOSX/at_2020_11_4/tslib/._preenv.f90
  inflating: at_2020_11_4/tslib/hilbert.f90
  inflating: __MACOSX/at_2020_11_4/tslib/._hilbert.f90
  inflating: at_2020_11_4/tslib/iso.f90
  inflating:   MACOSX/at_2020_11_4/tslib/._iso.f90
```

```
1 cd at_2020_11_4/
```

```
/content/at_2020_11_4
```

```
1 !make clean all
```

```
rm -f bin/*.exe
find . -name '*.dSYM' -exec rm -r {} +
find . -name '*.png'  -exec rm -r {} +
find . -name '*.eps'  -exec rm -r {} +
find . -name '*.mod'  -exec rm -r {} +
find . -name '*.grn'  -exec rm -r {} +
find . -name '*.shd'  -exec rm -r {} +
find . -name '*.shd.mat'  -exec rm -r {} +
find . -name '*.prt'  -exec rm -r {} +
(cd misc;       make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/misc'
rm -f *.a *.mod *.o *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/misc'
(cd tslib;      make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/tslib'
rm -f *.a *.mod *.o *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/tslib'
(cd Bellhop;    make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/Bellhop'
rm -f *.o *.mod *.exe *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/Bellhop'
(cd Kraken;     make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/Kraken'
rm -f *.o *.mod *.exe *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/Kraken'
(cd KrakenField;        make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/KrakenField'
rm -f *.o *.mod *.exe *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/KrakenField'
(cd Krakel;     make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/Krakel'
```

```
rm -f *.o *.exe
make[1]: Leaving directory '/content/at_2020_11_4/Krakel'
(cd Scooter;    make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/Scooter'
rm -f *.o *.mod *.exe *_genmod.f90
make[1]: Leaving directory '/content/at_2020_11_4/Scooter'
(cd tests;      make -k -i clean)
make[1]: Entering directory '/content/at_2020_11_4/tests'
rm -f *.prt
rm -f *.mod
rm -f *.grn
rm -f *.shd
rm -f *.rts
rm -f *.ray
rm -f *.brc
rm -f *.irc
rm -f *.dat
rm -f *.asc
rm -f fort.*
rm -f *.moA
rm -f *.mat
(cd halfspace;  make clean)
make[2]: Entering directory '/content/at_2020_11_4/tests/halfspace'
rm -f *.prt
rm -f *.mod
rm -f *.grn
```

```
1 !make all
```

```
Routines needed by SCOOTER\SPARC\BELLHOP\BOUNCE built
make[1]: Leaving directory '/content/at_2020_11_4/misc'
(cd tslib;      make -k all)
make[1]: Entering directory '/content/at_2020_11_4/tslib'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/content/at_2020_11_4/tslib'
(cd Bellhop;    make -k all)
make[1]: Entering directory '/content/at_2020_11_4/Bellhop'
gfortran -o bellhop.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
gfortran -o bellhop3d.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic
Bellhop built
*************

make[1]: Leaving directory '/content/at_2020_11_4/Bellhop'
(cd Kraken;     make -k all)
make[1]: Entering directory '/content/at_2020_11_4/Kraken'
gfortran -o bounce.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sh
gfortran -o kraken.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sh
gfortran -o krakenc.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
KRAKEN built
************

make[1]: Leaving directory '/content/at_2020_11_4/Kraken'
(cd KrakenField;        make -k all)
make[1]: Entering directory '/content/at_2020_11_4/KrakenField'
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
ReadModes.f90:44:36:

   44 |   COMPLEX            :: PhiT( MaxN )
      |                                    1
Warning: Array ‘phit’ at (1) is larger than limit set by ‘-fmax-stack-var-size=’, moved from stack to static storage. This
ReadModes.f90:41:33:

   41 |   REAL               :: Z( MaxN ), W( Nrd ), Depth( MaxMedium ), rho( MaxMedium ), Tolerance, wT
      |                                 1
Warning: Array ‘z’ at (1) is larger than limit set by ‘-fmax-stack-var-size=’, moved from stack to static storage. This mak
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
gfortran -o field.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sha
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
gfortran -o field3d.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
KRAKEN field routines built
************
```

```
gfortran -o sparc.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sha
Scooter and Sparc built
**********************

make[1]: Leaving directory '/content/at_2020_11_4/Scooter'

**********************************
***** Acoustics Toolbox built *****
**********************************
```

```
1 !make install
```

```
gfortran -o bounce.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sh
gfortran -o kraken.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sh
gfortran -o krakenc.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
KRAKEN built
************

for f in bounce.exe kraken.exe krakenc.exe ; do \
        echo "----- Installing $f"; cp -p $f ../bin; \
done
----- Installing bounce.exe
----- Installing kraken.exe
----- Installing krakenc.exe
make[1]: Leaving directory '/content/at_2020_11_4/Kraken'
(cd KrakenField;        make -k install)
make[1]: Entering directory '/content/at_2020_11_4/KrakenField'
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
ReadModes.f90:44:36:

   44 |   COMPLEX            :: PhiT( MaxN )
      |                                    1
Warning: Array ‘phit’ at (1) is larger than limit set by ‘-fmax-stack-var-size=’, moved from stack to static storage. This
ReadModes.f90:41:33:

   41 |   REAL               :: Z( MaxN ), W( Nrd ), Depth( MaxMedium ), rho( MaxMedium ), Tolerance, wT
      |                                 1
Warning: Array ‘z’ at (1) is larger than limit set by ‘-fmax-stack-var-size=’, moved from stack to static storage. This mak
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
gfortran -o field.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sha
gfortran -c -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-shadow -Wline
gfortran -o field3d.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
KRAKEN field routines built
************

for f in field.exe field3d.exe ; do \
        echo "----- Installing $f"; cp -p $f ../bin; \
done
----- Installing field.exe
----- Installing field3d.exe
make[1]: Leaving directory '/content/at_2020_11_4/KrakenField'
# (cd Krakel;   make -k install)
(cd Scooter;    make -k install)
make[1]: Entering directory '/content/at_2020_11_4/Scooter'
gfortran -o scooter.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-s
gfortran -o sparc.exe -march=native -Bstatic -Waliasing -Wampersand -Wsurprising -Wintrinsics-std -Wno-tabs -Wintrinsic-sha
Scooter and Sparc built
**********************

for f in scooter.exe sparc.exe ; do \
        echo "----- Installing $f"; cp -p $f ../bin; \
done
----- Installing scooter.exe
----- Installing sparc.exe
make[1]: Leaving directory '/content/at_2020_11_4/Scooter'

