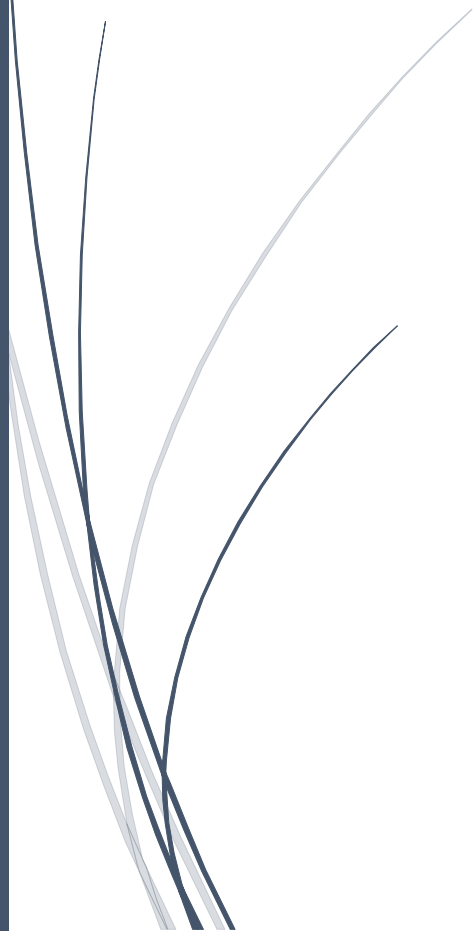
Corso:
FONDAMENTI DI ANTENNE

Corso di Laurea Triennale in Ingegneria Elettronica e delle Telecomunicazioni

ARRAY DI ANTENNE E PHASED ARRAY ANTENNAS

Simulazione in Matlab di un'array collineare di antenne, e relativo studio del loro impiego come antenne a puntamento elettronico (electronically scanned arrays)



Docente:
PROF. ING. FRANCESCO PRUDENZANO

Studente:
FRANCESCO CATERINA

INTRODUZIONE E CENNI TEORICI GENERALI

Il seguente lavoro ha lo scopo di illustrare dal punto di vista didattico il funzionamento di un' array di antenne, cioè un sistema costituito da più antenne (in questo caso utilizzate in modalità trasmittente, ma gli stessi concetti qui illustrati valgono in maniera duale per array di antenne riceventi) le cui porte di ingresso sono collegate tra loro in una configurazione particolare, attraverso linee di trasmissione passive opportunamente progettate, in modo da avere particolari relazioni fra i fasori delle alimentazioni di ogni singola porta, formando alla fine comunque un'unica porta di ingresso che si riferisce al sistema nel suo complesso; qualora invece si volesse controllare istante per istante l'alimentazione di ogni singolo elemento in ampiezza o fase le varie porte dei singoli elementi vengono collegate tra loro mediante blocchi elettronici attivi, quali amplificatori o sfasatori.

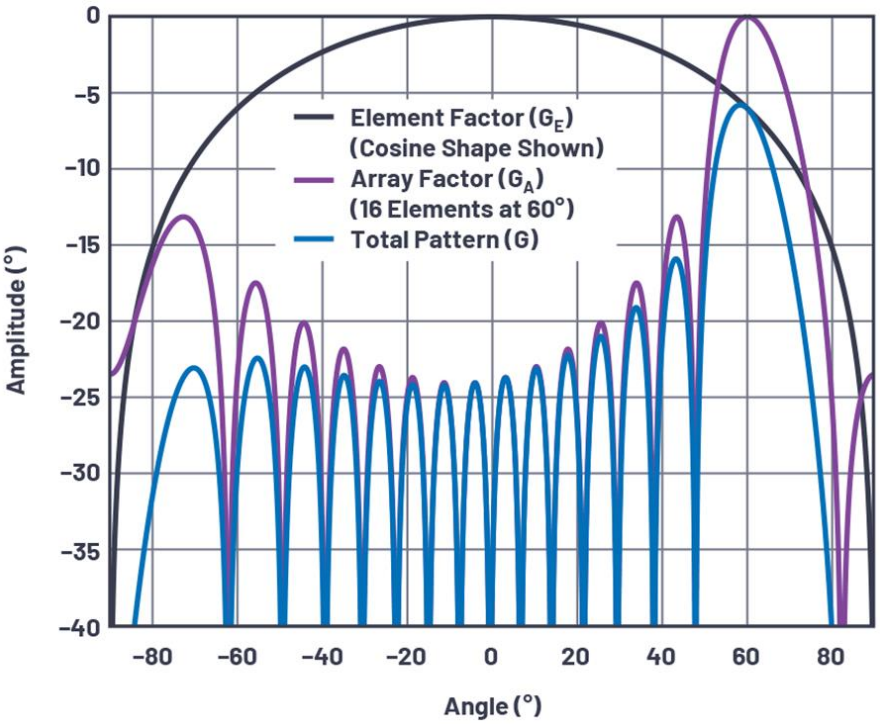
Lo scopo dei blocchi amplificatori e sfasatori posti su ciascuna porta è appunto quello di regolare l'ampiezza e la fase del segnale che arriva a ciascun elemento; come vedremo ciò ci dà ampio controllo sul pattern di direttività del sistema, in quanto regolando opportunamente le ampiezze e/o le fasi delle tensioni (o correnti) di alimentazione di ogni singolo elemento si vanno a far interferire i campi elettromagnetici prodotti da ciascun singolo elemento l'uno con l'altro, andando così a creare zone di spazio in cui avremo **interferenza costruttiva,** ed altre zone in cui avremo **interferenza distruttiva**, dando luogo così a picchi della funzione di direttività che non erano originariamente presenti nel pattern di radiazione del singolo elemento.

Il pattern di radiazione (funzione di direttività) dell'array di antenne nel suo insieme dunque è dovuto non soltanto al pattern di radiazione del singolo elemento, ma anche alla disposizione geometrica nella struttura dei singoli elementi, nonché delle relazioni fra le tensioni (o correnti) di alimentazione di ogni singolo elemento.
Tale fenomeno è modellato dal cosiddetto **fattore di gruppo**, che tiene appunto conto della geometria della struttura e delle alimentazioni sui singoli elementi; dal fattore di gruppo si ottiene la **funzione di direttività di gruppo**, che può essere immaginata come la funzione di direttività $f(\theta, \phi)$ di un sistema avente la stessa struttura e geometria di quello preso in considerazione, ma composto da antenne isotrope (ideali), che irradiano appunto con la stessa potenza in tutte le direzioni, alimentate con le stesse tensioni (o correnti) degli elementi della struttura originaria.
Ciò è in accordo con quanto detto: la funzione di direttività di gruppo $\mathbf{f_{gr}(\theta, \phi)}$ ed il fattore di gruppo modellano la geometria del sistema e le relazioni fra i segnali di alimentazione dei singoli elementi, non tengono conto del pattern di radiazione del singolo elemento.

Il pattern di radiazione della struttura nel suo insieme sarà dato dal prodotto della funzione di direttività del singolo elemento per la funzione di direttività di gruppo; ne risulta che quando la funzione di direttività del singolo elemento si mantiene più o meno costante nel range angolare in cui si hanno i lobi principali della funzione di direttività di gruppo, il pattern complessivo di radiazione del sistema sarà approssimativamente uguale a quello dato dalla sola funzione di direttività di gruppo.

Spesso anziché riferirsi al fattore di gruppo o alla direttività di gruppo, ci si riferisce al cosiddetto **fattore di schiera (Array Factor, AF)**, che altro non è che il fattore di gruppo dell'array normalizzato rispetto al modulo della corrente (che si presume uguale per tutti gli elementi).

Nella figura sottostante è riportato il pattern di radiazione di un' array di 16 elementi; è rappresentato il pattern di radiazione complessivo, quello del singolo elemento, e il fattore di schiera. Si noti come nella zona centrale, range azimutale = [-40°, +40°] dove il pattern di radiazione del singolo elemento è pressoché costante, la funzione di direttività complessiva segue quella di gruppo, mentre si scosta da essa al di fuori di questo range, in quanto la funzione di direttività del singolo elemento inizia a decadere, con un andamento di tipo cosinusoidale.



Element Factor vs Array Factor vs Total Pattern.
Source: https://www.analog.com/en/analog-dialogue/articles/phased-array-antenna-patterns-part1.html

L'elemento scelto nella simulazione da me ideata che sto per illustrare è il dipolo a mezz'onda (half-wave dipole). Com'è noto tale elemento ha una scarsa direttività, e nel piano azimutale la sua funzione di direttività è pressoché costante (tranne nella direzione in cui punta il dipolo stesso, dove si hanno dei minimi), dunque la funzione di direttività della struttura complessiva nel piano azimutale sarà quasi completamente determinata dall'array factor.

CLASSIFICAZIONE DELLE ARRAY DI ANTENNE IN BASE ALLA LORO GEOMETRIA

Le array di antenne, in base alla disposizione geometrica dei singoli elementi, si distinguono in:

- array collineari (**linear array**), in cui gli elementi sono tutti allineati lungo una specifica direzione, per esempio lungo uno degli assi coordinati, separati da una distanza comune d uguale fra tutti gli elementi (in tal caso si parla di array lineare uniforme) oppure distanziati opportunatamente con distanze variabili; una volta allineati tutti su un'unica direzione, essi possono anche esser ruotati di un certo angolo comune, a seconda dei fini progettuali del sistema e della direzione verso cui si vuole irradiare con massima potenza;
- array rettangolari (**rectangular array**), in cui i singoli elementi sono disposti in modo da formare una "griglia" discreta rettangolare o quadrata; è possibile così riferirsi al singolo elemento mediante una notazione matriciale del tipo (i, j); la spaziatura fra i singoli elementi ancora può essere costante o variabile da elemento ad elemento;
- array circolari: gli elementi sono disposti lungo una circonferenza di un certo raggio, ancora una volta la distanza può esser costante o variabile da elemento a elemento.

