RF Topologies

Robert McMahon

June 2018

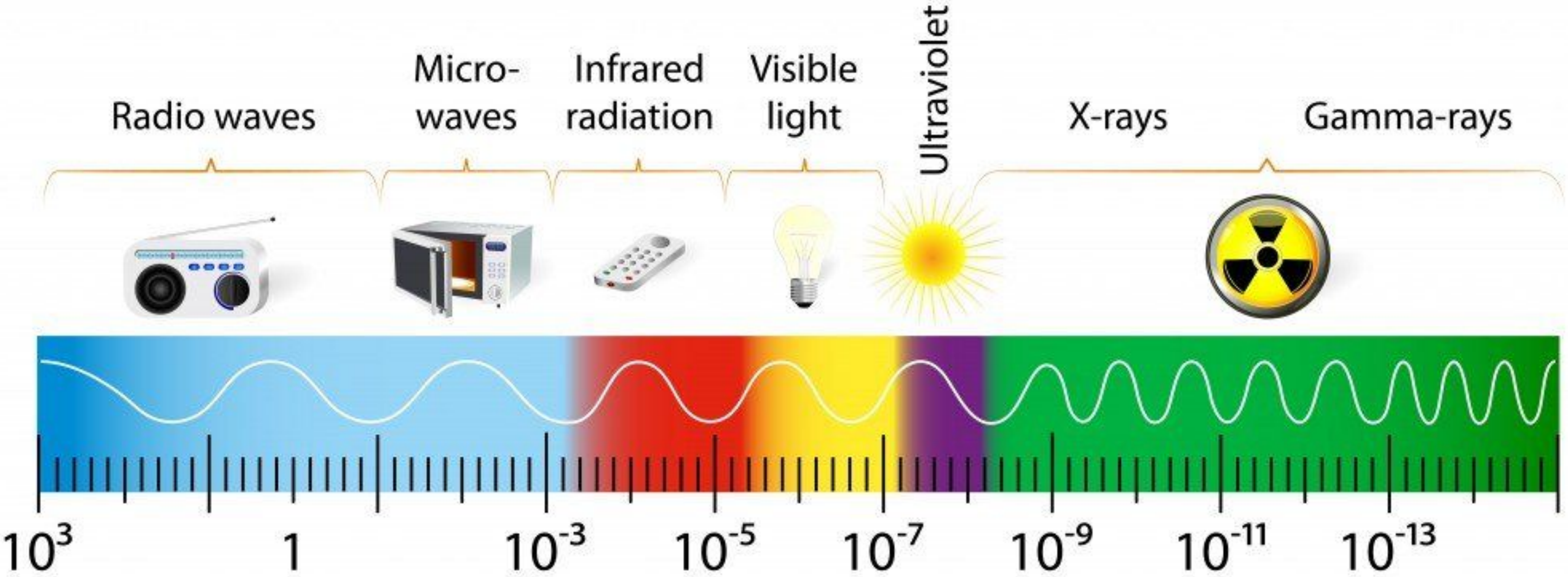
BROADCOM®

Goals

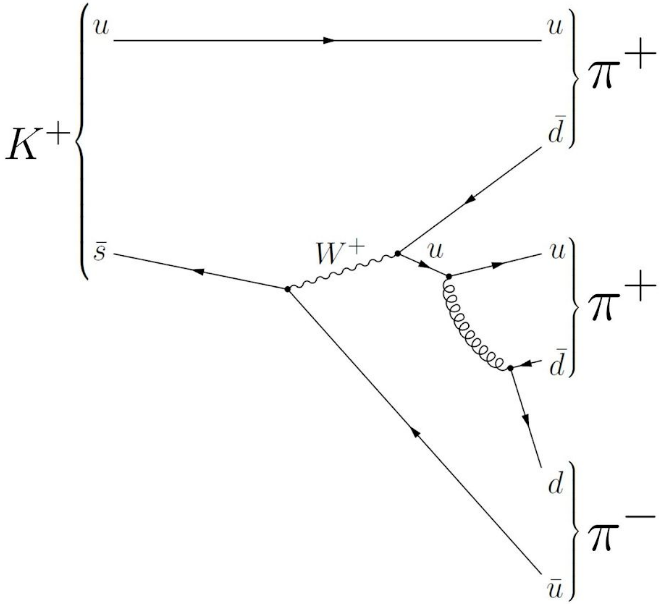
- Apply generalized topology diagrams towards standardizing WiFi test rig designs
- Better emulate over the air (OTA) scenarios when using “conducted equipment”
- Frequency band is 2.4-6 GHz
- Use “low cost” conducted equipment sold by outside vendors
- Automate with python

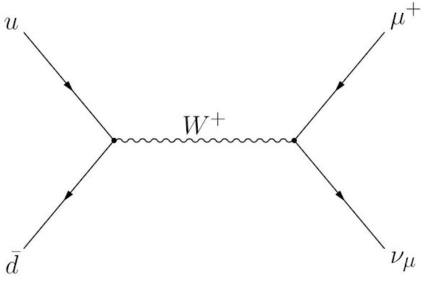
Theory

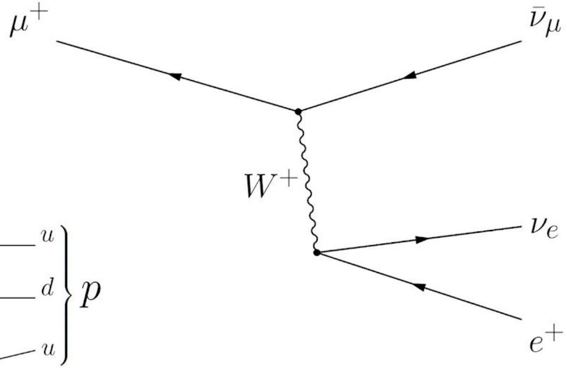
THE ELECTROMAGNETIC SPECTRUM

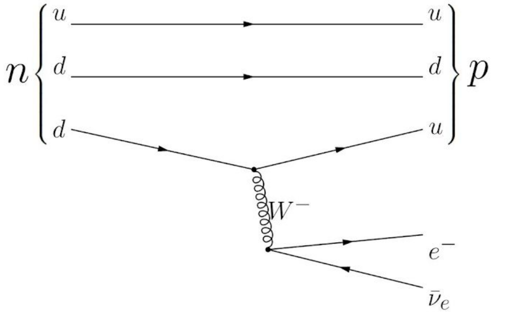


An introduction to:





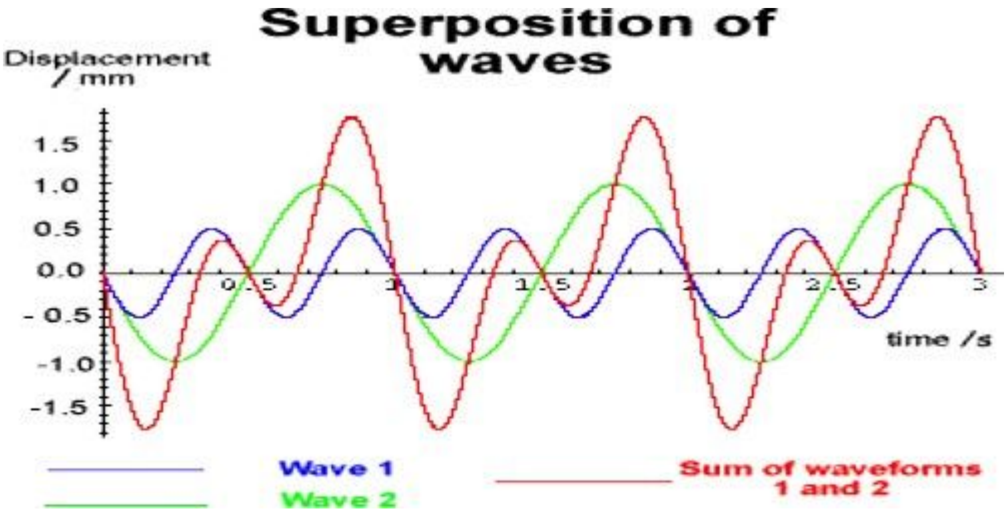




Feynman Diagrams

astronomy2GCSE

Superposition



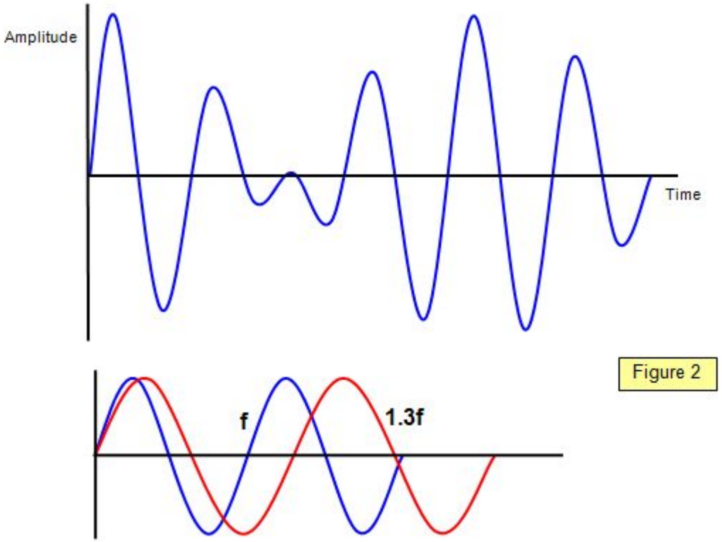
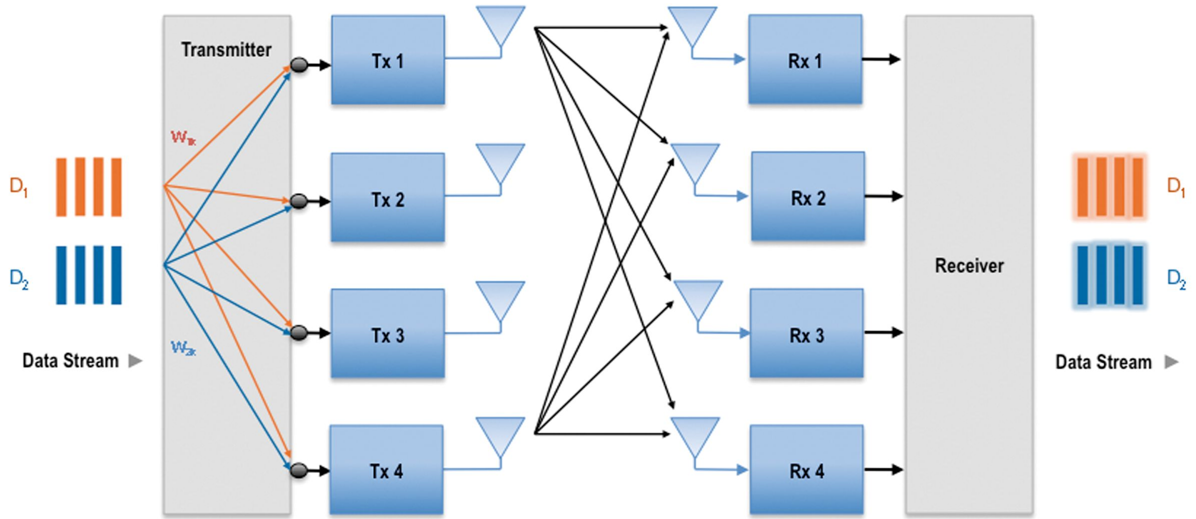


Figure 2

Mathematical Models

A Basic MiMo Model

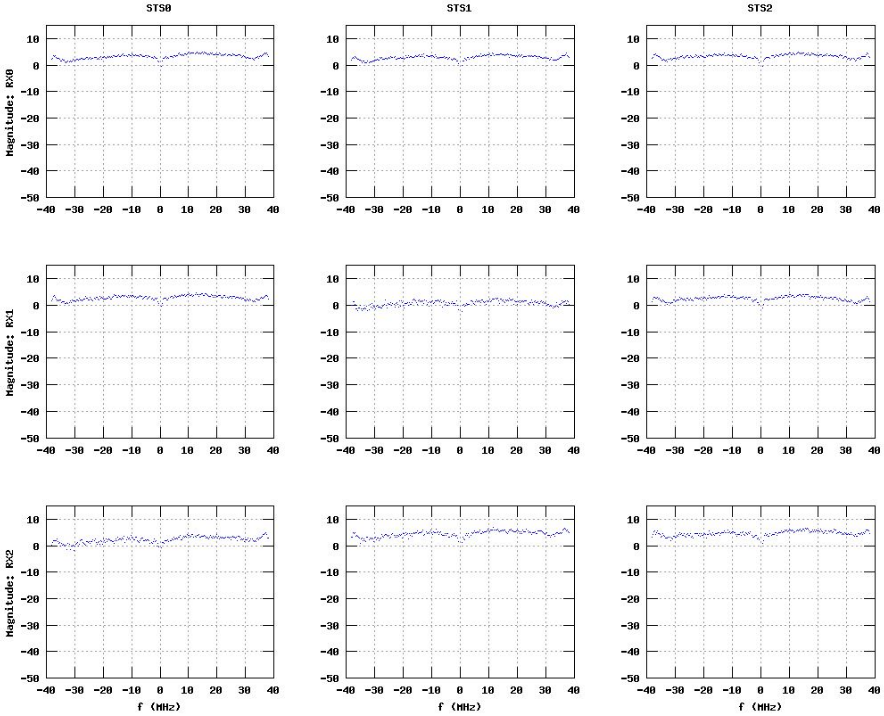


Transfer Matrix

$$\begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} = \begin{bmatrix} D1 & 0 & 0 & 0 \\ 0 & D2 & 0 & 0 \\ 0 & 0 & D3 & 0 \\ 0 & 0 & 0 & D4 \end{bmatrix} \times \begin{bmatrix} h11 & h12 & h13 & h14 \\ h21 & h22 & h23 & h24 \\ h31 & h32 & h33 & h34 \\ h41 & h34 & h43 & h44 \end{bmatrix} \times \begin{bmatrix} D1' & 0 & 0 & 0 \\ 0 & D2' & 0 & 0 \\ 0 & 0 & D3' & 0 \\ 0 & 0 & & D4' \end{bmatrix} \times \begin{bmatrix} S1 \\ S2 \\ S3 \\ S4 \end{bmatrix}$$

- S-Matrix is the transmitted signal (or energy) per antenna
- R-Matrix is the received signal
- D-Matrices are the cables and attenuators between the antenna and the H-Matrix (one for source and one for sink)
- H-Matrix is the MiMO “channel” (signal mixing, e.g butler matrix, splitter/combiner, power divider, MuMimo midbox)

Phychanest



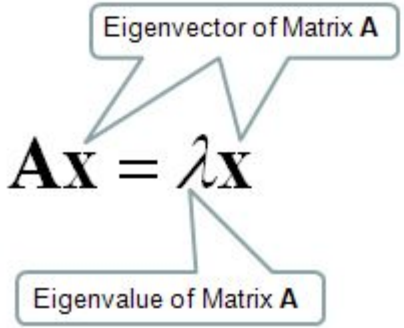
STS0
STS1
STS2
Magnitude: RX0
Magnitude: RX1
Magnitude: RX2
f (MHz)

Hermitian

Diagonalizing Hermitian Matrices

- **The eigenvalues of a Hermitian Matrix are always real**
- **For a Hermitian matrix the eigenvectors corresponding to two different eigenvalues are orthogonal**
- **A matrix has real eigenvalues and can be diagonolized by a unitary similarity transformation if and only if it is Hermitian**
- **A matrix has real eigenvalues and can be diagonolized by an orthogonal similarity transformation if and only if it is symmetric**