**************************************
***** Acoustics Toolbox installed *****
**************************************
```

```
1 !echo $PATH
```

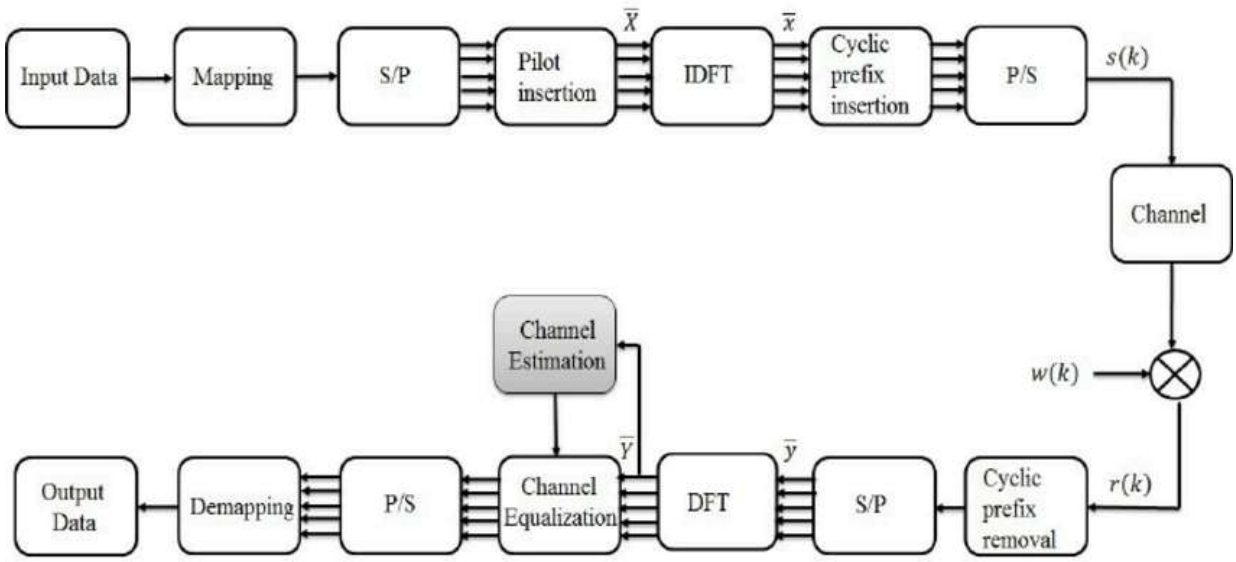
```
/opt/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/b
```

```
%env PATH=/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/b
```

```
env: PATH=/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/
```

```
!bellhop.exe
```

```
STOP Fatal Error: Check the print file for details
```



OFDM SYSTEM

```
from google.colab import files
f=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Screenshot (11).png to Screenshot (11).png
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import arlpy.uwapm as pm
import arlpy.plot as plt
import scipy as sp
```

```
spp=[
    [ 0, 1540],  # 1540 m/s at the surface
    [25, 1530],  # 1530 m/s at 10 m depth
     # 1532 m/s at 20 m depth
    [75, 1533],  # 1533 m/s at 25 m depth
    [100, 1537]   # 1535 m/s at the seabed
]
a=100*np.ones(10)-np.random.randint(10, size=10)
sea=100*np.arange(10)
bottom=[]
for i,j in zip(sea,a):
  bottom.append([i,j])
bottom.append([1000,100])

env = pm.create_env2d(
    depth=bottom,
    soundspeed=spp,
    bottom_soundspeed=1450,
    bottom_density=1900,
    bottom_absorption=0.8,
    tx_depth=40,
    rx_depth=40,
    rx_range=1000
)
surface =np.array([[r, 0.5+0.5*np.sin(2*np.pi*0.005*r)] for r in np.linspace(0,1000,1001)])
env['surface'] = surface
pm.print_env(env)
pm.plot_env(env, width=900)
pm.plot_ssp(env)
rays = pm.compute_eigenrays(env)
pm.plot_rays(rays, env=env, width=900)
arrivals = pm.compute_arrivals(env)
pm.plot_arrivals(arrivals, width=900)
```

```
                  name : arlpy
       bottom_absorption : 0.8
          bottom_density : 1900
        bottom_roughness : 0
       bottom_soundspeed : 1450
                   depth : [[   0.   97.]
                            [ 100.   96.]
                            [ 200.   96.]
                            [ 300.   98.]
                            [ 400.   92.]
                            [ 500.   92.]
                            [ 600.   91.]
                            [ 700.   92.]
                            [ 800.   97.]
                            [ 900.   99.]
                            [1000.  100.]]
            depth_interp : linear
               frequency : 25000
               max_angle : 80
               min_angle : -80
                  nbeams : 0
                rx_depth : 40
                rx_range : 1000
              soundspeed : [[   0. 1540.]
                            [  25. 1530.]
                            [  75. 1533.]
                            [ 100. 1537.]]
       soundspeed_interp : spline
                 surface : [[0.0000000e+00 5.0000000e-01]
                            [1.0000000e+00 5.1570538e-01]
                            [2.0000000e+00 5.3139526e-01]
                            ...
                            [9.9800000e+02 4.6860474e-01]
                            [9.9900000e+02 4.8429462e-01]
                            [1.0000000e+03 5.0000000e-01]]
          surface_interp : linear
                tx_depth : 40
       tx_directionality : None
                    type : 2D
```

FUNCTIONS.....