Queste sono quelle principalmente utilizzate, ma a seconda delle esigenze si possono configurare array aventi una configurazione geometrica qualsiasi.

Il lavoro qui presentato per semplicità si concentra sulle array collineari uniformi, ma sarebbe possibile fare simili considerazioni sulle altre geometrie.

CENNI TEORICI SU ARRAY DI ANTENNE COLLINEARI: ANTENNE BROADSIDE, ENDFIRE E PHASED ARRAY

In figura è rappresentata una linear array costituita da dipoli a mezz'onda, orientati tutti verso la direzione z, separati da una distanza d ed allineati nella direzione comune x; tale immagine è presa direttamente dalla simulazione Matlab che ho realizzato.

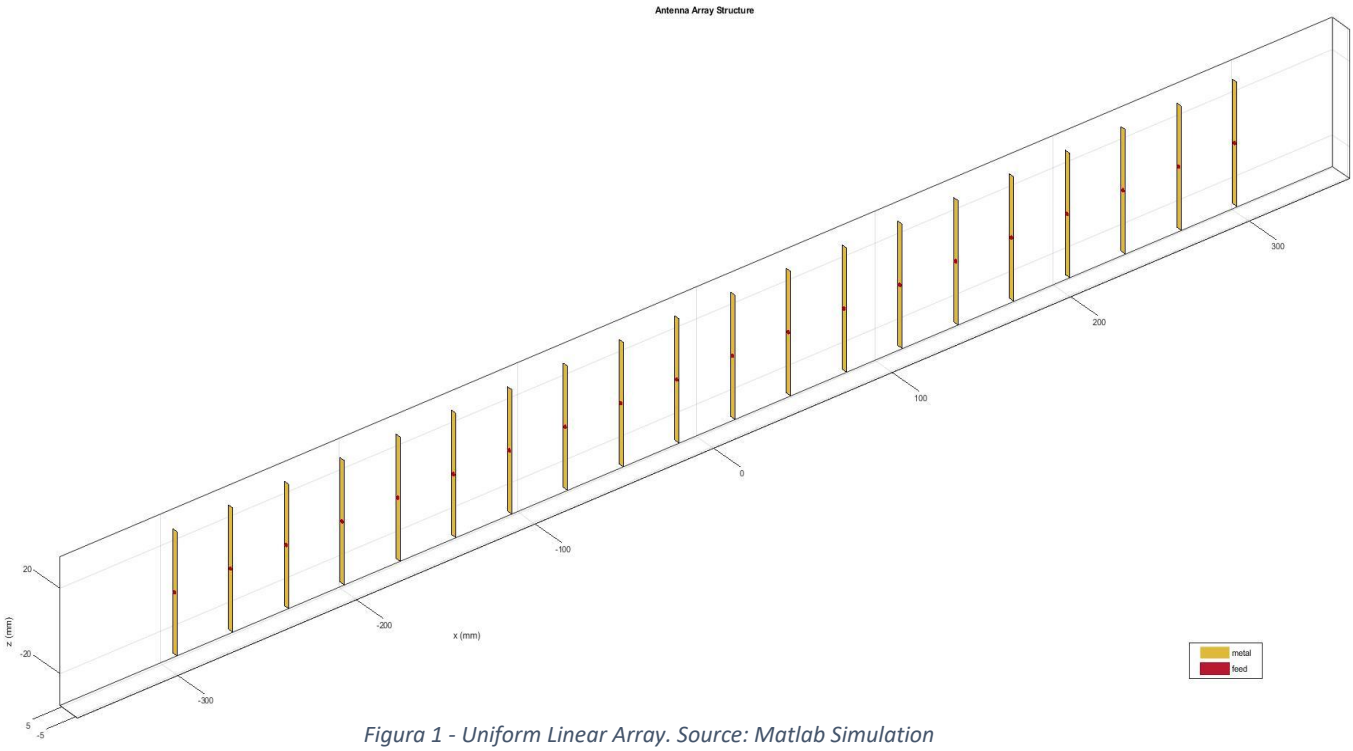


Figura 1 - Uniform Linear Array. Source: Matlab Simulation

Come ben noto dalla teoria, il fattore di gruppo dell'array si ottiene mediante sovrapposizione degli effetti, considerando ogni singolo elemento come una sorgente a sé stante e tenendo conto degli sfasamenti spaziali introdotti dal fatto che i singoli elementi si trovano in punti differenti dello spazio:

$$F(\theta,\varphi) = \sum_{i=1}^{N} I_i \, e^{j\beta r_{0i}\cos\delta} = I_0 \frac{e^{j\frac{N}{2}(\beta d\cos\delta + \alpha_0)}}{e^{\frac{j}{2}(\beta d\cos\delta + \alpha_0)}} \frac{sen\left[\frac{N}{2}(\beta d\cos\delta + \alpha_0)\right]}{sen\left[\frac{1}{2}(\beta d\cos\delta + \alpha_0)\right]}$$

con d distanza fra gli elementi, N numero di elementi, I_0 modulo della corrente che scorre negli elementi (che si suppone a fase 0 nel primo elemento) ed α_0 sfasamento fra l'alimentazione di un elemento ed il successivo.

Tale analisi tuttavia trascura gli effetti di mutua induzione che si presentano fra un elemento e l'altro, e dunque è valida solo quando la distanza fra un elemento e l'altro non è troppo piccola rispetto alla lunghezza d'onda; in caso contrario bisogna tener conto di tali effetti. Nella simulazione da me effettuata sono stati trascurati, in quanto si è posta una distanza tra gli elementi pari a d = λ/4, e sebbene a tale distanza inizino a comparire sicuramente effetti di mutua induzione essi sono trascurabili relativamente allo scopo didattico del progetto.

Le array collineari uniformi, in base alla relazione fra le correnti di alimentazione dei singoli elementi, producono un diverso pattern di radiazione; in base a ciò, esse si distinguono in tre macro-categorie:

- **Broadside** Linear Array Antennas: i lobi principali di radiazione si trovano lateralmente rispetto alla direzione di allineamento degli elementi; per esempio nella struttura rappresentata in figura 1, in cui i dipoli sono allineati lungo la direzione x, i lobi principali di radiazione qualora il sistema sia configurato come broadside si otterrebbero lungo l'asse y (un lobo nella direzione positiva delle y, l'altro nella direzione negativa). Dunque nel pattern di radiazione azimutale (piano xy) di tale array ritroveremo i lobi principali per $\varphi = +90^{\circ}, +270^{\circ}$.
  Per far funzionare un' array di antenne come broadside, lo sfasamento α_0 fra le correnti di alimentazione dei singoli elementi dev'essere nullo, $\alpha_0 = 0$.
- **Endfire** Linear Array Antennas: in tal caso il lobo principale di massima radiazione (che questa volta è unico) si trova nella stessa direzione dell'allineamento (da qui il nome endfire). Nel caso della nostra simulazione, ciò vuol dire che il lobo principale è orientato nella direzione positiva delle x ($\varphi = 0$). E' anche possibile irradiare nella stessa direzione, ma nel verso opposto, cioè nel nostro caso verso la direzione negativa delle x; in tal caso l'antenna "spara" all'indietro, ma è comunque un caso particolare di antenna endfire.
  Le due modalità di funzionamento endfire si ottengono rispettivamente massimizzando il rapporto tra i due seni del tipo sen(Nx)/sen(x) annullandone l'argomento, ed imponendo $\delta = 0$ e $\delta = 180^{\circ}$; dunque rispettivamente quando $\alpha_0 = -\beta d$ (modalità endfire in avanti) e per $\alpha_0 = +\beta d$ (modalità endfire "all'indietro").
- **Phased Linear Array**: questo è il caso più importante, e di applicazione notevole.
  Infatti variando mediante N-1 sfasatori a sfasamento variabile il ritardo (o anticipo) di fase α_0 nel range $[-\beta d , +\beta d]$, si passa da un comportamento endfire "in avanti" ad uno broadside dell'array, sino ad arrivare ad un pattern di radiazione di tipo endfire "all'indietro"; in pratica si riesce a coprire qualsiasi direzione φ nel piano azimutale, permettendo così di trasmettere l'informazione nella direzione desiderata.

Le phased array antennas, sia lineari che di qualsiasi altra geometria, trovano per esempio largo impiego nelle loro varie declinazioni nell'ambito delle telecomunicazioni mobili (ma non solo), in quanto permettono di direzionare gran parte della potenza elettromagnetica verso la direzione dell'utente, rendendo possibile variare la direzione di trasmissione in qualsiasi momento, in base alle necessità della rete e degli utenti "agganciati" alla cella.

Evoluzione diretta della tecnologia delle phased array antennas sono le **smart antennas:** esse sono costituite da phased array antennas in cui l'alimentazione dei singoli elementi (o gruppi di elementi) è controllata da un elemento di controllo digitale (microprocessori, FPGA, etc); tale sistema è poi abbinato ad una **sensor array antenna,** che altro non è che una seconda phased array antenna configurata come ricevente; in base al segnale ricevuto dal **sensor array**, processato mediante tecniche di **Spatial Digital Signal Processing** (Spatial DSP), il sistema è in grado di stimare la direzione di arrivo del segnale ricevuto (tali algoritmi prendono il nome di DOA, **Direction Of Arrival**), ed in base a ciò il microprocessore/FPGA genera un vettore **(beamforming vector)** contenente le informazioni circa l'ampiezza e la fase con cui dover guidare ogni singolo elemento dell'array trasmittente per poter trasmettere nella stessa direzione da cui si è ricevuto il segnale. Le possibili configurazioni di sistemi del genere, il numero di antenne usate per la ricezione/trasmissione del segnale, i tipi di algoritmi digitali applicati, etc sono ampiamente variabili e di diverso tipo, a seconda della tecnologia di applicazione, ma ciò fa capire l'importanza delle phased array antennas nell'ambito delle telecomunicazioni moderne.