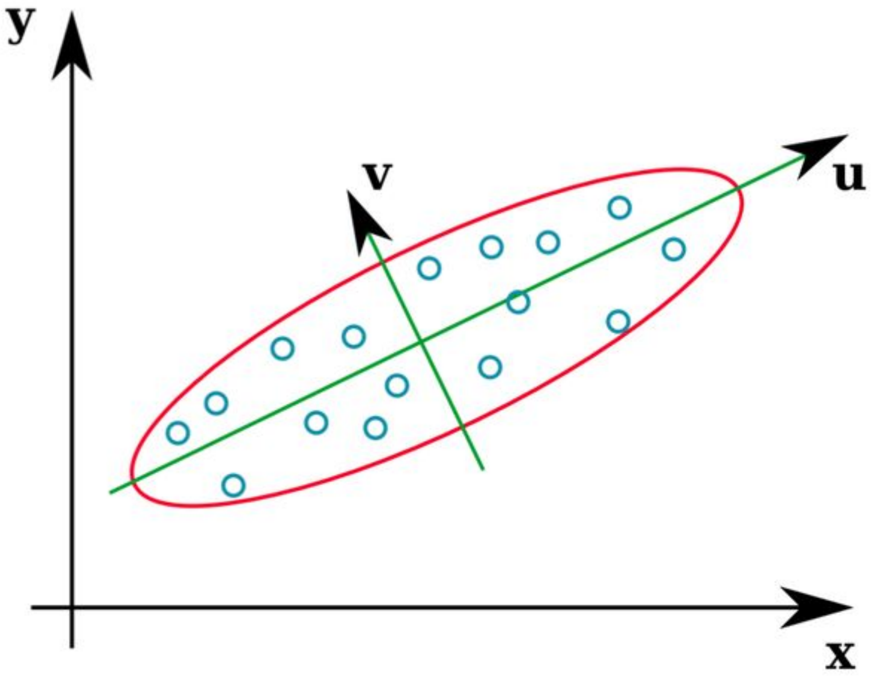
"Eigens"

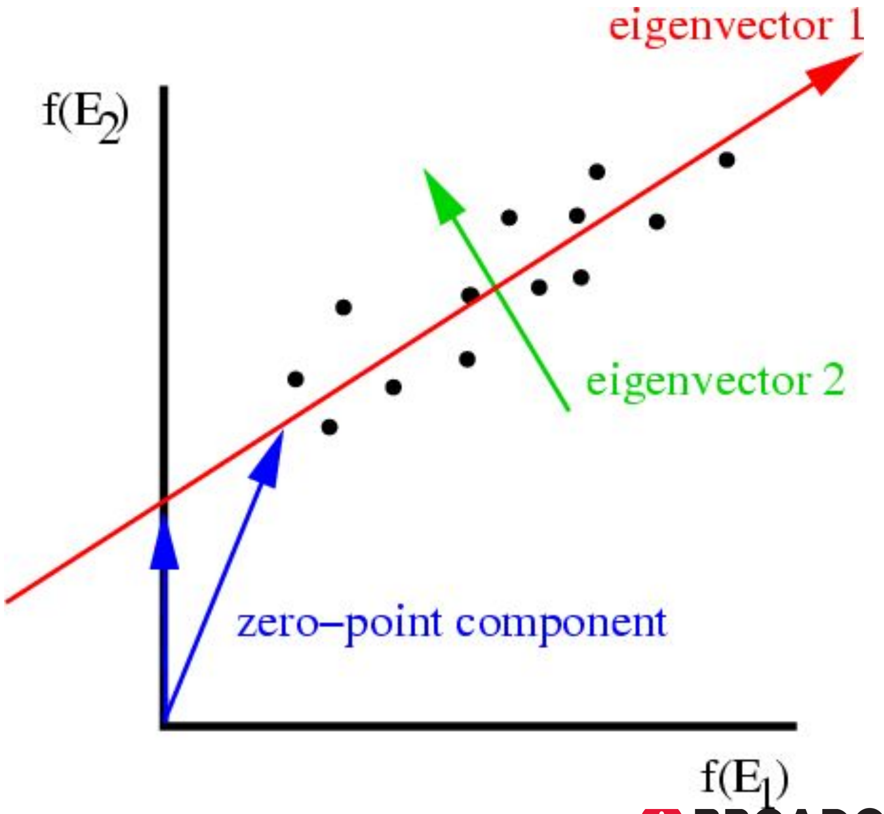


Eigen 'in Germany' means 'Characteristics' in English. So you may guess, 'Eigen vector' would be a special vector that represents a specific characteristics of a Matrix (a Square Matrix) and 'Eigen value' would be a special value that represents a specific characteristics of a Matrix (a Square Matrix)

If you think of a Matrix as a geometric transformer, the Matrix usually perform two types of transformational action. One is 'scaling(extend/shrink)' and the other one is 'rotation'.

Eigenvalues, Eigenvectors





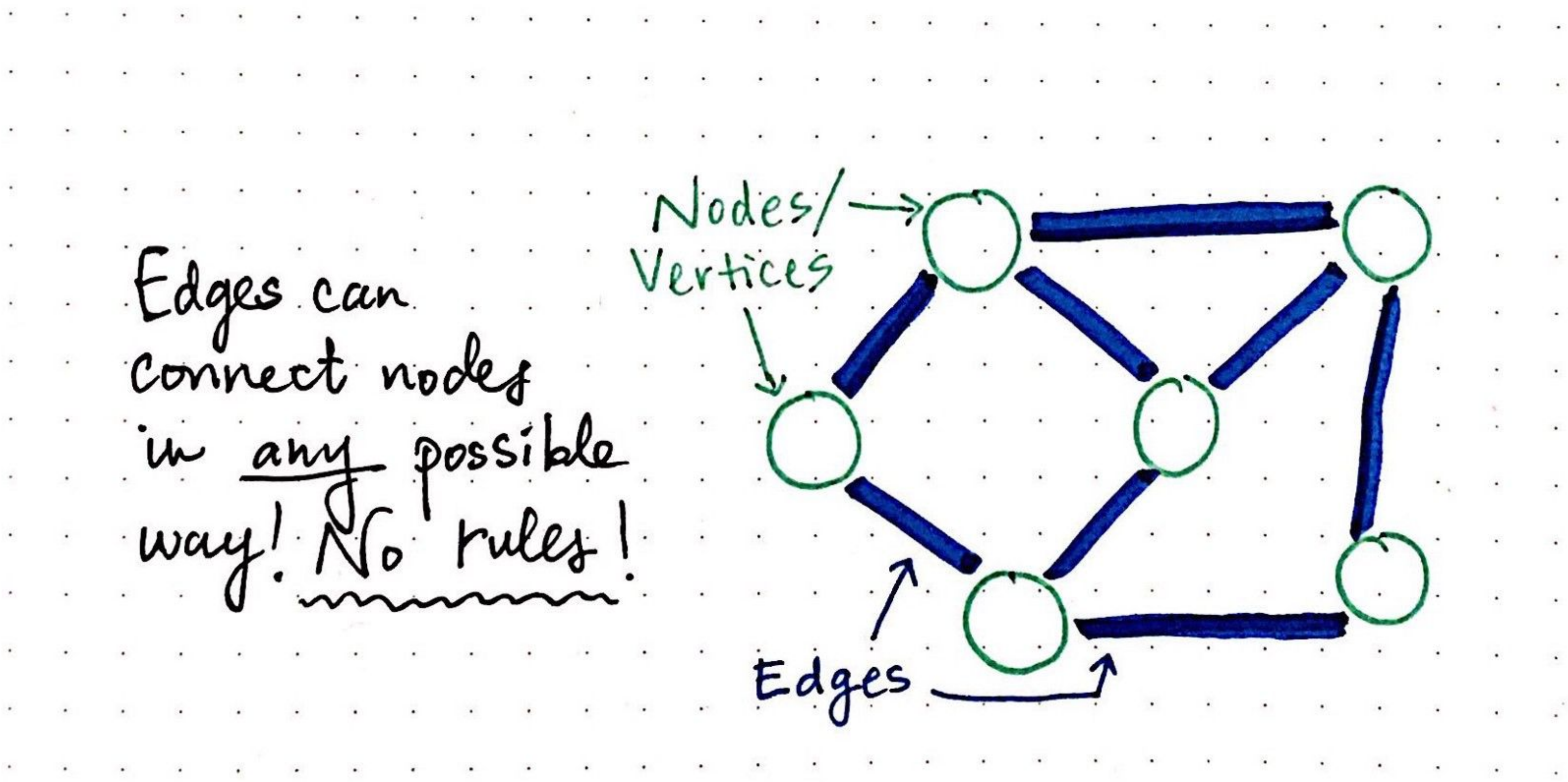
Graph Theory

In mathematics, graphs are a way to formally represent a network, which is basically just a collection of objects that are all interconnected.

