`network` module has the following Class objects

Network

This is the core object of the entire package. This object can be either defined using ABCD parameters or Scattering paramters. If the object is defined using ABCD then S-parameters are computed. If the object is defined using S-parameters, then ABCD are computed. Default assumption is that ABCD parameters are given. In the current implementation a symmetric and real port impedance is assumed. Default value of the symmetric port impedance is Z0=50 ohm.

> Network(p11, p12, p21, p22, parameter='abcd',Z0=50,omega=np.NaN):

Attributes:

ABCD parameters

- A
- B
- C
- D

Scattering parameters

- S11
- S12
- S21
- S22

`network` module has the following functions implemented.

from_Tx_line(l,Z0,gamma,omega=np.NaN):

> network.from_Tx_line(l,Z0,gamma,omega)

```
This function takes the following input paramters,
l = length of the transmission line. Should be a scalar.
Z0 = Characteristics impedance of the transmission line. Should be a scalar or
```

```
an array of the same dimensions as beta.
gamma = propagation constant of the transmission line. Should be the an array of
the same dimensions as omega.
The function returns the ABCD parameters of a transmission line section of
length l.
o-------------------------------------o
        Z0, \gamma = \alpha + j\beta
o-------------------------------------o
|<-------------- l ------------------->|
Based on equations on Page-185, 'Microwave engineering' by D. M. Pozar
```

from_series_Z(Z):

> network.from_series_z(Z)

```
Returns a network object for the following
o-----------------|Z|--------------------o
                   [ABCD]
o-----------------------------------------o
Based on equations on Page-185, 'Microwave engineering' by D. M. Pozar
"""
A = np.ones_like(Z)
B = Z
C = np.zeros_like(Z)
D = np.ones_like(Z)
return Network(A, B, C, D,parameter='abcd')
```

from_shunt_Y(Y):

> network.from_shunt_Y(Y)

```
""" Returns a network object for the following
o-------------------------------------o
                  _|_
                 |_Y_|
                   |
o-------------------------------------o
Based on equations on Page-185, 'Microwave engineering' by D. M. Pozar
"""
```

```
A = np.ones_like(Y)
B = np.zeros_like(Y)
C = Y
D = np.ones_like(Y)
return Network(A, B, C, D,parameter='abcd')
```

from_PI_Y(Y1,Y2,Y3,omega=np.nan):

> network.from_PI_Y(Y1,Y2,Y3,omega=np.nan)

```
""" Returns a network object for the following
                                  ____
o----------------|_Y3_|---------------------o
                      _|__         _|__
                     |_Y1_|       |_Y2_|
                          |           |
o-----------------------------------------o
Based on equations on Page-185, 'Microwave engineering' by D. M. Pozar
"""
A = 1 + Y2/Y3
B = 1/Y3
C = Y1 + Y2 + Y1*Y2/Y3
D = 1 + Y1/Y3
return Network(A, B, C, D,parameter='abcd',omega=omega)
```

from_T_Z(Z1,Z2,Z3):

> network.from_T_Z(Z1,Z2,Z3):

```
""" Returns a network object for the following
                        ____           ____
o----------|_Z1_|----------|_Z2_|-------o
                                     __|__
                                    |_Z3_|
                                       |
o-------------------------------------o
Based on equations on Page-185, 'Microwave engineering' by D. M. Pozar
"""
A = 1 + Z1/Z3
```

```
B = Z1 + Z2 + Z1*Z2/Z3
C = 1/Z3
D = 1 + Z2/Z3
return Network(A, B, C, D,parameter='abcd')
```