SIMULAZIONE DI UNA LINEAR PHASED ARRAY ANTENNA IN MATLAB

Nella simulazione Matlab di cui segue l'illustrazione si è configurata una semplice linear phased array antenna, formata da dipoli a mezz'onda (lunghezza del singolo dipolo pari a λ/2) spaziati uniformemente di una distanza d (posta pari a d = λ/4, ma è possibile cambiarne il valore in quanto tutte le caratteristiche del sistema sono state parametrizzate).
La creazione dei singoli elementi, dell'array, e tutte le simulazioni sono state effettuate in maniera programmatica via codice Matlab utilizzando le funzioni dell'**antenna toolbox**, libreria messa a disposizione da Matlab per la sintesi e la simulazione di antenne e sistemi di antenne.
Il codice Matlab scritto ed eseguito per ricavare i risultati qui di seguito è allegato a fine relazione; qui seguono in linea di massima i passaggi effettuati per creare la struttura nonché i risultati ottenuti durante la simulazione.

CREAZIONE DEL SINGOLO DIPOLO ELEMENTARE A MEZZ'ONDA; FINE-TUNING DEL DIPOLO ELEMENTARE PER RENDERLO RISONANTE ALLA FREQUENZA DI LAVORO, E SIMULAZIONE DEI PATTERN DI RADIAZIONE

Dapprima si è creato il singolo elemento di base per la realizzazione dell'array; la scelta per semplicità è ricaduta sul dipolo. Tale elemento è messo a disposizione da Matlab mediante la funzione "dipole", che permette di specificare parametri quali la lunghezza e la larghezza del dipolo.

Il dipolo realizzato mediante tale funzione in realtà è un dipolo costituito da un conduttore piatto, non cilindrico: in pratica è un elemento che andrebbe inserito su uno strato dielettrico, a formare una PCB.
Per semplicità si è lasciato tale dipolo "libero", senza definire uno strato dielettrico, cosa che non inficia sullo scopo didattico della simulazione. Si può immaginare appunto che il vuoto faccia da dielettrico in tale simulazione. Inoltre tale approccio è utilizzato anche nei tutorial e negli esempi disponibili nella documentazione di Matlab.

Il dipolo è stato creato con una lunghezza pari a lambda/2; la lunghezza d'onda è stata calcolata avendo inserito la frequenza di lavoro come parametro all'inizio dello script Matlab.

A tale lunghezza d'onda, tuttavia, il dipolo presenta una reattanza di ingresso non nulla: in pratica non è un dipolo risonante. Infatti questa è l'impedenza di ingresso del dipolo plottata in Matlab:

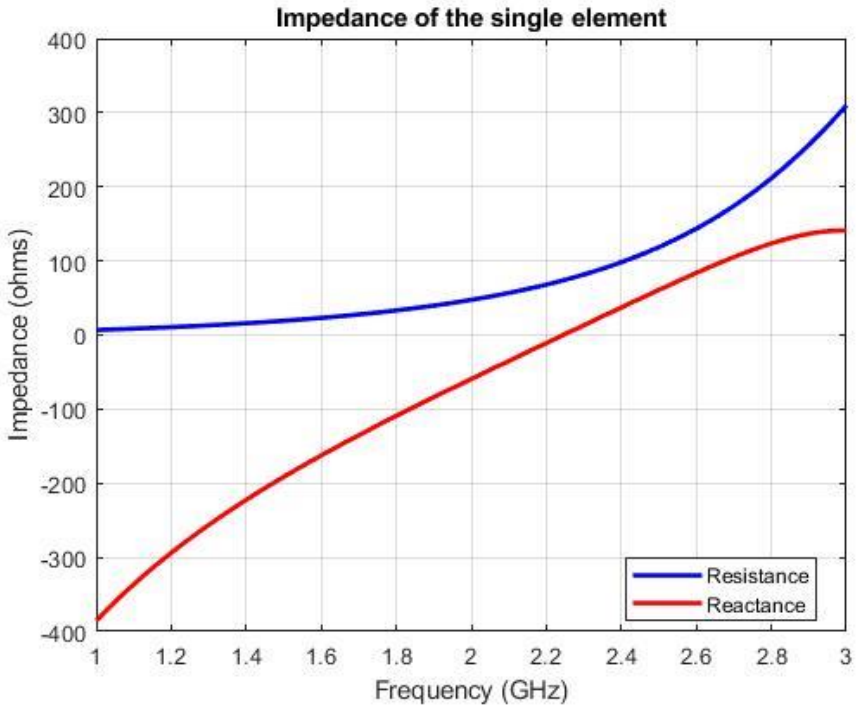


Figura 2 - Impedenza di ingresso del dipolo lambda/2

Come si può notare, alla frequenza di lavoro, posta pari a 2.4GHz, la reattanza non è nulla, nonostante la lunghezza del dipolo sia lambda/2: esse ha valore di qualche decina di Ohm, mentre la resistenza di ingresso è di un centinaio di Ohm.
Per porre il dipolo in risonanza alla frequenza di lavoro, e dunque massimizzare la potenza trasmessa, si può porre un carico capacitivo sul dipolo che annulli la reattanza induttiva, oppure più spesso si sceglie di porre la lunghezza del dipolo leggermente più piccola del valore lambda/2, in modo che lo zero della reattanza (che come si vede in figura al momento si trova verso i 2.2GHz) si trovi sulla frequenza di lavoro.

Optando per la seconda soluzione, si è utilizzato uno script messo a disposizione da Matlab, denominato **dipole_tuner.m :** tale script altro non fa che prendere in input un oggetto dipolo, parametri come la frequenza e la banda di lavoro (anche questo parametro impostato all'inizio del progetto), ed applicare un algoritmo iterativo che diminuisca progressivamente ad ogni iterazione la lunghezza del dipolo lambda/2 di una certa quantità (passata come parametro) sin quando la reattanza d'antenna, alla frequenza di lavoro, diventa trascurabile (l'utility tiene conto anche della banda di lavoro: se essa è molto larga, lo zero della reattanza non verrà posto alla frequenza scelta, ma sarà impostata in modo che all'interno delle frequenze di interesse la reattanza sia minima (potendosi questa annullare in una sola frequenza e non in un range esteso). Di seguito sono riportati i parametri ottenuti per il dipolo lambda/2, ed il dipolo finale risonante ottenuto mediante l'utility dipole_tuner. È infine riportata l'impedenza di ingresso del dipolo restituito da dipole_tuner. Si noti come il dipolo risonante abbia una lunghezza leggermente inferiore rispetto a quella del dipolo lambda/2, e come esso presenti reattanza nulla alla frequenza richiesta di circa 2.4GHz.

```
dip_elem =
  dipole with properties:

       Length: 0.0625
        Width: 0.0031
    FeedOffset: 0
         Tilt: 0
     TiltAxis: [1 0 0]
         Load: [1×1 lumpedElement]
```

Figura 3 - Dipolo lambda/2

```
dip_elem =
  dipole with properties:

       Length: 0.0583
        Width: 0.0031
    FeedOffset: 0
         Tilt: 0
     TiltAxis: [1 0 0]
         Load: [1×1 lumpedElement]
```

Figura 4 – Dipolo risonante

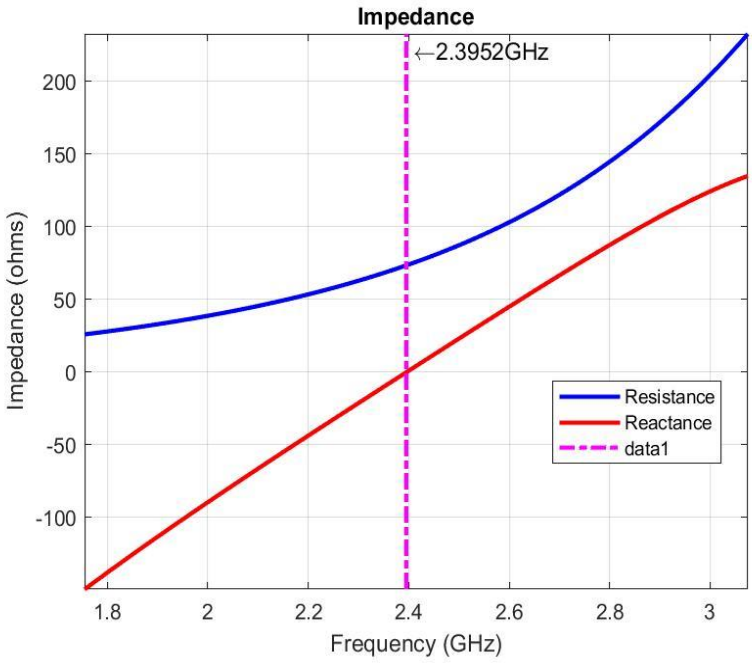


Figura 5 - Impedenza di ingresso del dipolo risonante

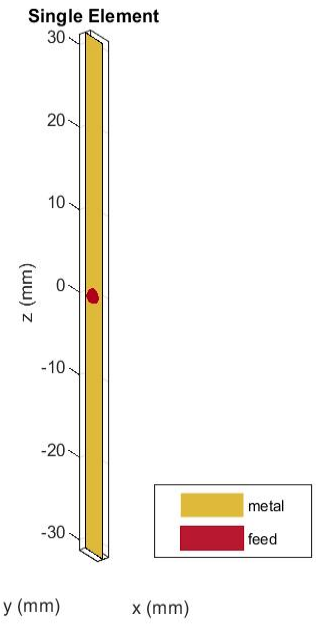


Figura 6 - Rappresentazione del dipolo mediante la funzione show

Si noti come la resistenza di ingresso del dipolo (che costituisce la resistenza di radiazione, non essendo tenute in conto le perdite) è scesa intorno alla cinquantina di Ohm alla frequenza di lavoro adesso che abbiamo posto il dipolo in risonanza alla frequenza di 2.4GHz (mentre teoricamente la resistenza di radiazione di un dipolo lambda mezzi è 73 Ohm appunto, quella di un dipolo risonante che è leggermente più corto di lambda/2 è circa 65 Ohm, quindi ci ritroviamo con il dato ottenuto dalla simulazione).