```
surface =np.array([[r, 0.5+0.5*np.sin(2*np.pi*0.005*r)] for r in np.linspace(0,1000,1001)])
env['surface'] = surface
pm.print_env(env)
pm.plot_env(env, width=900)
pm.plot_ssp(env)
rays = pm.compute_eigenrays(env)
pm.plot_rays(rays, env=env, width=900)
arrivals = pm.compute_arrivals(env)
pm.plot_arrivals(arrivals, width=900)
```

```
                 name : arlpy
       bottom_absorption : 0.8
          bottom_density : 1900
        bottom_roughness : 0
       bottom_soundspeed : 1450
                   depth : [[   0.   97.]
                            [ 100.   96.]
                            [ 200.   96.]
                            [ 300.   98.]
                            [ 400.   92.]
                            [ 500.   92.]
                            [ 600.   91.]
                            [ 700.   92.]
                            [ 800.   97.]
                            [ 900.   99.]
                            [1000.  100.]]
            depth_interp : linear
               frequency : 25000
               max_angle : 80
               min_angle : -80
                  nbeams : 0
                rx_depth : 40
                rx_range : 1000
              soundspeed : [[   0. 1540.]
                            [  25. 1530.]
                            [  75. 1533.]
                            [ 100. 1537.]]
       soundspeed_interp : spline
                 surface : [[0.0000000e+00 5.0000000e-01]
                            [1.0000000e+00 5.1570538e-01]
                            [2.0000000e+00 5.3139526e-01]
                            ...
                            [9.9800000e+02 4.6860474e-01]
                            [9.9900000e+02 4.8429462e-01]
                            [1.0000000e+03 5.0000000e-01]]
          surface_interp : linear
                tx_depth : 40
       tx_directionality : None
                    type : 2D
```

```
arrivals
```

	tx_depth_ndx	rx_depth_ndx	rx_range_ndx	tx_depth	rx_depth	rx_range	arrival_numb
1	0	0	0	40.0	40.0	1000.0	
2	0	0	0	40.0	40.0	1000.0	
3	0	0	0	40.0	40.0	1000.0	
4	0	0	0	40.0	40.0	1000.0	
5	0	0	0	40.0	40.0	1000.0	
6	0	0	0	40.0	40.0	1000.0	
7	0	0	0	40.0	40.0	1000.0	
8	0	0	0	40.0	40.0	1000.0	
9	0	0	0	40.0	40.0	1000.0	
10	0	0	0	40.0	40.0	1000.0	
11	0	0	0	40.0	40.0	1000.0	
12	0	0	0	40.0	40.0	1000.0	
13	0	0	0	40.0	40.0	1000.0	

```
def input_data(img_path):   ####### read the image from img_path for example "./ll.jp"
    img=cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img=cv2.resize(img,(1000,1024))
    img=np.asarray(img)
    return img
def encode(img):       ####### img is image return from input() function
    img2=np.reshape(img,[-1])
    img_en=[]
    for i in img2:
      s=bin(i)[2:].zfill(8)
      s=list(map(int,list(s)))
      img_en.append(s)
    return img_en

def QAM_64(input_encode):
    i=input_encode
    j=[0,0,0,0,0,0,0,0]
    input_data=np.reshape(i,[-1,6])
    mapping_table = {
    (0,0,0,0,0,0) : -7-7j,
    (0,0,0,0,0,1) : -7-5j,
    (0,0,0,0,1,0) : -7-1j,
    (0,0,0,0,1,1) : -7-3j,
    (0,0,0,1,0,0) : -7+7j,
    (0,0,0,1,0,1) : -7+5j,
    (0,0,0,1,1,0) : -7+1j,
    (0,0,0,1,1,1) : -7+3j,
    (0,0,1,0,0,0) : -5-7j,
    (0,0,1,0,0,1) : -5-5j,
    (0,0,1,0,1,0) : -5-1j,
    (0,0,1,0,1,1) : -5-3j,
    (0,0,1,1,0,0) : -5+7j,
    (0,0,1,1,0,1) : -5+5j,
    (0,0,1,1,1,0) : -5+1j,
    (0,0,1,1,1,1) : -5+3j,
    (0,1,0,0,0,0) : -1-7j,
    (0,1,0,0,0,1) : -1-5j,
    (0,1,0,0,1,0) : -1-1j,
    (0,1,0,0,1,1) : -1-3j,
    (0,1,0,1,0,0) : -1+7j,
    (0,1,0,1,0,1) : -1+5j,
    (0,1,0,1,1,0) : -1+1j,
    (0,1,0,1,1,1) : -1+3j,
    (0,1,1,0,0,0) : -3-7j,
    (0,1,1,0,0,1) : -3-5j,
    (0,1,1,0,1,0) : -3-1j,
    (0,1,1,0,1,1) : -3-3j,
    (0,1,1,1,0,0) : -3+7j,
    (0,1,1,1,0,1) : -3+5j,
    (0,1,1,1,1,0) : -3+1j,
    (0,1,1,1,1,1) : -3+3j,
    (1,0,0,0,0,0) :  7-7j,
    (1,0,0,0,0,1) :  7-5j,
    (1,0,0,0,1,0) :  7-1j,
    (1,0,0,0,1,1) :  7-3j,
    (1,0,0,1,0,0) :  7+7j,
    (1,0,0,1,0,1) :  7+5j,
    (1,0,0,1,1,0) :  7+1j,
    (1,0,0,1,1,1) :  7+3j,
    (1,0,1,0,0,0) :  5-7j,
    (1,0,1,0,0,1) :  5-5j,
    (1,0,1,0,1,0) :  5-1j,
    (1,0,1,0,1,1) :  5-3j,
    (1,0,1,1,0,0) :  5+7j,
    (1,0,1,1,0,1) :  5+5j,
    (1,0,1,1,1,0) :  5+1j,
    (1,0,1,1,1,1) :  5+3j,
    (1,1,0,0,0,0) :  1-7j,
    (1,1,0,0,0,1) :  1-5j,
    (1,1,0,0,1,0) :  1-1j,
```

```
    (1,1,0,0,1,1) :  1-3j,
    (1,1,0,1,0,0) :  1+7j,
    (1,1,0,1,0,1) :  1+5j,
    (1,1,0,1,1,0) :  1+1j,
    (1,1,0,1,1,1) :  1+3j,
    (1,1,1,0,0,0) :  3-7j,
    (1,1,1,0,0,1) :  3-5j,
    (1,1,1,0,1,0) :  3-1j,
    (1,1,1,0,1,1) :  3-3j,
    (1,1,1,1,0,0) :  3+7j,
    (1,1,1,1,0,1) :  3+5j,
    (1,1,1,1,1,0) :  3+1j,
    (1,1,1,1,1,1) :  3+3j

