Simulasi MATLAB : Biconegeom.m

```
% Fungsi ini menggambarkan bentuk antena biconical secara fisik
ant_angle = 31;
R= 15;
Ha = cos((ant_angle - 1)*pi/180);
Zmax = R*Ha;
N = 60;
Zprev = 0;
for i = 1:N+1
   for j = 1:N+1
      if (Zprev < Zmax)
         Z(i,j) = Ha*R*(i-1)/N;
         term = R*(i-1)/N;
         Zprev = term;
      else
      Z(i,j) = R*cos(pi/2*(2*(i-1)/N - 1)- pi/2);
      end
      arg = 2*pi*(j-1)/N;
   Y(i,j) = term*cos(arg + pi/2);
   X(i,j) = term*cos(arg + pi);
   end
end
surf([X;X],[Y;Y],[Z;Z])
```

Simulasi MATLAB : vsrc1.m

```
% Fungsi ini menampilkan pulsa bentuk tegangan UWB Gaussian untuk
% menggerakan antena UWB.
% Menerima sebuah maximum timestep "Nmax".
% Outputnya sebuah array.
function [Vout] = Vsrc1(Nmax,dt)
% Set  time step pada 1psec
dt = 1e-12;
Nmax = 400;
% Letakkan pulsa waktu center (nstart) = 200psec
nstart = 200;
% Waktu 1/e (ndecay) diambil 10dt
ndecay = 50;
% Frekuensi pulsa center (fc) adalah 6.5GHz.
fc = 6.5*10^9;
% Pulsa amplituda maximum (Vo) diambil 1 volt
Vo = 1.0;
Vout(n) = Vo*exp(-((n-nstart)/ndecay)^2)*cos(2*pi*fc*(n-nstart)*dt);
end
plot(Vout)
%title('Pulsa bentuk Gaussian:-Vo*exp(-((n-200)/50)^2)*sin(2*pi*fc*(n-200)dt)')
xlabel('Time (psec)');
ylabel('Vout(Volt)');
title('Pulsa bentuk Gaussian: Vo*exp(-((n-200)50)^2)*cos(2*pi*fc*(n-200)dt)')
```

Simulasi MATLAB : vsrc2.m

```
% Fungsi ini menampilkan pulsa bentuk tegangan UWB Gaussian untuk
% menggerakan antena UWB.
% Menerima sebuah maximum timestep "Nmax".
% Outputnya sebuah array.
function [Vout] = Vsrc1(Nmax,dt)
% Set  time step pada 1psec
dt = 1e-12;
Nmax = 400;
% Letakkan pulsa waktu center (nstart) = 200psec
nstart = 200;
% Waktu 1/e (ndecay) diambil 10dt
ndecay = 50;
% Frekuensi pulsa center (fc) adalah 6.5GHz, center dari FCC 3,1-10,6 GHz
fc = 6.5*10^9;
% Pulsa amplituda maximum (Vo) diambil 1 volt
Vo = 1.0;
Vout(n) = Vo*exp(-((n-nstart)/ndecay)^2)*sin(2*pi*fc*(n-nstart)*dt);
end
plot(Vout)
title('Pulsa bentuk Gaussian:-Vo*exp(-((n-200)/50)^2)*sin(2*pi*fc*(n-200)dt)')
xlabel('Time (psec)');
ylabel('Vout(Volt)');
```

Simulasi MATLAB : FDTD1.m

```
% 2D, simulasi FDTD spherical dari antena UWB biconical.
% Simulasi ini menggunakan 2D dari 3D persamaan spherical Maxwell.
% Komponen medan listrik dan medan magnet dikalkulasi untuk propagasi
% gelombang.
% Eksitasi atau pembangkit gelombang dapat dipilih antara gelombang sinus
% atau pulsa Gaussian.
function [Et] = FDTD1(tmax)
% Inisialisasi standar konstanta ruang bebas.
uo = 4*pi*1e-7;
eo = 1e-9/(36*pi);
Z = 120*pi;
c = 3e+8;
%tmax = 10;

% Defenisikan dimensi antena. Ini adalah 15dr (1 lambda_nom) panjangnya
% dengan sudut 30 derajat. Tambahkan nilai satu sebagai referensi matriks.
ant_length = 15;
ant_angle = 31;

% Defenisikan batas sisi. Rmax terjadi pada 10(lambda_nominal) dimana lambda
% nominal adalah panjang gelombang untuk frekuensi 6.5GHz. Bandwidth yang
% diharapkan adalah 3,1-10,6 GHz, dimana 3,1 GHz menjadi lambda_max dan
%10.6 GHz menjadi lambda_min.