Applied to MiMO

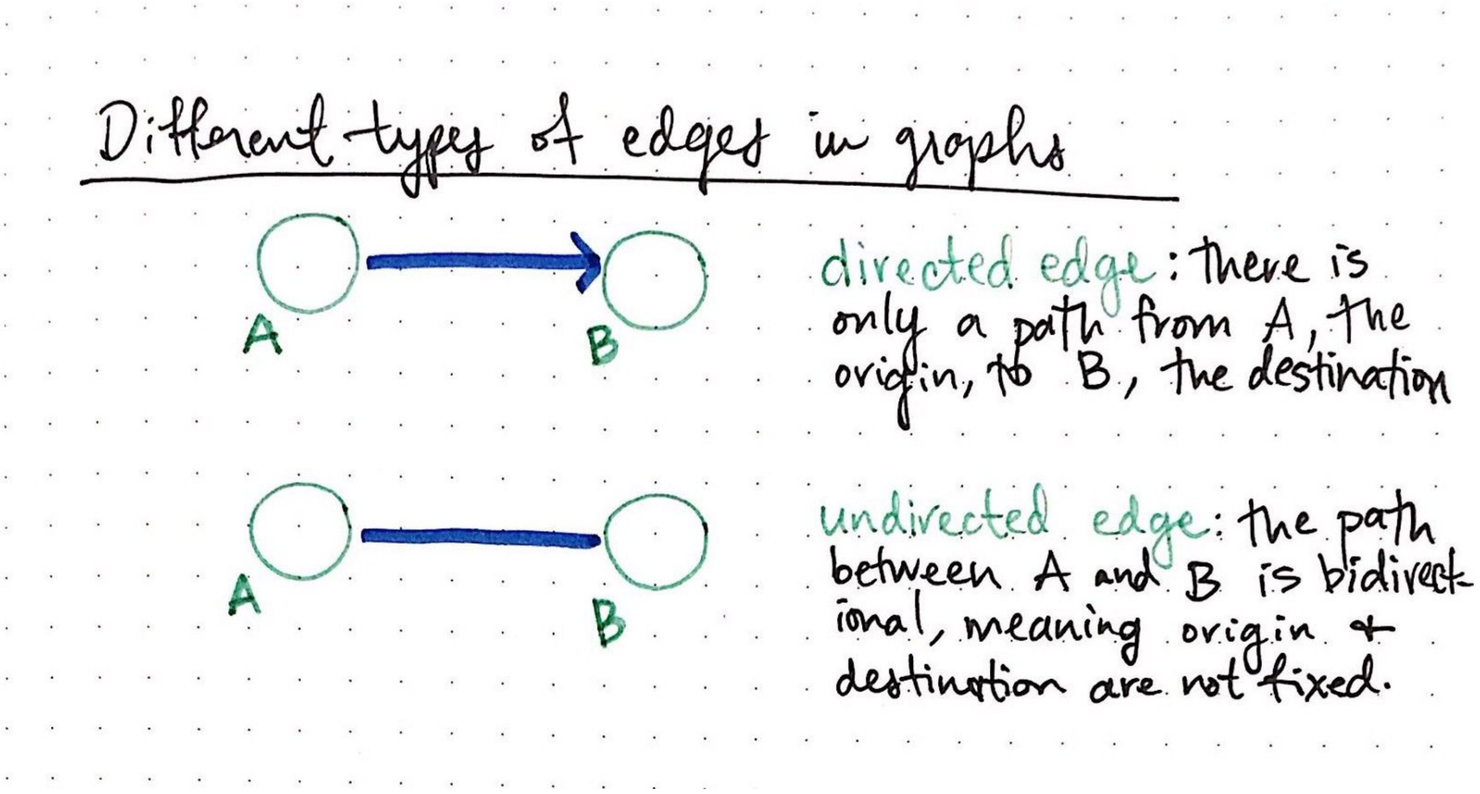
- The eigenvalues are the ones that characterize the MIMO channel capacity, while the eigenvector does not affect it (for a single-user point-point MIMO channel).
- The capacity-achieving transmission strategy is to send one data signal in each of the eigendirections. Hence, the eigenvalues tell us how strong these parallel channels are.

Gentle Introduction

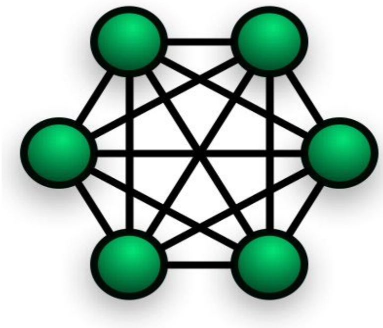


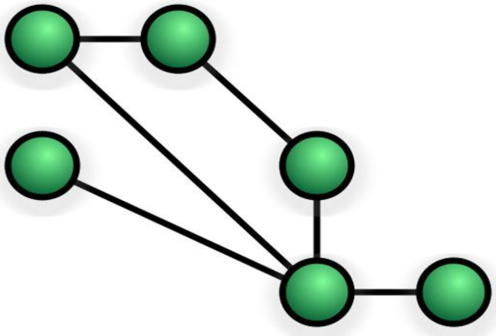
BROADCOM

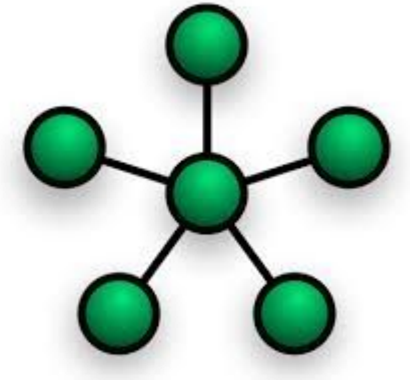
Edge types

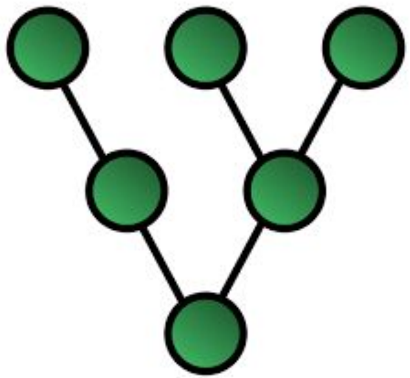

Different types of edges in graphs
A
B
directed edge: there is only a path from A, the origin, to B, the destination
A
B
undirected edge: the path between A and B is bidirectional, meaning origin + destination are not fixed.