    }
    OFDM_input=np.array([mapping_table[tuple(b)] for b in input_data])
    demapping_table = {v : k for k, v in mapping_table.items()}
    return OFDM_input,demapping_table
def QAM_32(input_encode):
    i=input_encode
    j=[0,0,0,0,0,0,0,0]
    input_data=np.reshape(i,[-1,5])
    mapping_table = {
    (0,0,0,0,0) : -3+5j,
    (0,0,0,0,1) : -5-1j,
    (0,0,0,1,0) :  3+3j,
    (0,0,0,1,1) : -1-3j,
    (0,0,1,0,0) : -5+3j,
    (0,0,1,0,1) :  3-1j,
    (0,0,1,1,0) : -1+1j,
    (0,0,1,1,1) : -3-5j,
    (0,1,0,0,0) :  1+5j,
    (0,1,0,0,1) : -1-1j,
    (0,1,0,1,0) :  1-5j,
    (0,1,0,1,1) :  3-3j,
    (0,1,1,0,0) : -1+3j,
    (0,1,1,0,1) : -5-3j,
    (0,1,1,1,0) :  3+1j,
    (0,1,1,1,1) :  1-5j,
    (1,0,0,0,0) : -1+5j,
    (1,0,0,0,1) : -3-1j,
    (1,0,0,1,0) :  3+5j,
    (1,0,0,1,1) :  1-3j,
    (1,0,1,0,0) : -3+3j,
    (1,0,1,0,1) :  5-1j,
    (1,0,1,1,0) :  1+1j,
    (1,0,1,1,1) : -1-5j,
    (1,1,0,0,0) :  3+5j,
    (1,1,0,0,1) :  1-1j,
    (1,1,0,1,0) : -3+1j,
    (1,1,0,1,1) :  5-3j,
    (1,1,1,0,0) :  1+3j,
    (1,1,1,0,1) : -3-3j,
    (1,1,1,1,0) :  5+1j,
    (1,1,1,1,1) :  3-5j
    }
    OFDM_input=np.array([mapping_table[tuple(b)] for b in input_data])
    demapping_table = {v : k for k, v in mapping_table.items()}
    return OFDM_input,demapping_table
def QAM_16(input_encode):
    input_data=np.reshape(input_encode,[-1,4])
    mapping_table = {
    (0,0,0,0) : -3-3j,
    (0,0,0,1) : -3-1j,
    (0,0,1,0) : -3+3j,
    (0,0,1,1) : -3+1j,
    (0,1,0,0) : -1-3j,
    (0,1,0,1) : -1-1j,
    (0,1,1,0) : -1+3j,
    (0,1,1,1) : -1+1j,
    (1,0,0,0) :  3-3j,
```

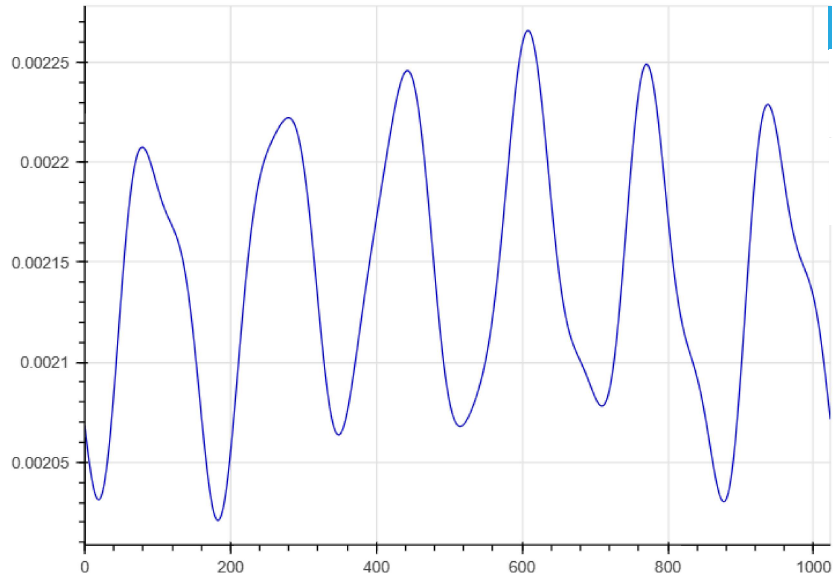
```
    (1,0,0,1) :  3-1j,
    (1,0,1,0) :  3+3j,
    (1,0,1,1) :  3+1j,
    (1,1,0,0) :  1-3j,
    (1,1,0,1) :  1-1j,
    (1,1,1,0) :  1+3j,
    (1,1,1,1) :  1+1j
    }
    OFDM_input=np.array([mapping_table[tuple(b)] for b in input_data])
    demapping_table = {v : k for k, v in mapping_table.items()}
    return OFDM_input,demapping_table
def modulation(en_data,mod):
    if mod == 'qam16':
      mod_data,dmap=QAM_16(en_data)
    if mod == 'qam32':
      mod_data,dmap=QAM_32(en_data)
    if mod == 'qam64':
      mod_data,dmap=QAM_32(en_data)
    return mod_data,dmap
def s2p(mod_data):
    mod_data=np.reshape(mod_data,(-1,1024))
    return mod_data
def pilot_insertion(pal_data):
    pilot_value=3-3j
    pilot_sig=np.ones(np.shape(pal_data),dtype=complex)*pilot_value
    pin_sig=[]
    for i,j in zip(pilot_sig,pal_data):
      pin_sig.append(i)
      pin_sig.append(j)
    return pin_sig
def IDFT(pin_sig):
    return np.fft.ifft(pin_sig)
def cyclic_pre(ifft_sig):
    s=ifft_sig
    ofdm_CP_sig=[]
    for i in s:
      l=np.hstack([i[:256],i])
      ofdm_CP_sig.append(l)
    return ofdm_CP_sig
def p2s(cp_sg):
    return cp_sg
def rm_cyclic_pre(rx_sig):
    rx_rmcp=[]
    for i in rx_sig:
      rx_rmcp.append(i[256:])
    return rx_rmcp
def s2p1(rx_sig):
    return rx_sig
def dft(rx_sp):
    return np.fft.fft(rx_sp)
def chann_est_ls(ofdm_fft,pilot_value):
    ls_chann_est=ofdm_fft/pilot_value
    return ls_chann_est
def corr_matrix(x,y):
    Rhh=np.correlate(x,y,mode='full')
    Rhh=Rhh[len(Rhh)//2:]
    rhh = sp.linalg.toeplitz(Rhh,np.hstack((Rhh[0], np.conj(Rhh[1:]))))
    return rhh
def noise_var(SNRdb):
    return np.power(10,0.1*SNRdb)/2
def chann_est_mmse(OFDM_demod,Rhh,noise_var,pilot_value,imp_res):
    n=np.shape(Rhh)[0]
    hest=[]
    res=chann_est_ls(OFDM_demod,pilot_value)
    a=np.matmul(corr_matrix(imp_res,res),np.linalg.inv(Rhh +(1/(noise_var))*np.identity(1024)))
    hest=np.matmul(a,res)
    return hest

def equalize(OFDM_demod, Hest):
    return OFDM_demod / Hest
def Demapping(QAM,demapping_table):
```