% Jarak ini didefenisikan untuk FCC untuk transmisi UWB . Theta
% luasnya dari 0 ke 90 derajat. Disini, dr = 0.003 m (lambda_min/10), dan
% dtheta = 1 derajat. Penggunaan dt = 0.2 psec akan menyederhanakan batas
% Courant untuk semua frekuensi.
dr = 0.003; % radius sel
dth = 1.0*pi/180; % sudut sel dalam radian
Rmax = 151; % solusi radius ruang
```

```
THmax = 91; % solusi sudut ruang
b = 0.006; % radius coaxial input terluar
a = 0.003; % radius coaxial input terdalam

% Set up konstanta untuk daerah ruang:Npml=# layer,R0= koefisien refleksi
% pada susut 90 derajat, sigma_space = konduktansi pada ruang
% bebas (0 layer), sigmaM_space = konduktansi magnet pada ruang bebas, L=1
% untuk profile konduktansi linear, L=2 untuk profile konduktansi parabolik,
% untuk sebuak konduktansi geometri
R0 = 1e-14; % pilih refleksi yang diinginkan
Npml = 20; % pilih jumlah layer yang digunakan
L = 1; % pilih profile konduktiviti

% Kalkulasikan konduktiviti ruang bebas untuk profile linear atau parabolik
sigma_space = - eo*c*log(R0)/(2^(L+2)*dr*Npml^(L+1));

% Kalkulasikan konduktiviti ruang bebas untuk profile geometri
% sigma_zero = - eo*c*log(g)*log(R0)/(2*dr*(g^(Npml)-1));
% sigma_space = sigma_zero*(sqrt(g) - 1)/log(g);
sigmaM_space = uo*sigma_space/eo; % kondidi matching impedansi

% Kalkulasikan final konduktiviti untuk profile linear atau parabolik final
sigmaPML(Npml) = sigma_space*(L+1)*2^(L+1)*Npml^L;

% Kalkulasikan final konduktiviti untuk profile geometri final
% sigmaPML(Npml) = sigma_zero*(g-1)*(g^Npml)/(sqrt(g)*log(g));
sigmaMPML(Npml) = uo*sigmaPML(Npml)/eo;
for I = 1:(Npml-1)
   % sigmaPML(I) = sigmaPML(Npml)*g^(I-Npml); % untuk profile geometric
   sigmaMPML(I) = uo*sigmaPML(I)/eo;
end
```

```
% Inisialisasikan sub komponen H untuk daerah PML
for I = 1:Npml
   for J = 1:THmax
      Hpr(I,J) = 0;
      Hpt(I,J) = 0;
   end
end
% Set up pengkonduktansian permukaan antena. Antena terbuat dari tembaga
% dengan konductiviti of 5.8e+7 mho/m. ruang bebas diberi sebuah
% konduktiviti listrik kecil dan a konduktiviti magnet matching menurut
% kalkulasi PML yang telah dilakukan di atas.
sigma_cu = 5.8e+7;
for I = 1:Rmax
for J = 1:THmax
if (J == ant_angle)
if (I <= ant_length)
sigma(I,J) = sigma_cu;
sigmaM(I,J) = sigmaM_cu;
elseif (I < Rmax-Npml+1)
sigma(I,J) = sigma_space;
sigmaM(I,J) = sigmaM_space;
else
sigma(I,J) = sigmaPML(I + Npml - Rmax);
sigmaM(I,J) = sigmaMPML(I + Npml - Rmax);
end
elseif (I == ant_length)
if (J <= ant_angle)
sigma(I,J) = sigma_cu;
sigmaM(I,J) = sigmaM_cu;
sigma(I,J) = sigma_space;
sigmaM(I,J) = sigmaM_space;
end
```

```
else
if (I < Rmax-Npml+1)
sigma(I,J) = sigma_space;
sigmaM(I,J) = sigmaM_space;
else
sigma(I,J) = sigmaPML(I + Npml - Rmax);
sigmaM(I,J) = sigmaMPML(I + Npml - Rmax);
end
end
end
end
```

% Untuk inisialisasi, nolkan medan untuk semua node bebas pada masalah tersebut.

```
for I = 1:Rmax
for J = 1:THmax
if ((I >= ant_length) | (J >= ant_angle))
Er(I,J) = 0;
Et(I,J) = 0;
Hp(I,J) = 0;
end
end
end
```

% Tetapkan parameter driving pulsa Gaussian. Centernya
% 200ps out, it has a decay time of 50ps and frekuensi center
% 6.5 GHz.

