```
clc;
clear all;
close all;
global X Y Z nx ny nz H R
tm = clock;

%% 物理定数
eps0 = 8.85418782e-12;      % 真空の誘電率
mu0 = 4*pi*1e-7;            % 真空の透磁率
c0 = 1/sqrt(eps0*mu0);      % 真空中での高速[m/s]

%% 入力パラメータ
%=====================================================================
N_ant = 36;
ddx = 0.004;    % ボクセルの一辺の長さ[m]
R=44;                                           % 半球の半径(mm)
H=4;                                            % ボクセルの１辺の長さ(mm)
%% モデリング(領域)
X=-R:H:R;nx=length(X);
Z=X;nz=nx;
Y=2:H:42;ny=length(Y);
estCstFile = 'model_ver4_non_Si_sheet.cst'; % CSTファイル
%
frq = 1.8e+9;           % 使用周波数
idx_dat = 9;            % 実測の1.8GHzに対応する周波数サンプル点のID(1スタート)
epr_cup = 9.4;          % 誘電体カップの比誘電率
tanD_cup = 2.5e-6;      % 誘電カップのtanδ@1.8GHz
epr_fat = 10;           % 脂肪の比誘電率
tanD_fat = 0.32;        % 脂肪のtanδ@1.8GHz
N_iter  = 10;           % DBIMの反復回数
km = 1;                 % 0:BIM 1:DBIM
kp = 1;                 % 0:相反ペア平均化しない　1:する
k_fwd = 1;              % 0:測定データをとらない　1:とる
mu = 1;                 % 更新のフィードバック係数
%
prm_isol.alp = 0.1;         % 残差に基づく正則化制御係数
prm_isol.tol = 1e-1;        % CGの相対残差に関する収束条件
prm_isol.maxiter = 2;       % CGの反復回数の上限
%
%=====================================================================

%% ローカル変数
w = 2*pi*frq;
vol_vox=ddx^3;
%
sgm_cup = w*eps0*epr_cup*tanD_cup;      % 誘電カップの導電率
sgm_fat = w*eps0*epr_fat*tanD_fat;      % ターゲットの導電率
tanD_cup = sgm_cup/(w*eps0*epr_cup);    % 誘電カップのtanδ
tanD_fat = sgm_fat/(w*eps0*epr_fat);    % がんのtanδ
%
prm_isol.range_epr = [epr_cup*0.99 80.0];   % 比誘電率の下限・上限
prm_isol.range_sgm = [sgm_cup*0.99 2.0];    % 導電率の下限・上限
prm_isol.bound_er2sg = [0.02 0.04];         % 比誘電率に基づく導電率の下限・上限

%% 処理開始時間をラベルとする
dtLabel = sprintf('%4d%02d%02d-%02dh%02dm%02ds',tm(1),tm(2),tm(3),tm(4),tm(5),cast(tm(6),'int32'));
dirPath = sprintf('mwsdbim_%s',dtLabel);

%% 処理結果を保存するフォルダを作成
mkdir(dirPath);

%% 計測データ
cst = actxserver('CSTStudio.application');
if k_fwd==1
    % 真値の誘電特性分布モデルのセル位置と誘電特性値(pes_tru)を設定
    [pes_tru, N_vox] = mmSet_truthVoxelValues(epr_cup,sgm_cup);
    dpv_tru = [pes_tru(1:N_vox,5); pes_tru(1:N_vox,6)];     % 真の比誘電率と導電率の分布を並べたベクトル
    mws = cst.invoke('OpenFile',strcat(pwd,'¥',estCstFile));
    fprintf('    ● MW-Studioのモデルに誘電特性値を設定. ¥n');
    mws.invoke('DeleteResults');
```

```
    mmSet_voxelMaterials_mws(pes_tru,mws);
    fprintf('    ● MW-Studioによる順解析 : ');
    tic;
    solver = mws.invoke('Solver');
    solver.invoke('FrequencyRange',1.0,3.0);     % 周波数範囲(GHz)