```
    # array of possible constellation points
    constellation = np.array([x for x in demapping_table.keys()])

    # calculate distance of each RX point to each possible point
    dists = abs(QAM.reshape((-1,1)) - constellation.reshape((1,-1)))

    # for each element in QAM, choose the index in constellation
    # that belongs to the nearest constellation point
    const_index = dists.argmin(axis=1)

    # get back the real constellation point
    hardDecision = constellation[const_index]

    # transform the constellation point into the bit groups
    return np.vstack([demapping_table[C] for C in hardDecision]), hardDecision
def bit_dec(rx_est):
    a=[]
    for i in rx_est:
      n=7
      x=0
      for j in i:
        x=x+j*pow(2,n)
        n=n-1
      a.append(x)
    return a
import scipy
def chann_est(rx_pilot_sig,pilot_value,imp_res,tx_sig1,noise):
  noise_v=np.var(tx_sig1)/np.var(noise[0])
  chann_ls=rx_pilot_sig/pilot_value
  rhh=corr_matrix(imp_res,imp_res)
  x=np.linalg.inv(rhh +(1/(noise_v))*np.identity(1024))
  chann_est=np.matmul(chann_ls,np.matmul(rhh,x))
  return chann_est,chann_ls
```

```
def tx(im_path,mod):
# img=input_data("")./lenna.jpg
# img_en=encode(img)
# mod_data,dmap=modulation(img_en,'qam16')
# pal_data=s2p(mod_data)
# pin_sig=pilot_insertion(pal_data)
# ifft_sig=IDFT(pin_sig)
# cp_sig=cyclic_pre(ifft_sig)
# tx_sig=p2s(cp_sig)
    img=input_data(im_path)
    img_en=encode(img)
    mod_data,dmap=modulation(img_en,mod)
    pal_data=s2p(mod_data)
    pin_sig=pilot_insertion(pal_data)
    ifft_sig=IDFT(pin_sig)
    cp_sig=cyclic_pre(ifft_sig)
    tx_sig=p2s(cp_sig)
    return tx_sig,dmap,img_en,pal_data

def channel_response(chann_res):
    amp=chann_res["arrival_amplitude"]
    amp=amp.to_numpy()
    time=chann_res["time_of_arrival"]
    time=time.to_numpy()
    time=np.round(time,3)*100
    imp_res=np.zeros(1024,dtype=complex)
    for i,j in zip(amp,time):
      j=int(j)
      imp_res[j]=imp_res[j]+i
    return imp_res
def channel(tx_sig,imp_res,SNRdb):
    rx_sig=[]
    rx_noise=[]
    rx_pow=[]
    noise_pow=[]
    for i in tx_sig:
      convolved = np.convolve(i, imp_res)
      signal_power = np.mean(abs(convolved**2))
      sigma2 = signal_power * 10**(-SNRdb/20)  # calculate noise power based on signal power and SNR
                                               # print ("RX Signal power: %.4f. Noise power: %.4f" % (signal_power, sigma2))
                                               # Generate complex noise with given variance
      noise = np.sqrt(sigma2/2) * (np.random.randn(*convolved.shape)+1j*np.random.randn(*convolved.shape))
      rx_pow.append(signal_power)
      noise_pow.append(sigma2)
      rx_sig.append((convolved+noise)[65:1280+65])
      rx_noise.append(noise)
    return rx_sig,rx_noise,rx_pow,noise_pow
```

```
imp_res=channel_response(arrivals)
```

```
a=np.fft.fft(imp_res,1024)
plt.plot(abs(a))

```




```
1 tx_sig1,dmap,img_en,mode_data=tx("./lenna.jpg",'qam16')
```

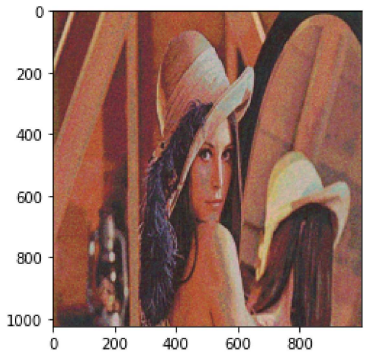
```
1 rx_sig,noise,rx_pow,noise_pw=channel(tx_sig1,imp_res,15)
```

```
1 noise_pw
```

```
1.2765658719831303e-08,
4.6848183467492375e-09,
1.2765658719831303e-08,
4.246101267044431e-09,
1.2765658719831303e-08,
4.670330517994471e-09,
1.2765658719831303e-08,
4.487890043568216e-09,
1.2765658719831303e-08,
6.0568896549915995e-09,
1.2765658719831303e-08,
4.434433784466875e-09,
1.2765658719831303e-08,
```

```
rx_rmcp=rm_cyclic_pre(rx_sig)
rx_sp=s2p1(rx_rmcp)
rx_dft=dft(rx_sp)
rx_pilot_sig=[]
rx_resig=[]
for i in range(int(np.shape(rx_dft)[0]/2)):
  rx_pilot_sig.append(rx_dft[2*i])
  rx_resig.append(rx_dft[2*i+1])
pilot_value=3-3j
rhh=corr_matrix(imp_res,imp_res)
n_var=noise_var(30)
rx_pilot_sig=np.asarray(rx_pilot_sig)
chann_mmse,chann_ls=chann_est(rx_pilot_sig,pilot_value,imp_res,tx_sig1,noise)
rx_demod=equalize(rx_resig,chann_mmse)
rx_est, hardDecision = Demapping(rx_demod,dmap)
rx_est=np.reshape(rx_est,(-1,8))
print("bit error rate   :   "+ str(np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est,(-1))))))
nmse=np.sum(np.power(abs(a-chann_mmse[0]),2))/np.sum(np.power(abs(a),2))/1024
print(nmse)
x=bit_dec(rx_est)
x=np.reshape(x,(1024,1000,3))
import matplotlib.pyplot as plt1
plt1.imshow(x)
```

```
bit error rate   :   0.13067647298177085
0.0018232062084376363
<matplotlib.image.AxesImage at 0x7fbd8c156130>
```