Graph examples

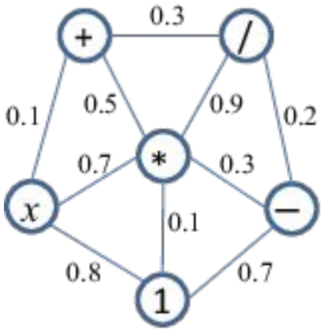




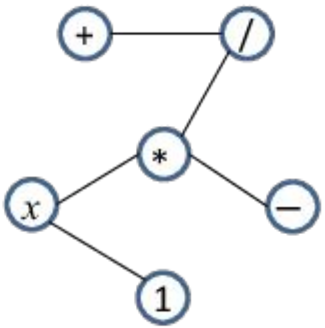




Graph->Tree->Distance Matrix



Weighted graph

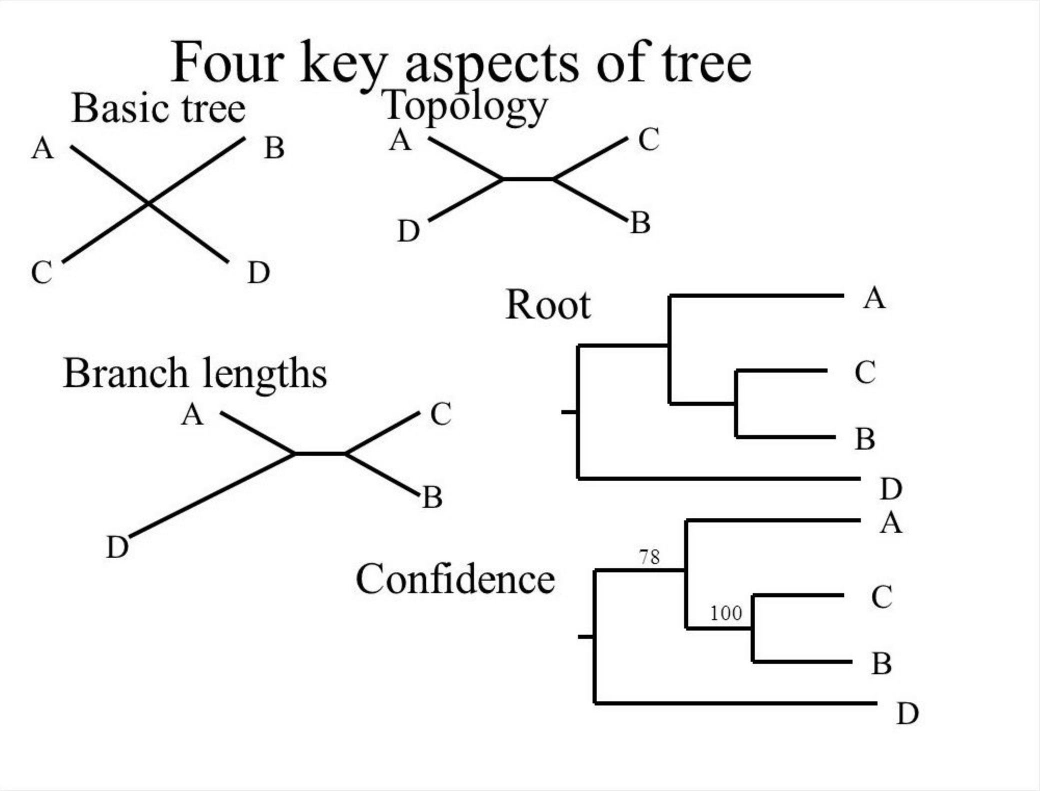


Minimum spanning tree

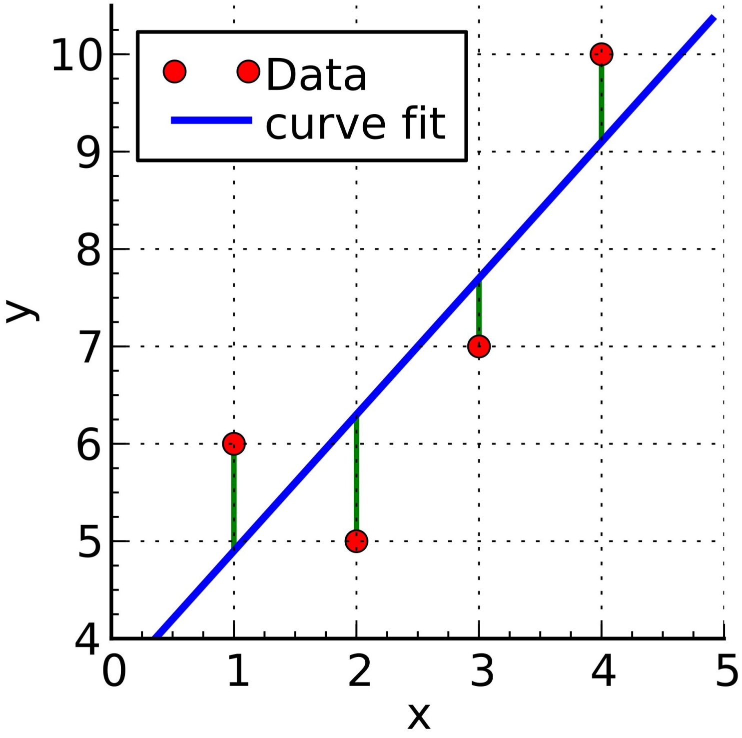
	+	-	*	/	x	1
+		3	2	1	3	4
-	3		1	2	2	3
*	2	1		1	1	2
/	1	2	1		2	3
x	3	2	1	2		1
1	4	3	2	3	1	

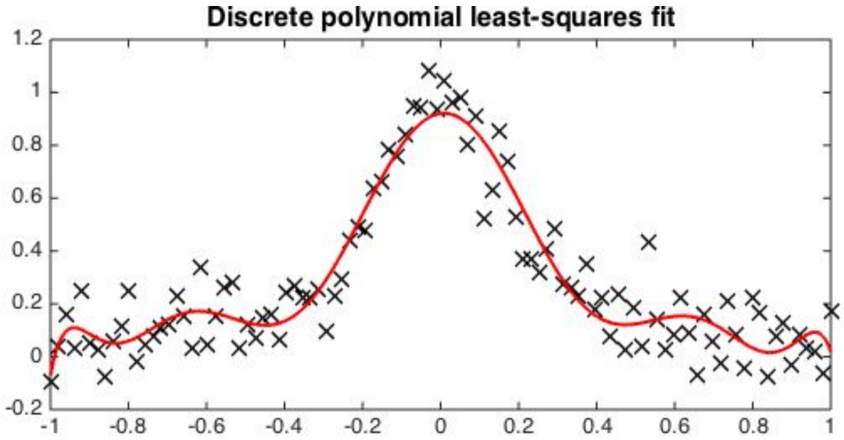
Distance matrix

Tree examples



Least Squares Fitting





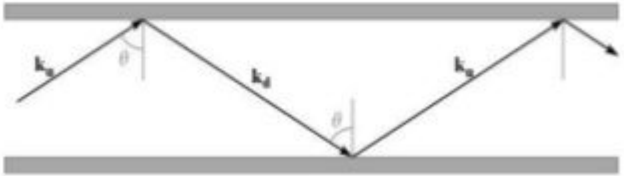
Building from available parts

Wave guides

Waveguide

- A waveguide is a structure that guides waves, such as electromagnetic waves or sound waves They enable a signal to propagate with minimal loss of energy by restricting expansion to one dimension or two
- Zigzag reflection, waveguide mode, cutoff frequency

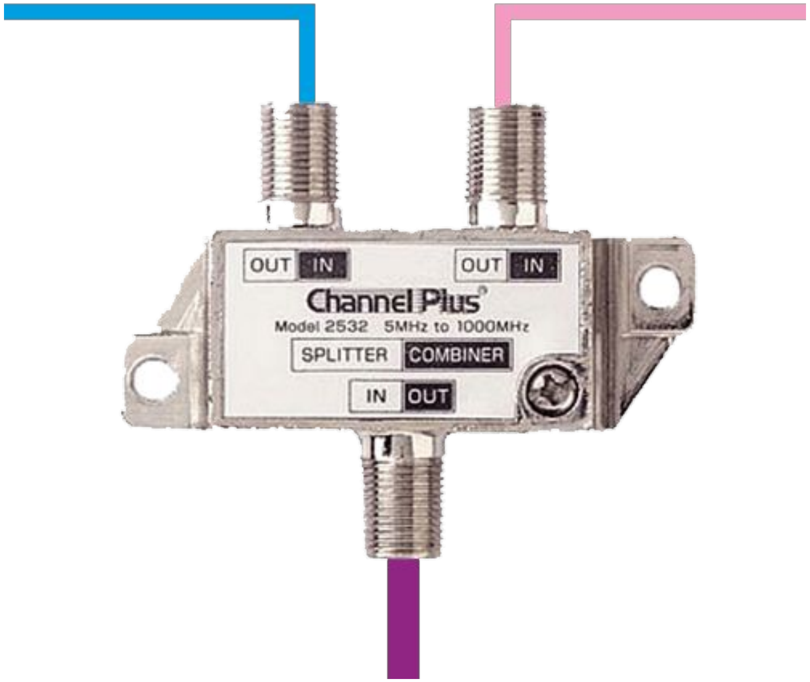
$$|\mathbf{k}_u| = |\mathbf{k}_d| = k = \omega\sqrt{\mu\varepsilon}$$