Sono stati plottati anche la distribuzione di corrente sul dipolo mediante la funzione **current**; il pattern di radiazione nello spazio tridimensionale mediante la funzione **pattern**; il diagramma di radiazione nel piano azimutale (denominato anche H-Plane, è il piano xy, in cui è variabile l'angolo azimutale ϕ) ed il diagramma di radiazione nel piano di elevazione (denominato anche E-Plane, è il piano xz, in cui è variabile l'angolo θ, che è il complementare dell'angolo di elevazione) mediante le funzioni **patternAzimuth** e **patternElevation**; infine è stato calcato il **beamwidth**, cioè l'ampiezza angolare del fascio d'onda emanato dal dipolo; il beamwidth è definito come l'apertura angolare relativa ai punti in cui la densità di potenza irradiata dal dipolo scende di 3dB rispetto alla densità di potenza massima, proprio come la banda passante di un sistema è definita alle frequenze di taglio, punti del grafico della risposta in frequenza in cui il guadagno scende di 3dB rispetto al guadagno massimo.
Come si può notare il beamwidth del dipolo è abbastanza grande, circa 78°. Ciò conferma quanto già sappiamo, e cioè che il dipolo è un'antenna poco direttiva.
Si noti come ultima cosa come la corrente sia nulla alle estremità del dipolo, mentre massima al centro, come previsto dalla teoria, in quanto il dipolo lambda mezzi è un'antenna ad **onda stazionaria.**

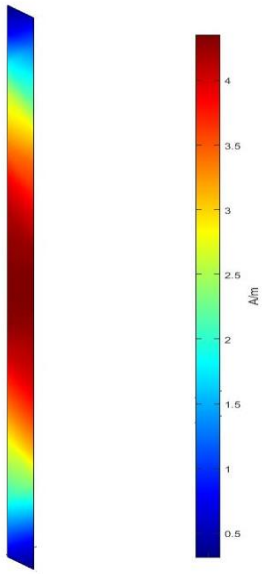


Figura 7 - Distribuzione di corrente sul dipolo

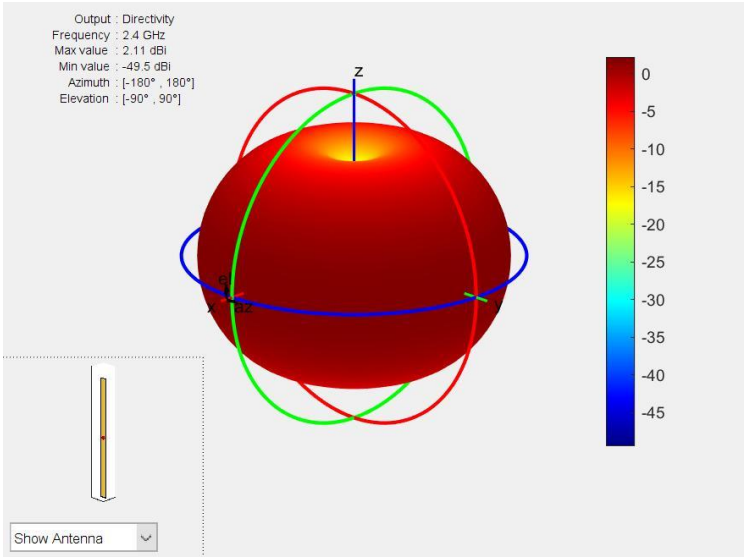


Figura 8 - Pattern di radiazione del dipolo

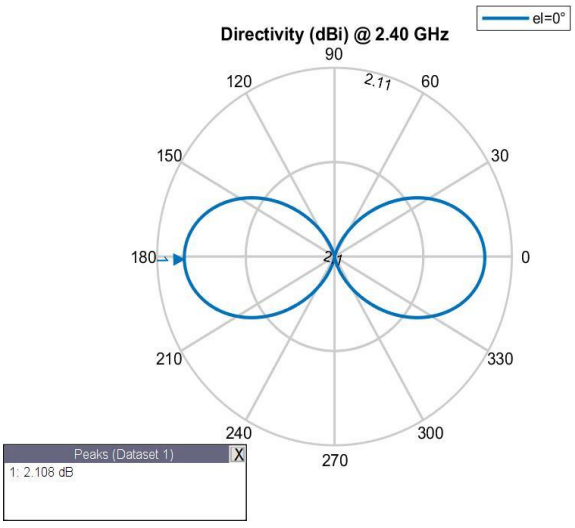


Figura 9 - Diagramma di radiazione piano azimutale (xy)

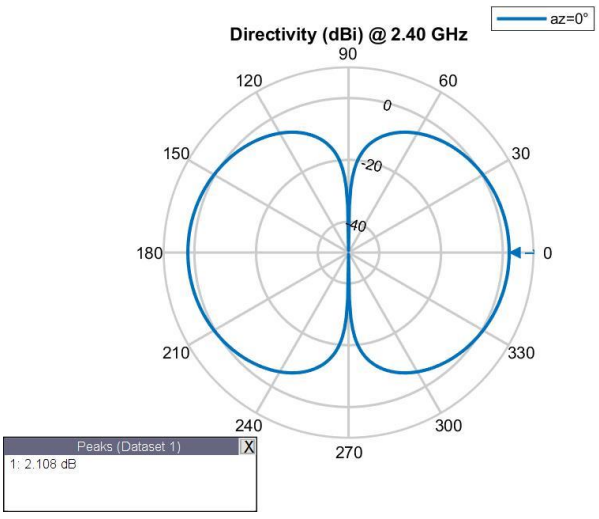


Figura 10 - Diagramma di radiazione piano di elevazione (xz)

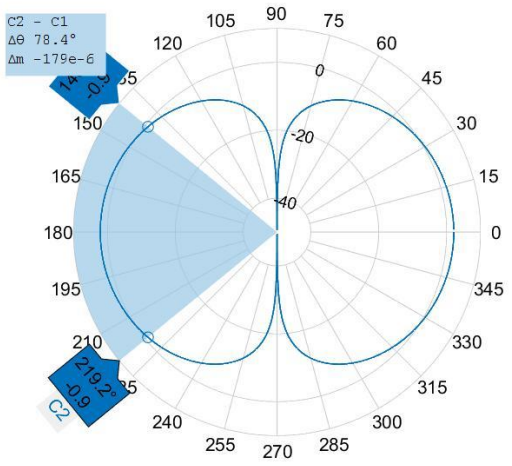


Figura 11 - Diagramma restituito dalla funzione beamwidth

CREAZIONE DELL'ARRAY COLLINEARE DI DIPOLI; SUA CONFIGURAZIONE E SIMULAZIONE COME BROADSIDE, ENDFIRE IN AVANTI, ENDFIRE ALL'INDIETRO

Una volta creato l'elemento che andrà a costituire la nostra antenna array (in questo caso il dipolo λ/2 risonante), si è creata l'array lineare in questione mediante la funzione **linearArray** dell'Antenna Toolbox, impostando programmaticamente parametri quali il tipo di elemento di cui è costituita l'array, il numero di elementi, la distanza fra ogni elemento, e l'ampiezza in tensione del segnale applicato a ciascun elemento. Tutte queste quantità possono essere variate ad ogni esecuzione, in quanto sono tutte state parametrizzate via codice, e possono essere impostate nella sezione iniziale dello script.
Per l'esecuzione analizzata in questa relazione si è scelto un numero di dipoli pari a 20 (array di medie dimensioni) ed una **distanza tra i singoli elementi pari a λ/4** (scelta che permette di **massimizzare il guadagno direttivo massimo**, come riportato in letteratura, avvicinandosi parecchio al limite teorico di una densità di potenza irradiata dall'array nella direzione di massima radiazione N^2 volte maggiore rispetto a quella che si otterrebbe col singolo elemento, con N numero di elementi dell'array).

La questione della distanziatura fra gli elementi irradianti nell'array collineare è di una certa importanza, in quanto una distanza troppo piccola fra i vari elementi porterebbe teoricamente ad avere un guadagno direttivo massimo (direttività) maggiore, ma in realtà così non avviene in quanto a piccole distanze tra gli elementi iniziano ad assumere una certa importanza gli effetti di **mutua induzione** fra i singoli dipoli, ed il guadagno direttivo massimo anziché innalzarsi si abbassa (ciò è dovuto al fatto che le transimpedenze tra un dipolo e l'altro diventano in tal caso comparabili in modulo alle impedenze di ingresso dei singoli dipoli, e quindi avremo **scambio di energia elettromagnetica fra i vari dipoli** dell'array non trascurabile, e dovendo esserci comunque un bilancio delle potenze fra quella messa a disposizione del generatore e le potenze in uscita dai singoli dipoli, quest'ultima comprendente quella irradiata nello spazio e quella irradiata verso altri dipoli, avremo che tutto ciò avviene a scapito della potenza irradiata dalla struttura in far-field).