```
rx_pilot_sig=np.asarray(np.reshape(rx_pilot_sig,(-1,1)))
```

```
with open('imp_res.npy', 'wb') as f:
  np.save(f,imp_res)
files.download("imp_res.npy")
train=[]
for i in rx_pilot_sig:
  x=[]
  for j in i:
    y=[]
    y.append(np.real(j))
    y.append(np.imag(j))
    x.append(y)
  train.append(x)

truth=[]
for i in chann_mmse:
  x=[]
  for j in i:
    y=[]
    y.append(np.real(j))
    y.append(np.imag(j))
    x.append(y)
  truth.append(x)
train=np.asarray(train,dtype='float32')
truth=np.asarray(truth,dtype='float32')
train=np.reshape(train,(-1,4))
truth=np.reshape(truth,(-1,4))
from tensorflow.python.keras import *
from tensorflow.python.keras.layers import *
import tensorflow as tf
from tensorflow.keras.layers import Dense

x = Input(shape=(4))
y1 = Dense(32, activation='tanh')(x)
y2 = Dense(32, activation='tanh')(y1)
y3 = Dense(32, activation='tanh')(y2)
output = Dense(4, activation='linear')(y3)
model = tf.keras.Model(inputs=x, outputs=output)
model.compile(optimizer='adam',loss='mae', metrics=["accuracy"])
model.summary()
model.fit(train,truth,epochs=20,batch_size=64)
```

```
files.download("imp_res.npy")
```

```
train=[]
for i in rx_pilot_sig:
  x=[]
  for j in i:
    y=[]
    y.append(np.real(j))
    y.append(np.imag(j))
    x.append(y)
  train.append(x)
```

```
truth=[]
for i in chann_mmse:
  x=[]
  for j in i:
    y=[]
    y.append(np.real(j))
    y.append(np.imag(j))
    x.append(y)
  truth.append(x)
```

```
train=np.asarray(train,dtype='float32')
truth=np.asarray(truth,dtype='float32')
train=np.reshape(train,(-1,4))
truth=np.reshape(truth,(-1,4))
```

DNN1

```
from tensorflow.python.keras import *
from tensorflow.python.keras.layers import *
import tensorflow as tf
from tensorflow.keras.layers import Dense

x = Input(shape=(4))
y1 = Dense(32, activation='tanh')(x)
y2 = Dense(32, activation='tanh')(y1)
y3 = Dense(32, activation='tanh')(y2)
output = Dense(4, activation='linear')(y3)
model = tf.keras.Model(inputs=x, outputs=output)
model.compile(optimizer='adam',loss='mae', metrics=["accuracy"])
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 4)]               0
_________________________________________________________________
dense (Dense)                (None, 32)                160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 132
=================================================================
Total params: 2,404
Trainable params: 2,404
Non-trainable params: 0
_________________________________________________________________
```

```
np.shape(train)
```

```
(3072000, 4)
```

```
model.fit(train,truth,epochs=20,batch_size=64)
```

```
Epoch 1/20
48000/48000 [==============================] - 107s 2ms/step - loss: 1.8726e-04 - accuracy: 0.7791
Epoch 2/20
48000/48000 [==============================] - 104s 2ms/step - loss: 9.5738e-05 - accuracy: 0.8531
Epoch 3/20
48000/48000 [==============================] - 103s 2ms/step - loss: 9.2720e-05 - accuracy: 0.8574
Epoch 4/20
48000/48000 [==============================] - 103s 2ms/step - loss: 9.0995e-05 - accuracy: 0.8602
Epoch 5/20
48000/48000 [==============================] - 102s 2ms/step - loss: 9.0617e-05 - accuracy: 0.8597
Epoch 6/20
48000/48000 [==============================] - 102s 2ms/step - loss: 9.0468e-05 - accuracy: 0.8619
Epoch 7/20
48000/48000 [==============================] - 102s 2ms/step - loss: 8.9325e-05 - accuracy: 0.8618
Epoch 8/20
48000/48000 [==============================] - 101s 2ms/step - loss: 9.0995e-05 - accuracy: 0.8608
Epoch 9/20
48000/48000 [==============================] - 101s 2ms/step - loss: 8.9240e-05 - accuracy: 0.8624
Epoch 10/20
48000/48000 [==============================] - 101s 2ms/step - loss: 8.8979e-05 - accuracy: 0.8636
Epoch 11/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.9473e-05 - accuracy: 0.8656
Epoch 12/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.8519e-05 - accuracy: 0.8663
Epoch 13/20
48000/48000 [==============================] - 100s 2ms/step - loss: 9.0021e-05 - accuracy: 0.8601
```

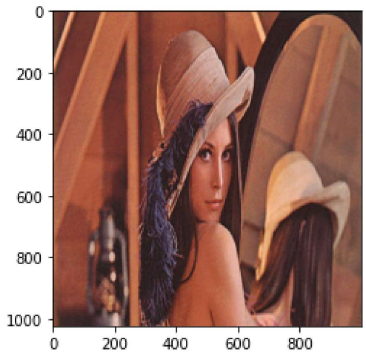
```
Epoch 14/20
48000/48000 [==============================] - 102s 2ms/step - loss: 8.8798e-05 - accuracy: 0.8672
Epoch 15/20
48000/48000 [==============================] - 103s 2ms/step - loss: 8.8590e-05 - accuracy: 0.8659
Epoch 16/20
48000/48000 [==============================] - 102s 2ms/step - loss: 8.8504e-05 - accuracy: 0.8618
Epoch 17/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.8459e-05 - accuracy: 0.8653
Epoch 18/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.8515e-05 - accuracy: 0.8658
Epoch 19/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.9060e-05 - accuracy: 0.8642
Epoch 20/20
48000/48000 [==============================] - 100s 2ms/step - loss: 8.7967e-05 - accuracy: 0.8677
<tensorflow.python.keras.callbacks.History at 0x7f49f287f390>
```