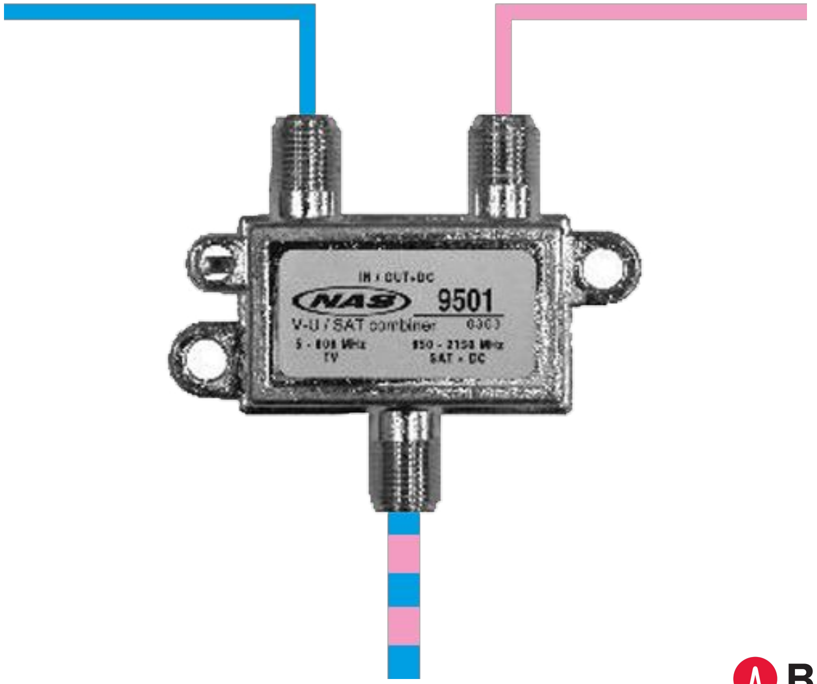
(b)

Fixed parts: e.g. Splitter/Combiner, Diplexers

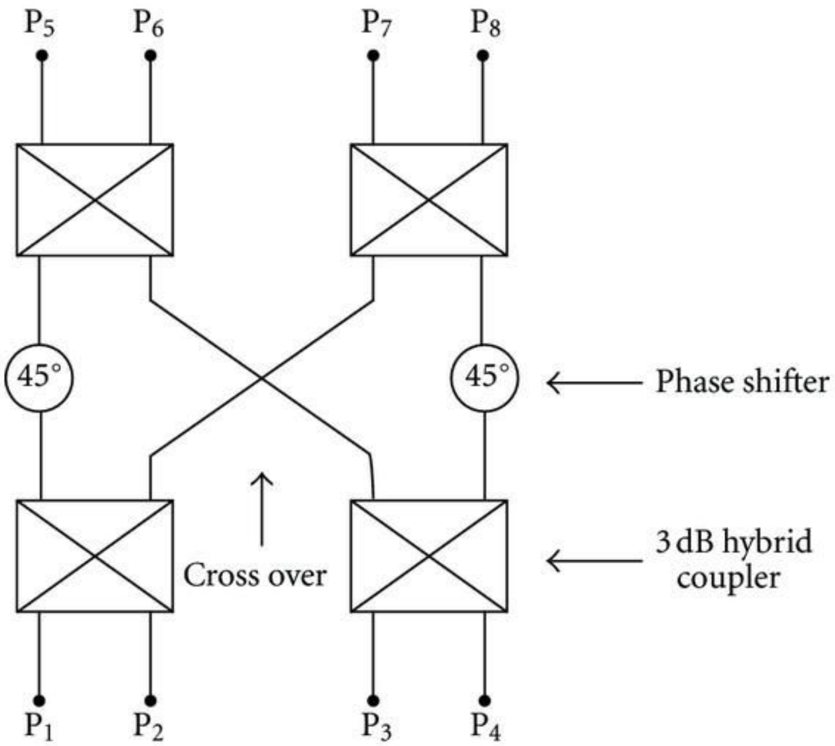
COMBINER



DIPLEXER



Butler Matrix

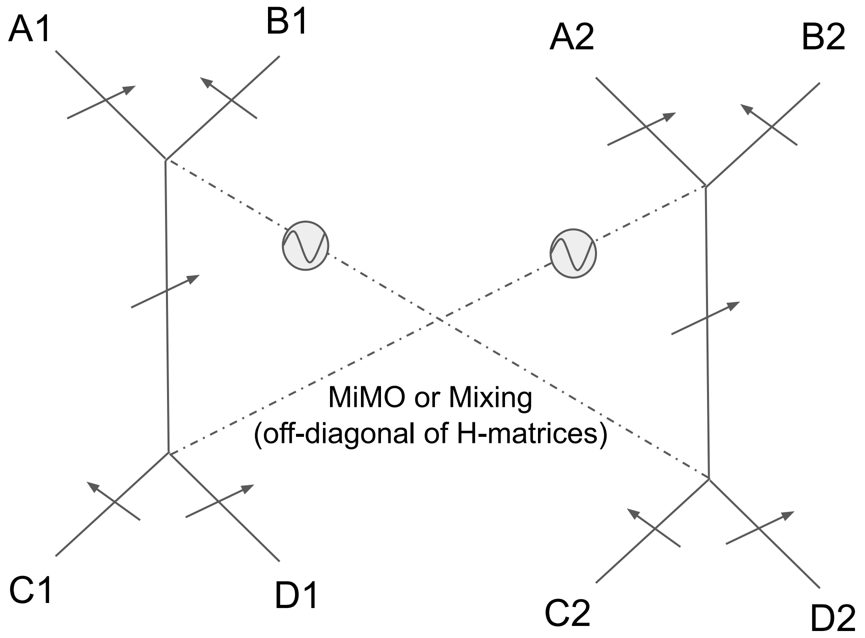


BROADCOM

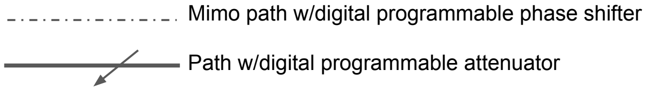
Programmable parts: attenuators and phase shifters