Allo stesso tempo tuttavia una distanziatura troppo elevata comporta la comparsa di lobi aggiuntivi nella funzione di direttività, di ampiezza pari a quella del lobo principale (tali lobi prendono il nome di **grating lobes**); ciò comporta dunque la ridistribuzione della potenza irradiata su più direzioni anziché su una sola, con l'ovvio svantaggio di un minore guadagno direttivo massimo rispetto al caso di presenza di un solo lobo principale.
In ricezione invece una distanziatura di medie dimensioni è desiderabile (sempre evitando però di dare origine a gratin globes), in quanto si diminuisce il grado di correlazione fra il rumore captato dai singoli elementi, abbattendo così il rumore complessivo captato ed aumentando il rapporto S/N (Signal to Noise Ratio, SNR).

Per simulare i vari tipi di comportamento della struttura (broadside, endfire e phased array) si è semplicemente configurato correttamente il vettore **PhaseShift**, membro dell'oggetto linearArray creato; tale vettore è composto da N scalari, dove lo scalare i-esimo indica semplicemente la fase della corrente di alimentazione del dipolo i-esimo.
Ponendo uno sfasamento nullo $\alpha_0 = 0$ fra le correnti di alimentazione di ciascun elemento si è configurata l'array come **broadside**, mentre si è imposto uno sfasamento progressivo pari ad $\alpha_0 = -\beta d$ fra ciascun elemento per configurare la stessa struttura come **endfire** che "spara in avanti", ed uno sfasamento progressivo pari ad $\alpha_0 = +\beta d$ per configurare la struttura come **endfire** che "spara all'indietro".
Per ognuna di queste configurazioni si sono tracciati i pattern di radiazione 3D, il pattern di radiazione nel piano azimutale (cioè per θ = 0° e ϕ variabile in [0°, 360°]) che fa vedere chiaramente quando la struttura è configurata come endfire e quando come broadside, ed infine si è plottato il pattern di radiazione in un sistema cartesiano, per mostrare il tipico andamento del tipo sin(Nx)/sin(x) previsto dalla teoria.

Pattern di radiazione per configurazione broadside:

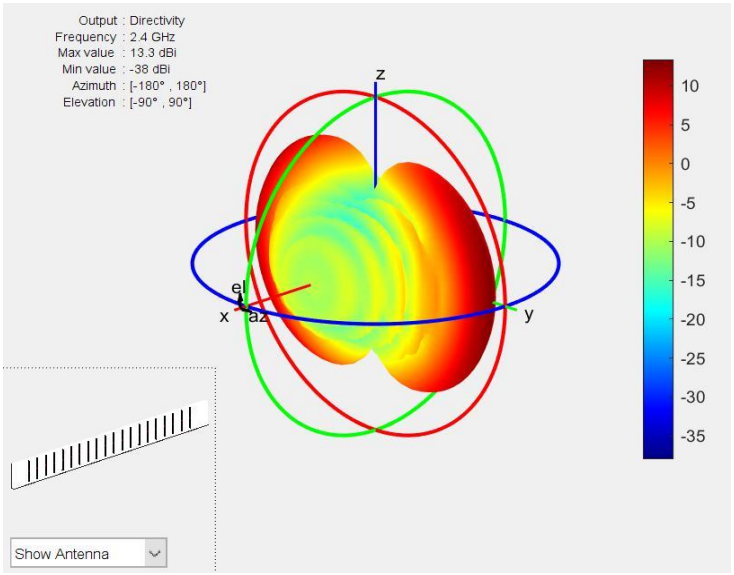


Figura 12a - Pattern di radiazione 3D view - broadside array

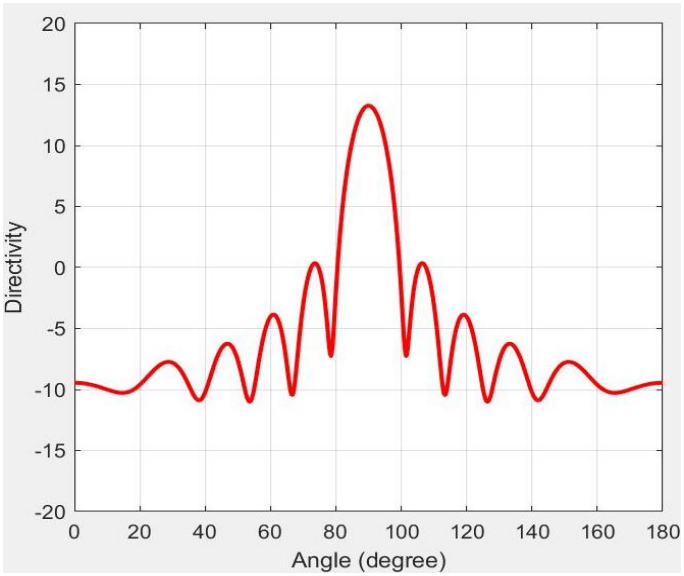


Figura 12b – Funzione direttività (non normalizzata, in dBi) nel piano azimutale, in coordinate cartesiane - broadside array

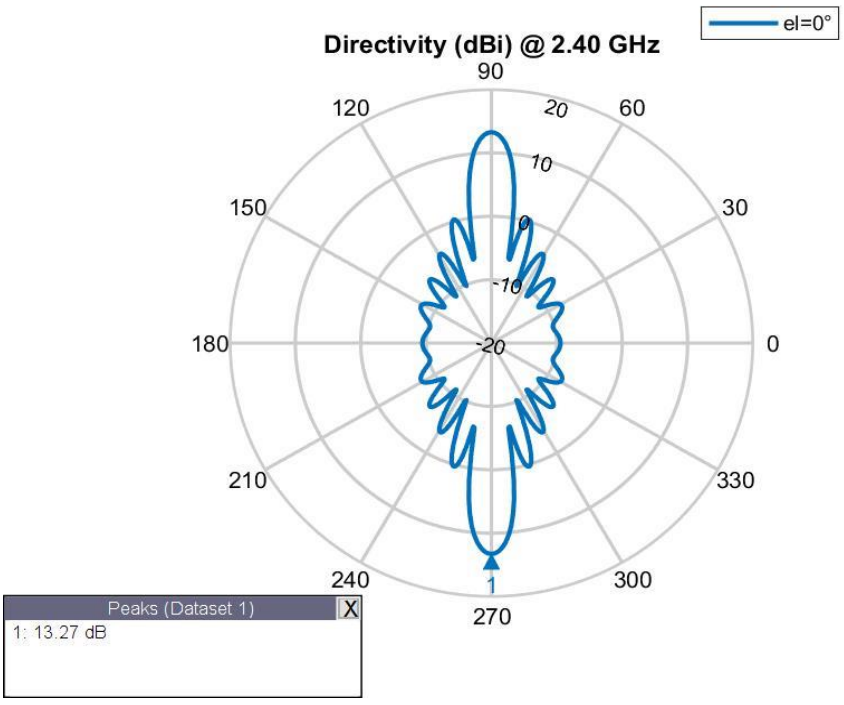


Figura 12c - Funzione direttività (non normalizzata, in dBi) nel piano azimutale, plot polare - broadside array

Pattern di radiazione per la configurazione endfire “in avanti”:

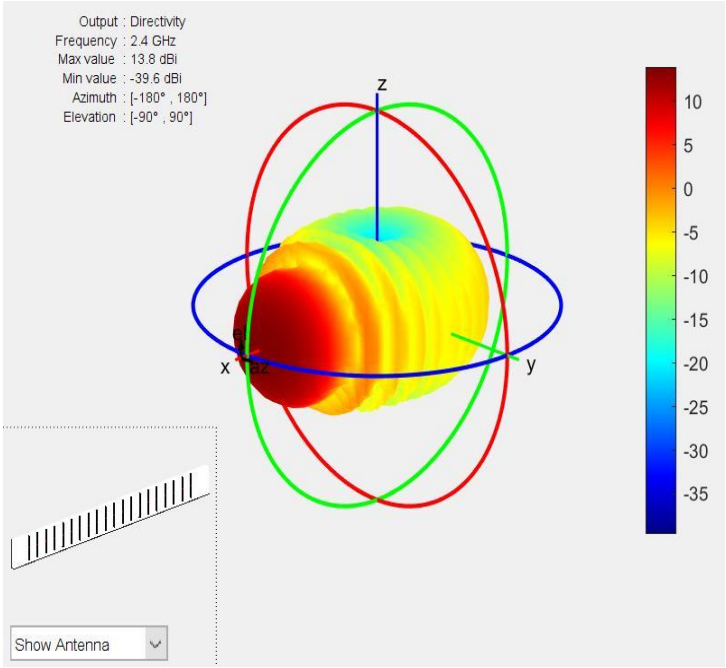


Figura 13a - Pattern di radiazione 3D view - endfire array “in avanti”