```
tstart = 200e-12;
tdecay = 50e-12;
tsample = 183e-12;
once = 0;
Vocw = 0.01;
Vouwb = 0.06;
Rs = 50;
```

```
Zin = 0;
% Set up beberapa konstanta untuk waktu iterasi
t = 0;
g1 = dt/(2*uo); % g1 = 7.6e-8
g2 = dt/(2*eo);   % g2 = 0.0113
g3 = dt/(dr*eo); % g3 = 7.53
g4 = dt/(dr*uo); % g4 = 5.3e-5
sphere_factor = 1/(b*log(2));
I_factor = 2*pi*b;
volt_factor = (log(sin(ant_angle*pi/180))-log(1-cos(ant_angle*pi/180)))/log(2);
% Mulai waktu iterasi
while (t < (tmax*dt))
% Untuk pertamanya 1/2 timestep, update medan Hphi dimana saja.
% Jangan step kesingularan sumber, point ground plane,
% jalur/garis simetri, atau daerah PML.
t = t + 0.5*dt;
for I = 3:(Rmax-Npml)
for J = 2:(THmax-1)
if ((I >=ant_length) | (J >= ant_angle))
g5 = sin(J*dth)/sin((J-1)*dth);
Da = (1-sigmaM(I,J)*g1)/(1+sigmaM(I,J)*g1);
Db = g4/(1+sigmaM(I,J)*g1);
ER1 = (g5*Er(I,J+1)-Er(I,J))/((I-1/2)*dth);
ET1 = (I/(I-1))*Et(I+1,J) - Et(I,J);
Hp(I,J) = Da*Hp(I,J) + Db*(ER1 - ET1);
end
end
% Hitung medan H pada ground plane, sisi luar daerah PML
J = THmax;
for I = 3:(Rmax-Npml)
Da = (1-sigmaM(I,J)*g1)/(1+sigmaM(I,J)*g1);
Db = g4/(1+sigmaM(I,J)*g1);
```

```
ET1 = (I/(I-1))*Et(I+1,J) - Et(I,J);
Hp(I,J) = Da*Hp(I,J) - Db*ET1;
end
% Hitung medan H sepanjang garis simetri.
for I = ant_length:(Rmax-Npml)
Hp(I,1) = Hp(I,2);
end
% Update medan H di daerah PML, kecuali ground plane
layer = 1;
for I = (Rmax-Npml+1):Rmax
Dar = (1-sigmaMPML(layer)*g1)/(1+sigmaMPML(layer)*g1);
Dbr = g4/(1+sigmaMPML(layer)*g1);
Dat = Dar;
Dbt = Dbr;
for J = 2:(THmax-1)
g5 = sin(J*dth)/sin((J-1)*dth);
ER1 = (g5*Er(I,J+1)-Er(I,J))/((I-1/2)*dth);
if (I == Rmax)
ET1 = 0;
else
ET1 = (I/(I-1))*Et(I+1,J) - Et(I,J);
end
Hpr(layer,J) = Dar*Hpr(layer,J) - Dbr*ET1;
Hpt(layer,J) = Dat*Hpt(layer,J) + Dbt*ER1;
Hp(I,J) = Hpr(layer,J) + Hpt(layer,J);
end
layer = layer + 1;
% Update medan H pada daerah PML sepanjang ground plane
J = THmax;
layer = 1;
for I = (Rmax-Npml+1):Rmax
Dar = (1-sigmaMPML(layer)*g1)/(1+sigmaMPML(layer)*g1);
```

```
Dbr = g4/(1+sigmaMPML(layer)*g1);
Dat = Dar;
if (I == Rmax)
ET1 = 0;
else
ET1 = (I/(I-1))*Et(I+1,J) - Et(I,J);
Hpr(layer,J) = Dar*Hpr(layer,J) - Dbr*ET1;
Hpt(layer,J) = Dat*Hpt(layer,J);
Hp(I,J) = Hpr(layer,J) + Hpt(layer,J);
layer = layer + 1;
end
% Untuk keduanya 1/2 timestep, update medan E dimana saja. Sekali lagi,
% jangan step driver pada sumber atau point ground plane.
t = t + 0.5*dt;
% Step sumber eksitasi, pulsa Gaussian, or the steady
% state driver sinusoidal 6.5GHz.
% if (t < 1.0e-9)
% Vsrc = Vocw*cos(2*pi*fc*t); % CW source
% else
% Vsrc = 0;
% end
% Vsrc = Vouwb*exp(-((t-tstart)/tdecay)^2)*cos(2*pi*fc*(t-tstart));
% Sekarang update medan E. Step pertama driving sphere...