Generalized WiFi Graphs

Four Node, 5-Branch Tree with 2x2 MIMO

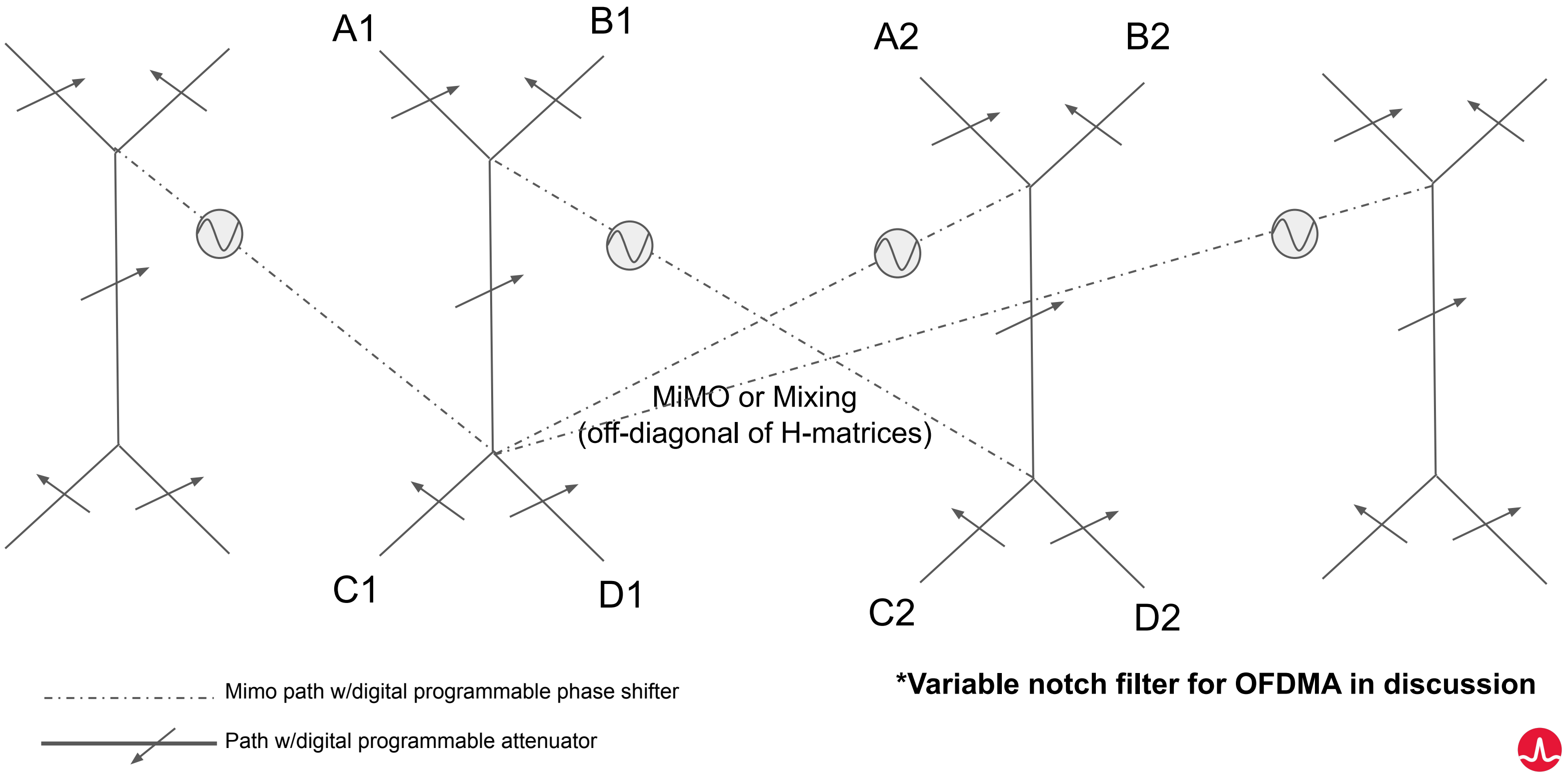


***Variable notch filter for OFDMA in discussion**



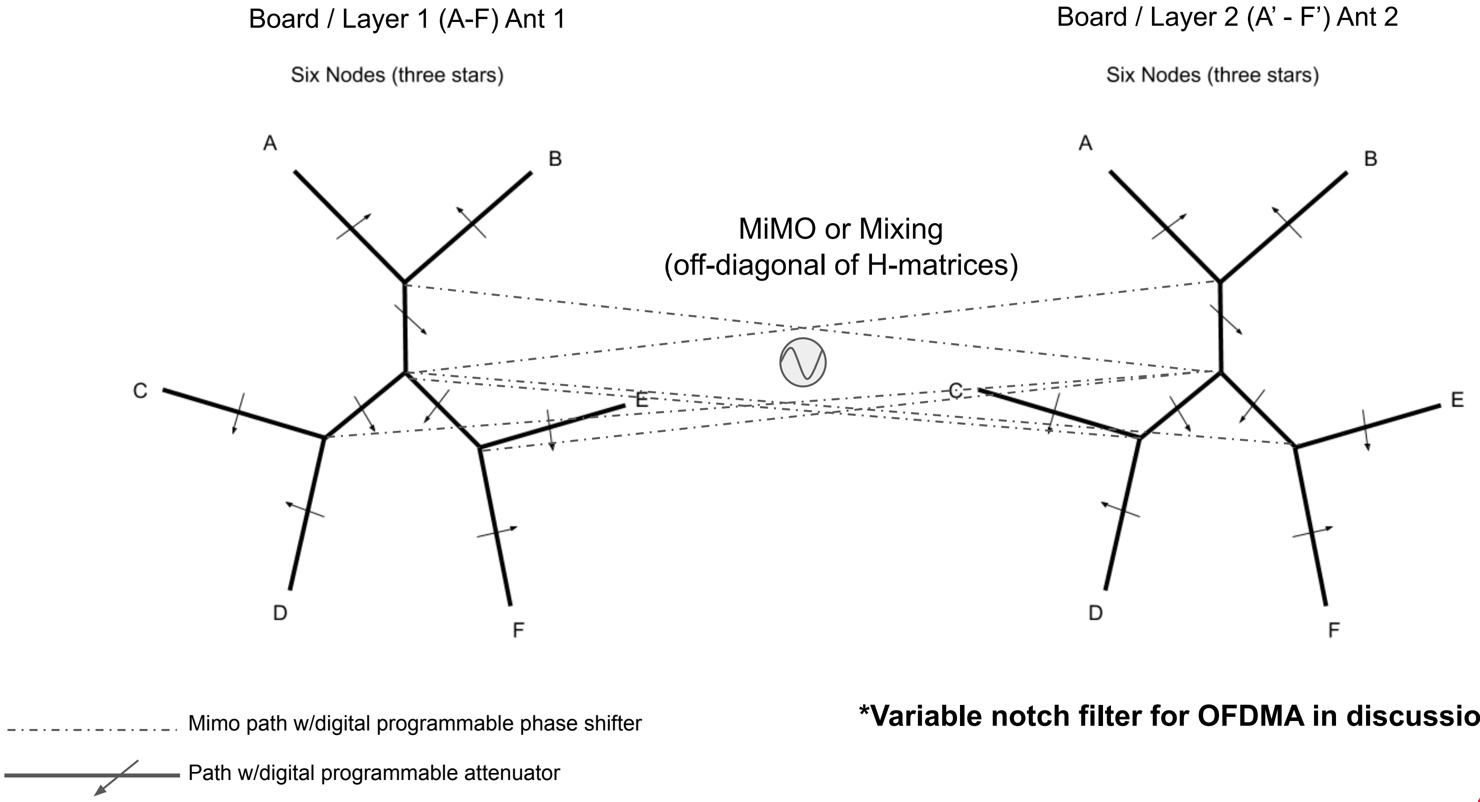
BROADCOM

Four Node, 5-Branch Tree with 4x4 MIMO



***Variable notch filter for OFDMA in discussion**

6 Node, 3 BSS topology (3 BSS, 6 nodes, 2x2 MIMO)



***Variable notch filter for OFDMA in discussion**

Python Code

Minimizing Total Error (Partial diff eq)