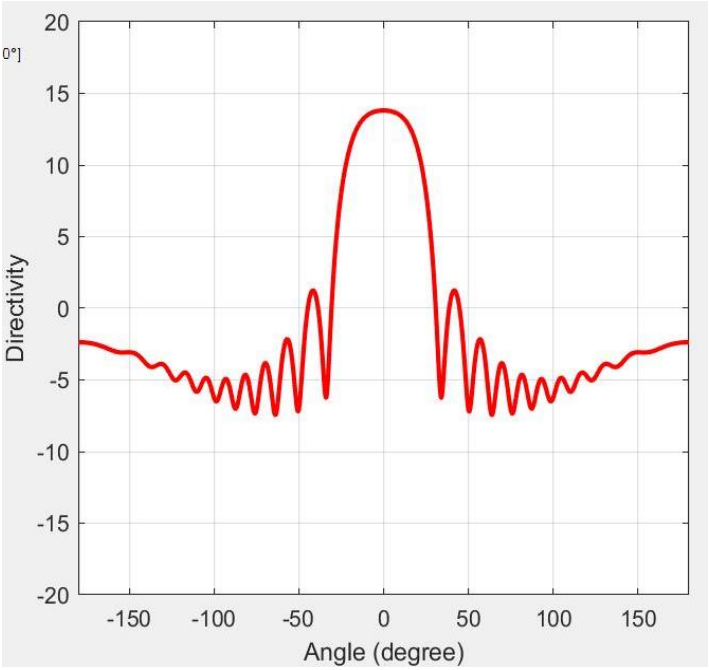


Figura 13b – Funzione direttività (non normalizzata, in dBi) nel piano azimutale, in coordinate cartesiane - endfire array “in avanti”

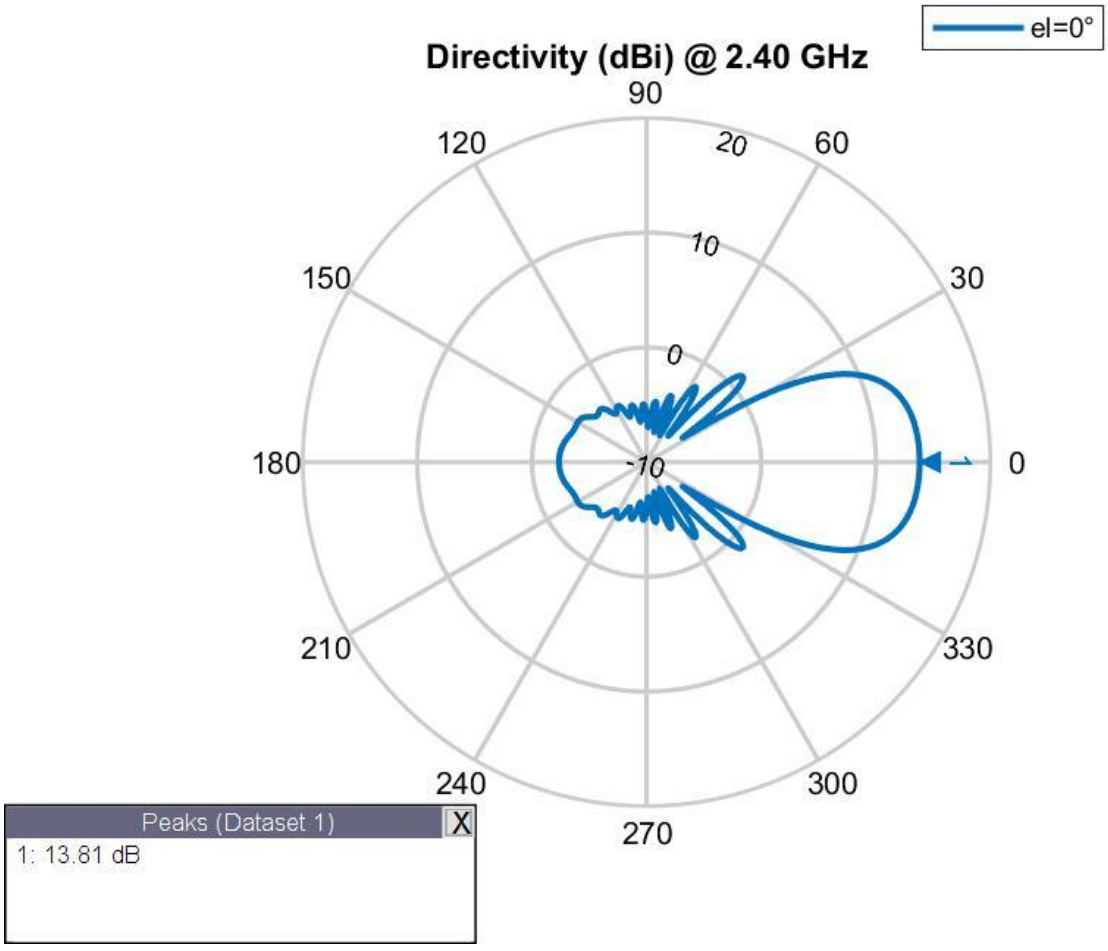


Figura 13c - Funzione direttività (non normalizzata, in dBi) nel piano azimutale, plot polare - endfire array “in avanti”

Pattern di radiazione per la configurazione endfire “all’indietro”:

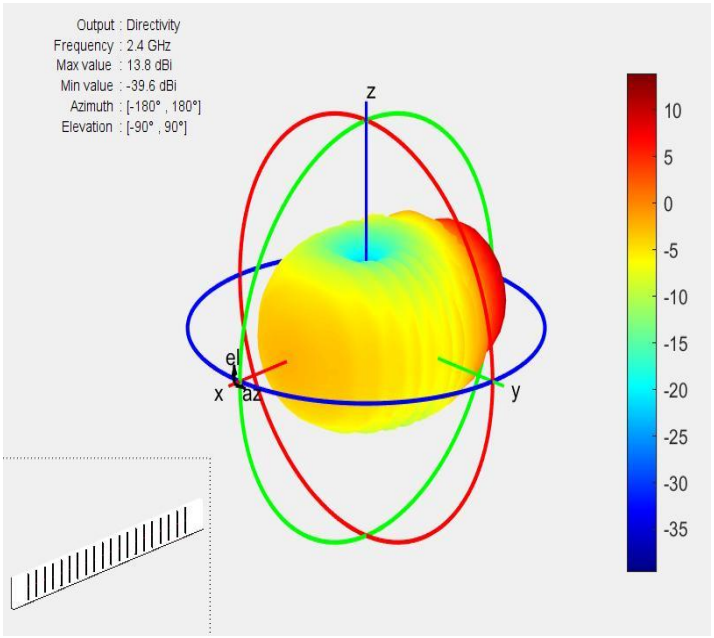


Figura 14a - Pattern di radiazione 3D view - endfire array “all’ indietro”

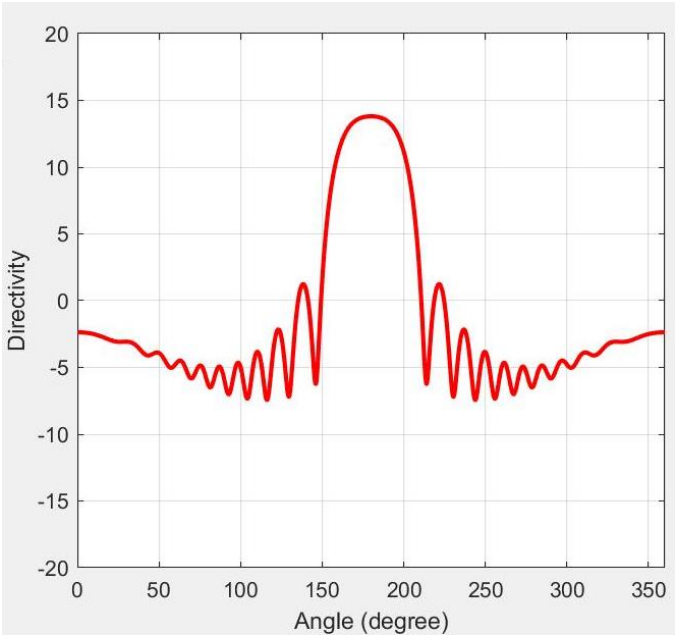


Figura 14b – Funzione direttività (non normalizzata, in dBi) nel piano azimutale, in coordinate cartesiane - endfire array “all’ indietro”