Vin = 0; Iins = 0;
for J = ant_angle:THmax
 Iin = I_factor*sin((J-1/2)*dth)*Hp(3,J);
 Vdrv = Vsrc - Rs*Iin;
if ((t >= tsample) & (J > ant_angle))
Vin = Vin + b*Et(3,J)*dth;
Iins = Iins + Iin;
if (J == (THmax - 1))
tsample = 1e-6;
```

```
Iins = Iins/(THmax - ant_angle - 1);
exVin = Vin
exIin = Iin
Zin = abs(Vin/Iin)
end
end
Et(3,J) = Vdrv*sphere_factor/sin((J-1/2)*dth);
end
% Sekarang step medan E untuk node bebas
for I = 4:(Rmax-Npml)
for J = 2:(THmax-1)
if ((I >= ant_length) | (J >= ant_angle))
g6 = sin((J-1/2)*dth)/sin((J-3/2)*dth);
Ca = (1 - g2*sigma(I,J))/(1 + g2*sigma(I,J));
Cb = g3/(1 + g2*sigma(I,J));
HPHI1 = (g6*Hp(I,J) - Hp(I,J-1))/((I-1/2)*dth);
HPHI2 = Hp(I-1,J) - ((I-1/2)/(I-3/2))*Hp(I,J);
Er(I,J) = Ca*Er(I,J) + Cb*HPHI1;
Et(I,J) = Ca*Et(I,J) + Cb*HPHI2;
end
end
% Hitung medan E pada ground plane
J = THmax;
for I = 4:(Rmax-Npml)
Ca = (1- g2*sigma(I,J))/(1 + g2*sigma(I,J));
Cb = g3/(1 + g2*sigma(I,J));
HPHI2 = Hp(I-1,J) - ((I-1/2)/(I-3/2))*Hp(I,J);
Er(I,J) = 0;
Et(I,J) = Ca*Et(I,J) + Cb*HPHI2;
end
% Hitung medan E sepanjang garis simetri.
for I = ant_length:(Rmax-Npml)
```

```
HPHI1 = (Hp(I,2) - Hp(I,1))/((I-1/2)*dth);
Er(I,1) = Ca*Er(I,1) + Cb*HPHI1;
Et(I,1) = 0;
end
% Update medan E di daerah PML, kecuali ground plane
layer = 1;
for I = (Rmax-Npml+1):Rmax
for J = 2:(THmax-1)
g6 = sin((J-1/2)*dth)/sin((J-3/2)*dth);
Car = (1 - g2*sigmaPML(layer))/(1 + g2*sigmaPML(layer));
Cbr = g3/(1 + g2*sigmaPML(layer));
Cat = Car;
Cbt = Cbr;
HPHI1 = (g6*Hp(I,J) - Hp(I,J-1))/((I-1/2)*dth);
HPHI2 = (Hp(I-1,J) - ((I-1/2)/(I-3/2))*Hp(I,J));
Er(I,J) = Cat*Er(I,J) + Cbt*HPHI1;
Et(I,J) = Car*Et(I,J) + Cbr*HPHI2;
end
layer = layer + 1;
end
% Hitung medan E daerah PML pada ground plane
layer = 1;
J = THmax;
for I = (Rmax-Npml+1):Rmax
Car = (1 - g2*sigmaPML(layer))/(1 + g2*sigmaPML(layer));
Cbr = g3/(1 + g2*sigmaPML(layer));
HPHI2 = (Hp(I-1,J) - ((I-1/2)/(I-3/2))*Hp(I,J));
Er(I,J) = 0;
Et(I,J) = Car*Et(I,J) + Cbr*HPHI2;
layer = layer + 1;
end
end
```

```
% Plot hasilnya
% Ubah kutub koordinat ke koordinat rectangular dan cerminkan
% data simulasi antena untuk menunjukkan +x and -x dari peradiasian
% medan.