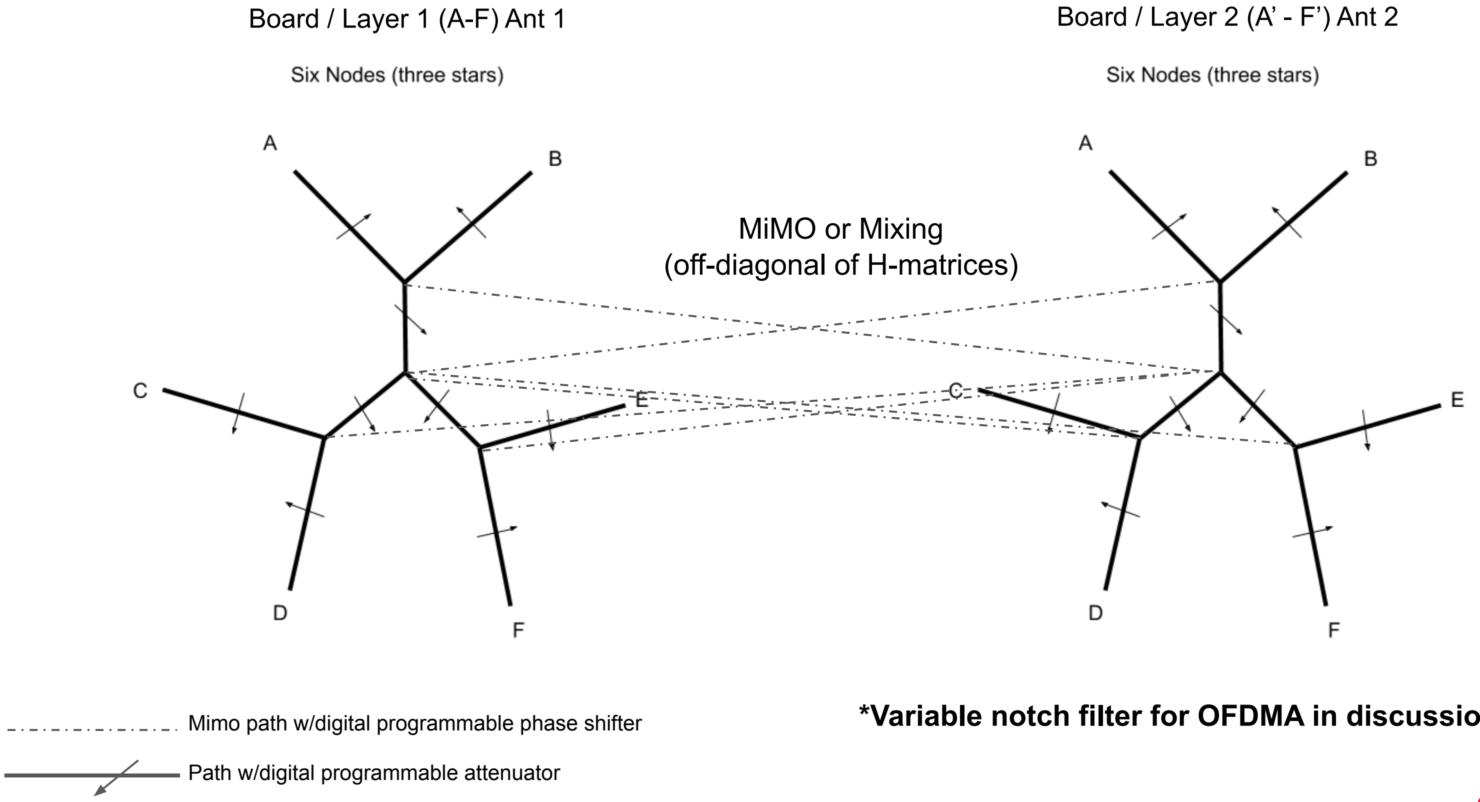
```
#
#  S1                    S3
#    \                  /
#   a \      b         /c
#      \______________/
#      /              \
#   d /                \ e
#    /                  \
#  S2                    S4
#
# D12 = a + d
# D13 = a + b + c
# D14 = a + b + e
# D23 = d + b + c
# D24 = d + b + e
# D34 = c + e
#
# Q = (D12 - a - d)**2 + (D13 - a - b - c)**2 + (D14 - a - b - e)**2 - (D23 - d - b - c)**2 + (D24 - d - b - e)**2 + (D34 - c - e)**2
#
# Substitute the desired RSSI (or distance) into D(i,j) then
# Find the minimum for Q by taking the partial derivatives respect to a,b,c,d and e
# and then solve the five simultaneous equations
```

Python Code

Q = (D12 - a - d)2 + (D13 - a - b - c)**2 + (D14 - a - b - e)**2 - (D23 - d - b - c)**2 + (D24 - d - b - e)**2 + (D34 - c - e)**2**

```
@property
def attnsettings(self):
    # Create the expression for the tree, i.e. the Summation of the squared errors
    q = ((Float(self.distances[0]) - self.a - self.d) ** 2) + ((Float(self.distances[1]) - self.a - self.b - self.c) **
2) + ((Float(self.distances[2]) - self.a - self.b - self.e) ** 2) + ((Float(self.distances[3]) - self.d - self.b - self.c) **
2) + ((Float(self.distances[4]) - self.d - self.b - self.e) ** 2) + ((Float(self.distances[5]) - self.c -  self.e ) ** 2)
    # perform the five partial differentiations and solve the five equations to minimize Q
    solution = linsolve([diff(q, self.a), diff(q,self.b), diff(q,self.c), diff(q,self.d), diff(q,self.e)],
self.a,self.b,self.c,self.d,self.e)
    (a0, b0, c0, d0, e0) = next(iter(solution))
    error = q.subs([(self.a,a0),(self.b,b0),(self.c,c0),(self.d,d0),(self.e,e0)])
    # print("{}".format(solve([diff(q, self.a), diff(q,self.b), diff(q,self.c), diff(q,self.d), diff(q,self.e)])))
    print('Error:{:.2f}'.format(float(sqrt(error))))
    return solution
```

Common/Generalized WiFi topology (3 BSS, 6 nodes, 2x2 MIMO)



***Variable notch filter for OFDMA in discussion**

6 Nodes, (11 degrees vs 22)

```
#              S3      S4
#               \     /
#              f \   / g
#                 \ /
#  S1              *
#     \           /
#    d \         / b
#       \   a   /
#        *-----*
#       /       \
#      /         \ c
#   e /           \
#    /             *
#  S2             / \
#              h /   \ i
#               /     \
#              S5     S6
#
#  0) D12 = d + e
#  1) D13 = d + a + b + f
#  2) D14 = d + a + b + g
#  3) D15 = d + a + c + h
#  4) D16 = d + a + c + i
#  5) D23 = e + a + b + f
#  6) D24 = e + a + b + g
#  7) D25 = e + a + c + h
#  8) D26 = e + a + c + i
#  9) D34 = f + g
# 10) D35 = f + b + c + h
# 11) D36 = f + b + c + i
# 12) D45 = g + b + c + h
# 13) D46 = g + b + c + i
# 14) D56 = h + i
```

Python Code

Q = (D12 - d - e)2 + (D13 - d - a - c)**2 + (D14 - a - b - e)**2 - (D23 - d - b - c)**2 + (D24 - d - b - e)**2 + (D34 - c - e)**2**

```
@property
def attnsettings(self):
    # Create the expression for the tree, i.e. the Summation of the squared errors
    q = ((Float(self.distances[0]) - self.d - self.e) ** 2) + ((Float(self.distances[1]) - self.d - self.a - self.b - self.f) ** 2) + ((Float(self.distances[2]) - self.d - self.a - self.b - self.g) ** 2) +
((Float(self.distances[3]) - self.d - self.a - self.c - self.h) ** 2) + ((Float(self.distances[4]) - self.d - self.a - self.c - self.i) ** 2) + ((Float(self.distances[5]) - self.e - self.a - self.b - self.f) ** 2) +
((Float(self.distances[6]) - self.e - self.a - self.b - self.g) ** 2) + ((Float(self.distances[7]) - self.e - self.a - self.c - self.h) ** 2) + ((Float(self.distances[8]) - self.e - self.a - self.c - self.i) ** 2) +
((Float(self.distances[9]) - self.f - self.g) ** 2) + ((Float(self.distances[10]) - self.f - self.b - self.c - self.h) ** 2) + ((Float(self.distances[11]) - self.f - self.b - self.c - self.i) ** 2) +
((Float(self.distances[12]) - self.g - self.b - self.c - self.h) ** 2) + ((Float(self.distances[13]) - self.g - self.b - self.c - self.i) ** 2) + ((Float(self.distances[14]) - self.h - self.i) ** 2)
    # perform the nine partial differentiations and solve the nine equations to minimize Q
    solution = linsolve([diff(q, self.a), diff(q,self.b), diff(q,self.c), diff(q,self.d), diff(q,self.e), diff(q,self.f), diff(q,self.g), diff(q,self.h), diff(q,self.i)], self.a, self.b, self.c, self.d, self.e, self.f, self.g,
self.h, self.i)
    (a0, b0, c0, d0, e0, f0, g0, h0, i0) = next(iter(solution))
    error = q.subs([(self.a,a0),(self.b,b0),(self.c,c0),(self.d,d0),(self.e,e0), (self.f,f0),(self.g,g0),(self.h,h0), (self.i,i0)])
    print('Error:{:0.2f}'.format(float(sqrt(error))))
    return solution
```

Take aways

- Seminal MIMO paper
- David Reed's myth of interference
- Least Squares Five Branch Tree link