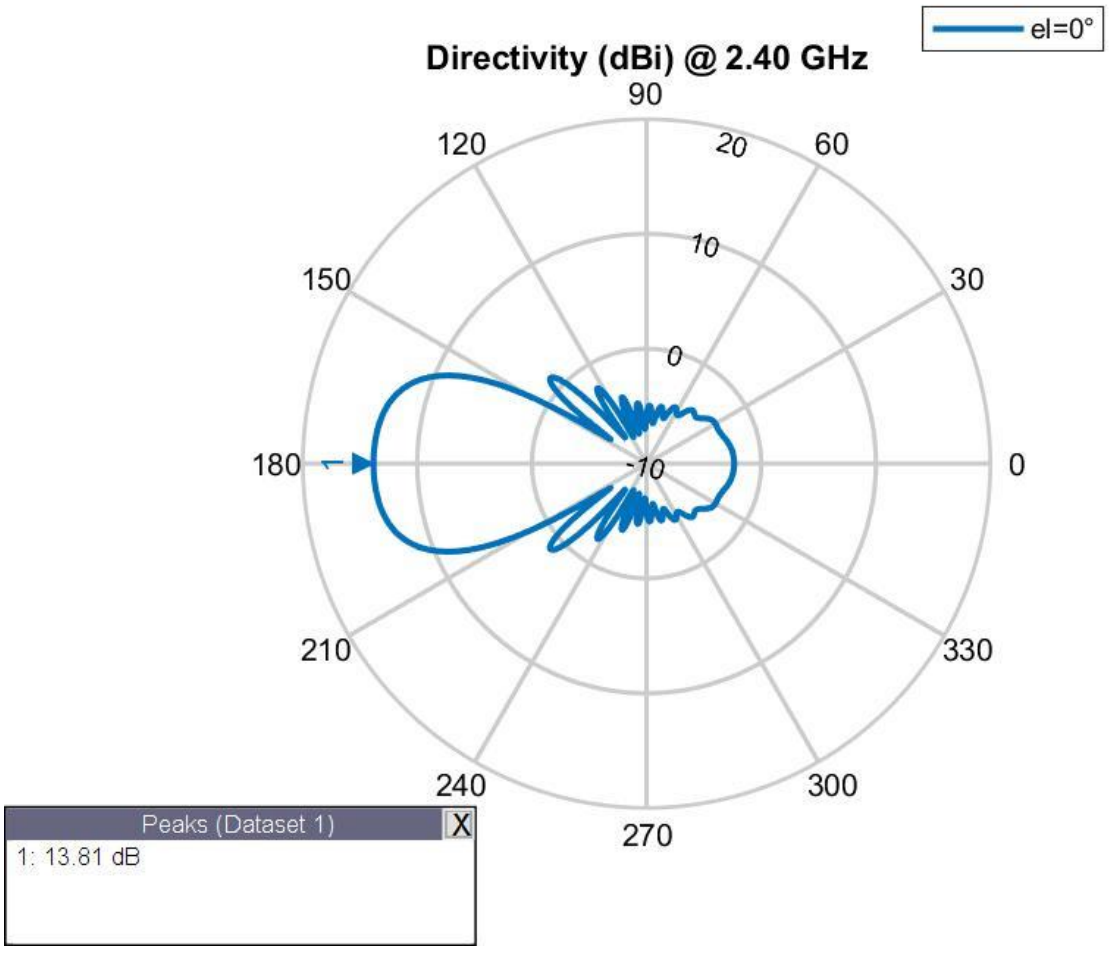


Figura 14c - Funzione direttività (non normalizzata, in dBi) nel piano azimutale, plot polare - endfire array “all’ indietro”

OSSERVAZIONI SUI RISULTATI OTTENUTI:

Tutti i risultati ottenuti dalle simulazioni sono in accordo con quanto previsto dalla teoria. Nello specifico si noti come l'array irradi con massima intensità nel piano azimutale nelle direzioni $\theta = \pm 90°$ quando configurata come broadside, nella direzione $\theta = 0°$ quando configurata come endfire "in avanti" e nella direzione $\theta = 180°$ quando configurata come endfire "all'indietro".

Si può osservare inoltre come la massima densità di potenza irradiata nella direzione di massimo sia all'incirca uguale in tutte e tre le configurazioni, intorno ai 13dBi, con uno scarto di soli 0.6dBi fra la configurazione broadside e quelle endfire. Ci si poteva aspettare che le configurazioni endfire avessero un massimo più pronunciato rispetto a quelle broadside, essendovi due lobi principali nelle broadside e solo un lobo nelle endfire su cui poter concentrare più potenza; tuttavia come si può notare bene dai pattern di direttività polare nel piano azimutale **l'Half Power BeamWidth (HPBW)** delle due configurazioni **endfire** è **molto più ampio** rispetto a quello della configurazione broadside (circa il doppio): è come se nella tipologia di array endfire i due lobi della configurazione broadside si uniscano, formando un unico lobo principale avente all'incirca stesso massimo, ma HPBW doppio rispetto a quello della configurazione broadside.
Il fatto che effettivamente i due lobi della configurazione broadside si uniscano a formare un unico lobo nella configurazione endfire può esser visualizzato nella simulazione successiva, in cui si è simulata la modalità "phased array" dell'antenna.

Da notare infine come il guadagno massimo dell'array, composta di N = 20 elementi, sia di circa 10dBi più elevato di quello del singolo elemento. Anche ciò è in accordo con quanto previsto dalla teoria: infatti sappiamo che in un' array uniforme il guadagno direttivo massimo, in numero puro, è pari ad N volte il guadagno del singolo elemento. Ciò in dBi si traduce in un aumento di $10*\log(N)$ dBi $= 10*\log(20) = 13$dBi della potenza emessa dall'array rispetto alla potenza del singolo elemento, più o meno in accordo con i 10dBi in più trovati sulla simulazione (da tener conto infatti che il risultato teorico $G_{arr} = N*G_{elem}$ è valutato nel "best case", in cui non sono valutate assolutamente effetti quali le mutue induzioni fra i singoli elementi).

VARIAZIONE DELLO SFASAMENTO PROGRESSIVO α_0 NEL RANGE [-βd, +βd] : PHASED ARRAY ANTENNA

L'ultima simulazione effettuata riguarda la configurazione dell'antenna come phased array: variando lo sfasamento progressivo α_0 nel range del cosiddetto "spazio del visibile della schiera" [-βd, +βd] è stato possibile visualizzare il pattern di radiazione in modalità 3D e la funzione di direttività non normalizzata in dBi nel piano azimutale sia in coordinate cartesiane che polari al variare di α_0 : ciò ha permesso di creare svariati grafici per diversi valori di α_0, poi convertiti in immagini e queste ultime impacchettate ad ogni iterazione del ciclo for che genera i grafici in questione in tre diversi file GIF per ogni tipo di grafico, creando così delle brevi animazioni che mostrano come i lobi principali si muovano al variare dello sfasamento progressivo α_0.

Nelle GIF create si può vedere chiaramente come variando α_0 nel range [-βd, +βd] l'array passi da un pattern di radiazione di tipo endfire "in avanti" (per $\alpha_0 = -\beta d$), ad un pattern di radiazione di tipo broadside (per $\alpha_0=0$), ad un pattern di radiazione di tipo endfire "all'indietro" (per $\alpha_0 = +\beta d$).

Tali animazioni fanno comprendere a pieno il funzionamento di una phased array, e di come questa possa dirigere gran parte della potenza elettromagnetica trasmessa nella direzione desiderata, semplicemente controllando il parametro di fase α_0.

Le GIF generate non possono essere riportate in questo documento in quanto né i formati .docx né il formato PDF supportano animazioni GIF o altro tipo di contenuto interattivo.
Esse per comodità del lettore finale sono state generate ed incluse nella stessa cartella in cui è contenuto questo documento; è comunque possibile generarle da zero partendo dal codice Matlab scritto (esse verranno salvate nella stessa cartella in cui è presente il file Matlab che si andrà ad eseguire).

Allo stesso modo tutte le simulazioni effettuate in questo lavoro possono essere replicate eseguendo il codice Matlab allegato; tuttavia per quest'ultima simulazione sulle phased array potrebbe esser necessario disporre di un calcolatore con una buona potenza di calcolo per replicare i risultati, in quanto per la creazione delle animazioni vengono create un gran numero di immagini dai plot (già di per sé onerosi dal punto di vista computazionale), ed il processo potrebbe prendere troppo tempo su macchine più datate.

Nelle pagine successive sono riportate conclusioni ed osservazioni relative al progetto, nonché il codice Matlab scritto.

CONCLUSIONI E POSSIBILI SVILUPPI SUCCESSIVI

Tale progetto ha permesso di comprendere appieno la differenza fra array broadside ed endfire, e come queste in realtà ricadano nel caso delle phased array.

La visualizzazione animata dei pattern di radiazione di una phased array illustra in maniera esplicativa il funzionamento di tale tecnologia, nonché come essa venga applicata (sebbene in forma molto più avanzata) nei moderni standard di telecomunicazioni, quali il 4g ed il più recente 5g.

Possibili sviluppi successivi di tale lavoro potrebbero essere:

- Una simulazione più realistica di tale array, tenendo in conto gli effetti di mutua induzione fra i vari elementi; una simulazione di questo tipo è fattibile in Matlab, utilizzando un tipo di simulazione che il software denomina **"Embedded Element Solution"**: la simulazione che abbiamo effettuato infatti è stata ottenuta mediante sovrapposizione dei pattern di radiazione dei singoli elementi, senza tener conto degli effetti di mutua induzione tra di essi. Ovviamente è una soluzione un po' approssimata; la simulazione ad elementi embedded invece simula gli effetti di mutua induzione facendo irradiare ogni singolo elemento uno per volta, cortocircuitando tutti gli altri al contempo; in tal modo l'elemento che sta irradiando "vede" tutti gli altri elementi metallici, ed il pattern di radiazione terrà conto degli effetti di mutua induzione fra l'elemento che sta irradiando e tutti gli altri elementi "passivati". Iterando questo tipo di simulazione per tutti gli elementi dell'array e sommando i campi ottenuti, si ottiene il pattern di radiazione complessivo della struttura, che tiene così conto anche degli effetti di mutua induzione. In pratica quest'algoritmo altro non è che un'applicazione nel "settore" elettromagnetico del principio di sovrapposizione degli effetti, che ci dice che passivando tutti i generatori di una rete e facendo agire solo uno per volta, possiamo sommare i singoli effetti ottenuti facendo agire i singoli generatori per ottenere l'effetto complessivo (il campo elettrico o magnetico irradiato in questo caso) che si ottiene quando tutti i generatori sono accesi.

- Un ulteriore sviluppo potrebbe essere lo studio di un' array avente geometria differente, per esempio rettangolare; oppure un' array costituita da elementi differenti dai dipoli, per esempio si potrebbe prendere per singolo elemento un' antenna patch realizzata su PCB.