```
pred=model.predict(train,batch_size=64)
```

```
pred=model.predict(train,batch_size=64)
pred=np.reshape(pred,(-1,2))
pred_imp=[]
for i in pred:
  # print(i)
  pred_imp.append(complex(i[0],i[1]))
pred_imp=np.reshape(pred_imp,(-1,1024))
rx_demod=equalize(rx_resig,pred_imp)
rx_est, hardDecision = Demapping(rx_demod,dmap)
rx_est=np.reshape(rx_est,(-1,8))
print("bit error rate   :   "+ str(np.sum(abs(np.reshape(img_en,(-1))-np.reshape(rx_est,(-1))))))
x=bit_dec(rx_est)
x=np.reshape(x,(1024,1000,3))
import matplotlib.pyplot as plt1
plt1.imshow(x)'
model.save('qam16_15db_dnn1_model.h5')
files.download("qam16_15db_dnn1_model.h5")
from tensorflow.python.keras import *
from tensorflow.python.keras.layers import *
import tensorflow as tf
from tensorflow.keras.layers import Dense

x = Input(shape=(16))
y1 = Dense(64, activation='tanh')(x)
y2 = Dense(64, activation='tanh')(y1)
y3 = Dense(64, activation='tanh')(y2)
output = Dense(16, activation='linear')(y3)
model1 = tf.keras.Model(inputs=x, outputs=output)
model1.compile(optimizer='adam',loss='mae', metrics=["accuracy"])
model1.summary()
train1=np.reshape(train,(-1,16))
truth1=np.reshape(truth,(-1,16))
model1.fit(train1,truth1,epochs=30,batch_size=64)
pred=model1.predict(train1,batch_size=64)
pred=np.reshape(pred,(-1,2))
pred_imp1=[]
for i in pred:
  # print(i)
  pred_imp1.append(complex(i[0],i[1]))
pred_imp1=np.reshape(pred_imp,(-1,1024))
rx_demod=equalize(rx_resig,pred_imp1)
rx_est, hardDecision = Demapping(rx_demod,dmap)
rx_est=np.reshape(rx_est,(-1,8))
print("bit error rate   :   "+ str(np.sum(abs(np.reshape(img_en,(-1))-np.reshape(rx_est,(-1))))))
x=bit_dec(rx_est)
x=np.reshape(x,(1024,1000,3))
import matplotlib.pyplot as plt1
plt1.imshow(x)
```

```
rx_demod=equalize(rx_resig,pred_imp)
rx_est, hardDecision = Demapping(rx_demod,dmap)
rx_est=np.reshape(rx_est,(-1,8))
print("bit error rate   :   "+ str(np.sum(abs(np.reshape(img_en,(-1))-np.reshape(rx_est,(-1))))))
x=bit_dec(rx_est)
x=np.reshape(x,(1024,1000,3))
import matplotlib.pyplot as plt1
plt1.imshow(x)
```

```
bit error rate   :   308833
<matplotlib.image.AxesImage at 0x7f49d5922ed0>
```




```
model.save('qam16_15db_dnn1_model.h5')
```

```
ls
```

```
assets/            Krakel/        Matlab/                   'RAM off oalib'/
at_init_matlab.m   Kraken/        misc/                      saved_model.pb
Bellhop/           KrakenField/   qam16_15db_dnn1_model/     Scooter/
bin*               lenna.jpg      qam16_15db_dnn1_model.h5   tests/
doc/               LICENSE        RAM/                       tslib/
index.htm*         Makefile       RAM_ljh/                   variables/
```

```
files.download("qam16_15db_dnn1_model.h5")
```

DNN2

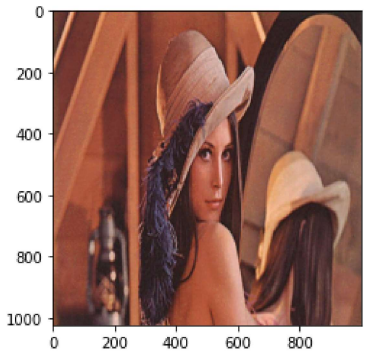
```
from tensorflow.python.keras import *
from tensorflow.python.keras.layers import *
import tensorflow as tf
from tensorflow.keras.layers import Dense

x = Input(shape=(16))
y1 = Dense(64, activation='tanh')(x)
y2 = Dense(64, activation='tanh')(y1)
y3 = Dense(64, activation='tanh')(y2)
output = Dense(16, activation='linear')(y3)
model1 = tf.keras.Model(inputs=x, outputs=output)
model1.compile(optimizer='adam',loss='mae', metrics=["accuracy"])
model1.summary()
train1=np.reshape(train,(-1,16))
truth1=np.reshape(truth,(-1,16))
model1.fit(train1,truth1,epochs=30,batch_size=64)
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 16)]              0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                1088
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
```

```
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 16)                1040
=================================================================
Total params: 10,448
Trainable params: 10,448
Non-trainable params: 0
_________________________________________________________________
Epoch 1/30
12000/12000 [==============================] - 27s 2ms/step - loss: 2.4580e-04 - accuracy: 0.4466
Epoch 2/30
12000/12000 [==============================] - 26s 2ms/step - loss: 1.0450e-04 - accuracy: 0.6176
Epoch 3/30
12000/12000 [==============================] - 26s 2ms/step - loss: 9.6278e-05 - accuracy: 0.6495
Epoch 4/30
12000/12000 [==============================] - 26s 2ms/step - loss: 9.4037e-05 - accuracy: 0.6619
Epoch 5/30
12000/12000 [==============================] - 26s 2ms/step - loss: 9.1141e-05 - accuracy: 0.6649
Epoch 6/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.8835e-05 - accuracy: 0.6746
Epoch 7/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.7696e-05 - accuracy: 0.6796
Epoch 8/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.6941e-05 - accuracy: 0.6803
Epoch 9/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.6784e-05 - accuracy: 0.6822
Epoch 10/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.6586e-05 - accuracy: 0.6809
Epoch 11/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.5578e-05 - accuracy: 0.6860
Epoch 12/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.4634e-05 - accuracy: 0.6898
Epoch 13/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.4954e-05 - accuracy: 0.6899
Epoch 14/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.3339e-05 - accuracy: 0.6979
Epoch 15/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.3131e-05 - accuracy: 0.6943
Epoch 16/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.2822e-05 - accuracy: 0.6944
Epoch 17/30
12000/12000 [==============================] - 25s 2ms/step - loss: 8.3183e-05 - accuracy: 0.6959
Epoch 18/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.2093e-05 - accuracy: 0.7008
Epoch 19/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.3002e-05 - accuracy: 0.6968
Epoch 20/30
12000/12000 [==============================] - 26s 2ms/step - loss: 8.3246e-05 - accuracy: 0.6963
```

```
pred=model1.predict(train1,batch_size=64)
pred=np.reshape(pred,(-1,2))
pred_imp1=[]
for i in pred:
  # print(i)
  pred_imp1.append(complex(i[0],i[1]))
pred_imp1=np.reshape(pred_imp,(-1,1024))
rx_demod=equalize(rx_resig,pred_imp1)
rx_est, hardDecision = Demapping(rx_demod,dmap)
rx_est=np.reshape(rx_est,(-1,8))
print("bit error rate   :   "+ str(np.sum(abs(np.reshape(img_en,(-1))-np.reshape(rx_est,(-1))))))
x=bit_dec(rx_est)
x=np.reshape(x,(1024,1000,3))
import matplotlib.pyplot as plt1
plt1.imshow(x)
```