for I = 1:Rmax
for J = 1:THmax-1
x = 151 + round((I*sin((J-1)*dth)));
x2 = 303 - x;
y = 151 + round((I*cos((J-1)*dth)));
y2 =303- y;
Ecart(x,y) = Et(I,J);
Ecart(x2,y) = Et(I,J);
Ecart(x,y2) = Et(I,J);
Ecart(x2,y2) = Et(I,J);
end
end
% Load sebuah matrik Cartesian untuk menyisipkan elemen yang hilang
for I = 1:(2*Rmax-1)
for J = 1:(2*Rmax-1)
EcartNew(I,J) = Ecart(I,J);
end
end
% Sisipkan elemen yang hilang di dalam matrik Cartesian
Imin = 2; Imax = 2*Rmax - 2; Jmin = 2; Jmax = Rmax - 1;
for I = Imin:Imax
for J = Jmin:Jmax
if ((Ecart(I,J) == 0) & (Rmax*cos((I-Rmax)*dth*91/151) + 25 >= J))
ItempLo = I - 1;
ItempHi = I + 1;
while ((Ecart(ItempLo,J) == 0) & (ItempLo > 1))
ItempLo = ItempLo - 1;
end
```

```
while ((Ecart(ItempHi,J) == 0) & (ItempHi < 2*Rmax -1))
ItempHi = ItempHi + 1;
end
M = Ecart(ItempLo,J);
N = Ecart(ItempHi,J);
if (M == 0)
temp1 = N;
elseif (N == 0)
temp1 = M;
else
 temp1 = sign(M+N)*sqrt(abs(M*N));
end
JtempLo = J - 1;
JtempHi = J + 1;
while ((Ecart(I,JtempLo) == 0) & (JtempLo > 1))
JtempLo = JtempLo - 1;
end
while ((Ecart(I,JtempHi) == 0) & (JtempHi < Rmax))
JtempHi = JtempHi + 1;
end
M = Ecart(I,JtempLo);
N = Ecart(I,JtempHi);
if (M == 0)
temp2 = N;
elseif (N == 0)
temp2 = M;
else
temp2 = sign(M+N)*sqrt(abs(M*N));
end
if (temp1 == 0)
EcartNew(I,J) = temp2;
elseif (temp2 == 0)
```

```
EcartNew(I,J) = temp1;
else
EcartNew(I,J) = sign(temp1+temp2)*sqrt(abs(temp1*temp2));
end
end
EcartNew(1,151) = 1.2; % point palsu ditambahkan kekuatan penskalaan
EcartNew(2,151) = -1.0; % final plot
end
I = 1:301;
J = 1:301;
x(I) = I;
y(J) = J;
xlabel('Y-axis cm'), ylabel('X-axis cm')
zlabel('Etheta V/m')
% title('FDTD Antena Biconical 6.5GHz: CW Driver')
title('FDTD Antena Biconical 6.5GHz: UWB Driver')
shading interp
colormap pink
```

Simulasi MATLAB : Idealcone.m

```
% Solusi bentuk tertutup untuk infinite antena biconical
function [Et] = IdealCone()
% Gunakan parameter yang sama (dr, dth, dt, ranges, etc.) dengan yang digunakan
% pada simulasi FDTD.
dr = 0.003;
dth = 1;
dt = 5e-12;
Z = 377;
c = 3e+8;
% Range diperluas sampai 15 median panjang gelombang di R dan pada 0-90
% derajat (θ). Sudut cone antena infinite sama dengan yang digunakan dalam
% simulasi FDTD.
Rmax = 151;
THmax = 151;
cone_angle = 31;
theta = cone_angle;
r = dr;
% Inisialisasi parameter gelombang, frekuensi (fo), amplituda (Eo),
% wave number (k) dan lainnya.
Vo = 0.033;
Eo = Vo/(2*log(cot((pi/180)*(cone_angle-1)/2)));
fo = 6.5e+9; % 6.5GHz
k = 2*pi*fo/c; % 136.1
t = 0;
for i = 1:Rmax
   theta = cone_angle;
   for j =1:(THmax-1)
      if (j >= cone_angle)
         wave = cos(2*pi*fo*t - k*r);
      else
```

```
            Et(i,j) = 0.0;
        end
        Hp(i,j) = Et(i,j)/Z;
        theta = theta + dth;
    end
    r = r + dr;
end
% Plot hasilnya
% Ubah ke koordinat rectangular cerminkan data simulasi antena untuk
% menampilkan radiasi medan.
for i = 1:Rmax
    for j = 1:THmax-1
        x = 32 + round((i* sin((j-1)*dth*pi/180))/5);
        y  = 30 + round((i*cos((j-1)*dth*pi/180))/5);
        Ecart(x,y) = Et(i,j);
        Ecart(x2,y) = Et(i,j);
    end
end
i = 1:62;
j = 1:56;
x(i) = i*1.5;
y(j) = j*1.5;
xlabel('Y-axis cm'), ylabel('X-axis cm')
zlabel('Etheta V/m')
title('Infinite 30 Degree Cone Antenna at 6.5GHz: Closed Form Solution')
shading interp
colormap pink
```