- Particolarmente interessante sarebbe l'utilizzo del Phased Array System Toolbox messo a disposizione da Matlab, dedicato esclusivamente alla progettazione, sintesi, design e simulazione di Phased Array; esso si integra perfettamente inoltre con toolbox quali il DSP System Toolbox, il Signal Processing Toolbox, ed il Communications Toolbox, rendendo possibile la progettazione e la simulazione in Matlab e Simulink di sistemi di radiocomunicazione avanzati che utilizzino Phased Array Antennas come riceventi e/o trasmittenti ed algoritmi di Spatial Digital Signal Processing applicati nel senso già accennato alle Phased Array, col fine di ottimizzare il funzionamento dell'intero sistema e sfruttare al massimo le risorse fisiche messe da esso a disposizione, minimizzando i costi di produzione.

Nelle pagine seguenti è riportato il codice Matlab utilizzato nella simulazione.
Esso è disponibile in allegato assieme al presente documento sia nella versione ".mlx", Matlab Live-Script che nella versione classica ".m"

```
% INIT WORKSPACE AND COMMAND WINDOW
clear;
clc;
```

```
% SET WORK FREQUENCY AND OTHER PARAMETERS
f0 = 2.4*10^9;                  % Standard WiFi, Bluetooth, etc frequency
WLratio = 0.05;                 % Ratio Width/Length of the single dipole element
BWrel = 0.05;                   % Bandwidth percentual relative to center freq f0
numelem = 20;                   % Number of elements that compose the antenna array
dspacingrel = 0.25;             % Spacing between each element of the array, in
                                % wavelengths units
```

```
% EVALUATE WAVELENGTH, SINGLE ELEMENT PARAMETERS AND OTHER STUFF
c = physconst('lightspeed');
lambda0 = c/f0;
ldip = lambda0/2;
wdip = WLratio*ldip;
fmin = f0 - BWrel*f0/2;
fmax = f0 + BWrel*f0/2;

alpha_broadside = 0;            % Parameter that controls the maximum direction in
                                % the broadside configuration
alpha_backfire = rad2deg( (2*pi)*(dspacingrel) );
alpha_endfire = - alpha_backfire;

indexvect = (0:1:(numelem -1));
phasevector_broad = indexvect*alpha_broadside;
phasevector_efire = indexvect*alpha_endfire;
phasevector_bfire = indexvect*alpha_backfire;
```

```
% CREATE A STRIP DIPOLE ELEMENT, AND TUNE IT TO THE SPECIFIED BANDWIDTH RANGE;
% THEN SHOW RADIATION PATTERN AND CURRENT DISTRIBUTION ON THE SINGLE ELEMENT
dip_elem = dipole('Length', ldip, 'Width', wdip)

fig_1a_dipshow = figure('Name','Single Element');
dip_elem.show;
axis tight
title('Single Element');

fig_1b_zin = figure('Name', 'Impedance of the single element');
impedance(dip_elem, linspace(1e9,3e9,101) );
title('Impedance of the single element');

fig_1c_zintun = figure('Name', 'Impedance of the single tuned element');
dip_elem = dipole_tuner(dip_elem, f0, fmin, fmax, 0.01, 0.001)
title('Impedance of the single tuned element');

fig_1d_curr = figure('Name', 'Current Distribution on the single element');
current(dip_elem,f0);
title('Current Distribution on the single element');
```

```
fig_1e_ptrndip = figure('Name', 'Radiation Pattern of the single element');
pattern(dip_elem,f0);
title('Radiation Pattern of the single element');

fig_1f_azptrndip = figure('Name', 'Azimuthal radiation pattern of the single element');
patternAzimuth(dip_elem,f0);
title('Azimuthal radiation pattern of the single element');

fig_1g_elptrndip = figure('Name', 'Elevation radiation pattern of the single element');
patternElevation(dip_elem,f0);
title('Elevation radiation pattern of the single element');

fig_1h_bmwdip = figure('Name', 'Beamwidth of the single element');
beamwidth(dip_elem, f0, 0, 1:0.1:360);
title('Beamwidth of the single element');
```

```
% CREATE AN ANTENNA ARRAY OF N ELEMENTARY DIPOLE, AND CONFIGURE IT AS BROADSIDE
AntArray = linearArray('Element', dip_elem, 'NumElements', numelem);
AntArray.ElementSpacing = dspacingrel*lambda0;
AntArray.AmplitudeTaper = 1;

fig_2a_brsarr = figure('Name', 'Antenna Array Structure');
AntArray.show();
title('Antenna Array Structure');

fig_2b_brsarrlt = figure('Name', 'Array Layout');
AntArray.layout();
title('Array Layout');

fig_2c_brsarrptrn = figure('Name', 'Array Broadside radiation pattern');
AntArray.pattern(f0);
title('Array Broadside radiation pattern');

fig_2d_brsarrazptrn = figure('Name', 'Array Broadside radiation azimuth-plane pattern');
patternAzimuth(AntArray, f0);
title('Array Broadside radiation azimuth-plane pattern');

fig_2e_brsrectptrn = figure('Name', 'Array Broadside rect radiation pattern');
azimuth = 0:0.25:180;
pattern(AntArray, f0, azimuth, 0, 'CoordinateSystem', 'rectangular');
axis([0 180 -20 20]);
```

```
% RECONFIGURE THE PREVIOUS ARRAY AS ENDFIRE
AntArray.PhaseShift = phasevector_efire;

fig_3a_efptrn = figure('Name', 'Array Endfire radiation pattern');
AntArray.pattern(f0);
title('Array Endfire radiation pattern');

fig_3b_efazptrn = figure('Name', 'Array Endfire Azimuth-plane radiation pattern');
AntArray.patternAzimuth(f0);
```

```
title('Array Endfire Azimuth-plane radiation pattern');

fig_3c_efrectptrn = figure('Name', 'Array Endfire rect radiation pattern');
azimuth = -180:0.25:180;
pattern(AntArray, f0, azimuth, 0, 'CoordinateSystem', 'rectangular');
axis([-180 180 -20 20]);
```

```
% RECONFIGURE THE PREVIOUS ARRAY AS BACKFIRE AND PLOT PATTERNS
AntArray.PhaseShift = phasevector_bfire;

fig_4a_bckfptrn = figure('Name', 'Array Backfire radiation pattern');
AntArray.pattern(f0);
title('Array Backfire radiation pattern');

fig_4b_bckfazptrn = figure('Name', 'Array Backfire Azimuth-plane radiation pattern');
AntArray.patternAzimuth(f0);
title('Array Backfire Azimuth-plane radiation pattern');

fig_4c_bckfrectptrn = figure('Name', 'Array Backfire rect radiation pattern');
azimuth = 0:0.25:360;
pattern(AntArray, f0, azimuth, 0, 'CoordinateSystem', 'rectangular');
axis([0 360 -20 20]);
```

```
% ARRAY SCAN: ALPHA FROM -beta*dspacing TO beta*dspacing (FROM ENDFIRE TO BROADSIDE TO
% BACKFIRE)

% Creating vectors with elements from one vector to another spaced by 5 deg
NumSteps = 50;
delta = 180/NumSteps;
alpha_scan = alpha_endfire:delta:alpha_backfire;
phasevector_scan = phasevector_efire;
azimuth = 0:0.25:360;

fig_temp_a = figure('visible', 'off');
fig_temp_b = figure('visible', 'off');
fig_temp_c = figure('visible', 'off');

gif_3dview = 'phased_array_scan_3dview.gif';
gif_polar_azimuth = 'phased_array_scan_polar.gif';
gif_cartesian_azimuth = 'phased_array_scan_cartesian.gif';


for i = 1:(NumSteps+1)

    % plot the azimuth-plane pattern for the current phasevector_scan
    phasevector_scan = indexvect*alpha_scan(1,i);
    AntArray.PhaseShift = phasevector_scan;

    % Draw azimuth polar pattern
    figure(fig_temp_a);
    AntArray.patternAzimuth(f0);
```

```
    % get the image file from the plot; convert it to proper format for gif
    drawnow
    frame_pol = getframe(fig_temp_a);
    img_pol = frame2im(frame_pol);
    [imind_pol,cm_pol] = rgb2ind(img_pol,256);

    % Write to the GIF File
      if i == 1
          imwrite(imind_pol,cm_pol,gif_polar_azimuth,'gif', 'Loopcount',inf);
      else
          imwrite(imind_pol,cm_pol,gif_polar_azimuth,'gif','WriteMode','append');
      end


    % draw 3d view pattern
    figure(fig_temp_b);
    AntArray.pattern(f0);

    % get the image file from the plot; convert it to proper format for gif
    drawnow
    frame_3dv = getframe(fig_temp_b);
    img_3dv = frame2im(frame_3dv);
    [imind_3dv,cm_3dv] = rgb2ind(img_3dv,256);

    % Write to the GIF File
      if i == 1
          imwrite(imind_3dv,cm_3dv,gif_3dview,'gif', 'Loopcount',inf);
      else
          imwrite(imind_3dv,cm_3dv,gif_3dview,'gif','WriteMode','append');
      end


    % cartesian directivity pattern
    figure(fig_temp_c);
    pattern(AntArray, f0, azimuth, 0, 'CoordinateSystem', 'rectangular');
    axis([0 360 -20 20]);

    % get the image file from the plot; convert it to proper format for gif
    drawnow
    frame_cart = getframe(fig_temp_c);
    img_cart = frame2im(frame_cart);
    [imind_cart,cm_cart] = rgb2ind(img_cart,256);

    % Write to the GIF File
      if i == 1
          imwrite(imind_cart,cm_cart,gif_cartesian_azimuth,'gif', 'Loopcount',inf);
      else
          imwrite(imind_cart,cm_cart,gif_cartesian_azimuth,'gif','WriteMode','append');
      end
end
```