```
bit error rate   :   308833
<matplotlib.image.AxesImage at 0x7f49d2cd3850>
```




```
model1.save('qam16_15db_dnn2_model.h5')
```

```
files.download('qam16_15db_dnn2_model.h5')
```

```
def model_chann(model,rx_pilot_sig):
  train=[]
  for i in rx_pilot_sig:
    x=[]
    for j in i:
      y=[]
      y.append(np.real(j))
      y.append(np.imag(j))
      x.append(y)
    train.append(x)
  train=np.asarray(train,dtype='float32')
  train=np.reshape(train,(-1,4))
  pred=model.predict(train,batch_size=64)
  pred=np.reshape(pred,(-1,2))
  pred_imp=[]
  for i in pred:
  # print(i)
    pred_imp.append(complex(i[0],i[1]))
  pred_imp=np.reshape(pred_imp,(-1,1024))
  return pred_imp
def model1_chann(model,rx_pilot_sig):
  train=[]
  for i in rx_pilot_sig:
    x=[]
    for j in i:
      y=[]
      y.append(np.real(j))
      y.append(np.imag(j))
      x.append(y)
    train.append(x)
  train=np.asarray(train,dtype='float32')
  train=np.reshape(train,(-1,16))
  pred=model.predict(train,batch_size=64)
  pred=np.reshape(pred,(-1,2))
  pred_imp=[]
  for i in pred:
  # print(i)
    pred_imp.append(complex(i[0],i[1]))
  pred_imp=np.reshape(pred_imp,(-1,1024))
  return pred_imp
```

```
snr=[5,10,15,20,25]
result_mmse=[]
result_ls=[]
# result_model=[]
# result_model1=[]
mmse_ber=[]
ls_ber=[]
# model_ber=[]
# model1_ber=[]
for i in snr:
  rx_sig,noise,rx_pow,noise_pw=channel(tx_sig1,imp_res,i)
  rx_rmcp=rm_cyclic_pre(rx_sig)
  rx_sp=s2p1(rx_rmcp)
  rx_dft=dft(rx_sp)
  rx_pilot_sig=[]
  rx_resig=[]
  for i in range(int(np.shape(rx_dft)[0]/2)):
    rx_pilot_sig.append(rx_dft[2*i])
    rx_resig.append(rx_dft[2*i+1])
  pilot_value=3-3j
  rhh=corr_matrix(imp_res,imp_res)
  n_var=noise_var(30)
  rx_pilot_sig=np.asarray(rx_pilot_sig)

  pred_imp=model_chann(model,rx_pilot_sig)
  pred_imp1=model1_chann(model1,rx_pilot_sig)
  chann_est_mmse,chann_ls=chann_est(rx_pilot_sig,pilot_value,imp_res,tx_sig1,noise)

  # rx_demod_m=equalize(rx_resig,pred_imp)
  # rx_est_m, hardDecision = Demapping(rx_demod_m,dmap)
  # rx_est_m=np.reshape(rx_est_m,(-1,8))
  # ber_m=np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_m,(-1))))

  # rx_demod_m1=equalize(rx_resig,pred_imp1)
  # rx_est_m1, hardDecision = Demapping(rx_demod_m1,dmap)
  # rx_est_m1=np.reshape(rx_est_m1,(-1,8))
  # ber_m1=np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_m1,(-1))))

  rx_demod_mmse=equalize(rx_resig,chann_est_mmse)
  rx_est_mmse, hardDecision = Demapping(rx_demod_mmse,dmap)
  rx_est_mmse=np.reshape(rx_est_mmse,(-1,8))
  ber_mmse=np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_mmse,(-1))))

  rx_demod_ls=equalize(rx_resig,chann_ls)
  rx_est_ls, hardDecision = Demapping(rx_demod_ls,dmap)
  rx_est_ls=np.reshape(rx_est_ls,(-1,8))
  ber_ls=np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_ls,(-1))))

  print("bit error rate mmse  :   "+ str(np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_mmse,(-1))))))
  print("bit error rate ls  :   "+ str(np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_ls,(-1))))))
  # print("bit error rate model :   "+ str(np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_m,(-1))))))
  # print("bit error rate model1 :   "+ str(np.mean(abs(np.reshape(img_en,(-1))-np.reshape(rx_est_m1,(-1))))))
  # nmse=np.sum(np.power(abs(a-chann_est_mmse[0]),2))/np.sum(np.power(abs(a),2))/1024
  # print(nmse)
  x=bit_dec(rx_est_mmse)
  x=np.reshape(x,(1024,1000,3))
  x_ls=bit_dec(rx_est_ls)
  x_ls=np.reshape(x_ls,(1024,1000,3))
  # x_m=bit_dec(rx_est_m)
  # x_m=np.reshape(x_m,(1024,1000,3))
  # x_m1=bit_dec(rx_est_m1)
  # x_m1=np.reshape(x_m1,(1024,1000,3))
  result_mmse.append(x)
  result_ls.append(x_ls)
  # result_model.append(x_m)
  # result_model1.append(x_m1)
  mmse_ber.append(ber_mmse)
  ls_ber.append(ber_ls)
  # model_ber.append(ber_m)
  # model1_ber.append(ber_m1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-38-a1f8116ac3fa> in <module>
     23   rx_pilot_sig=np.asarray(rx_pilot_sig)
     24
---> 25   pred_imp=model_chann(model,rx_pilot_sig)
     26   pred_imp1=model1_chann(model1,rx_pilot_sig)
```

```
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(15, 8))
columns = 5
rows = 1
for i in range(1, columns*rows +1):
    img = result_mmse[i-1]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
plt.show()
fig=plt.figure(figsize=(15, 8))
columns = 5
rows = 1
for i in range(1, columns*rows +1):
    img = result_ls[i-1]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
plt.show()
fig=plt.figure(figsize=(15, 8))
columns = 5
rows = 1
for i in range(1, columns*rows +1):
    img = result_model[i-1]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
plt.show()
fig=plt.figure(figsize=(15, 8))
columns = 5
rows = 1
for i in range(1, columns*rows +1):
    img = result_model1[i-1]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
plt.show()
```