    solver.invoke('FrequencySamples',21);        % サンプル点数
    solver.invoke('Start');
    release(solver);
    mws.invoke('Save');
    procTime_fwdSolve = toc;
    fprintf('%.1f[sec] ¥n',procTime_fwdSolve);
    fprintf('    ● 順解析の結果からＳ行列を取得 .¥n');
    Stm = mmGet_SparaMatrix_proto_mws(N_ant,idx_dat,kp,mws);
    Vtm = mmTransform_S2Vrec_707(Stm);
    save inv#000.mat Stm Vtm pes_tru pes N_vox dpv;
    Nrs=1;                                       %使う繰り返し番号+1
else
    load inv#000.mat Stm Vtm pes_tru pes N_vox dpv;
    Nrs=1;                                       %使う繰り返し番号+1
    mws = cst.invoke('OpenFile',strcat(pwd,'¥',estCstFile));
end

%% 画像再構成
% 初期の誘電特性分布モデルのセル位置と誘電特性値(pes)を設定
[pes, N_vox] = mmSet_initVoxelValues(epr_cup,sgm_cup);
dpv = [pes(1:N_vox,5); pes(1:N_vox,6)];          % 現在の比誘電率と導電率の分布を並べたベクトル
mmSet_voxelProbes_mws(pes,mws);                  % 電界プローブを設定
nRatio_dV = []; % 残差の推移
for iItr = Nrs:N_iter
    % 各反復開始
    fprintf('Iter#%03d ¥n',iItr);
    fprintf('    ● MW-Studioのモデルに誘電特性値を設定. ¥n');
    mws.invoke('DeleteResults');
    mmSet_voxelMaterials_mws(pes,mws);
    fprintf('    ● MW-Studioによる順解析 : ');
    tic;
    solver = mws.invoke('Solver');
    solver.invoke('FrequencyRange',1.0,3.0);     % 周波数範囲(GHz)
    solver.invoke('FrequencySamples',21);        % サンプル点数
    solver.invoke('Start');
    release(solver);
    mws.invoke('Save');
    procTime_fwdSolve = toc;
    fprintf('%.1f[sec] ¥n',procTime_fwdSolve);
    fprintf('    ● 順解析の結果から総合電界を取得 .¥n');
    [Ebx,Eby,Ebz] = mmGet_totE_proto_mws(N_ant,pes,idx_dat,mws);
    fprintf('    ● 順解析の結果からＳ行列を取得 .¥n');
    Sbc = mmGet_SparaMatrix_proto_mws(N_ant,idx_dat,kp,mws);
    if iItr==1
        Sbm=Sbc;
    end
    fprintf('    ● Ｓパラ→電圧変換.¥n');
    Vbc = mmTransform_S2Vrec_707(Sbc);
    if iItr == 1
        fprintf('    ● 癌なし実測・解析データを元に癌あり実測データを補正 .¥n');
        R_calib = Sbc ./ Sbm;
        Stm = Stm .* R_calib;
        fprintf('    ● Ｓパラ→電圧変換.¥n');
        Vtm = mmTransform_S2Vrec_707(Stm);
    end

    fprintf('    ● 受信電界の残差を算出.¥n');
    dV = mmCalc_scatteredV_proto(Vtm,Vbc);
    if iItr == 1
        dV0 = dV;   % 初期モデルに対する残差
    end
    nRatio_dV = [nRatio_dV; norm(dV)/norm(dV0)];
    fprintf('    ● 現在の処理状況の図示・ファイル保存.¥n');
    mmFig_residualProcess_proto(N_iter,nRatio_dV,dirPath);  % 受信電界残差の推移

```

```
    if iItr == 1 || km == 1
        fprintf('    ● Green関数を算出.¥n');
        Gb = mmCalc_greenFunc_proto_mws2v(Ebx,Eby,Ebz);
    end

    fprintf('    ● Jacobianを算出.¥n');
    J = mmCalc_jacobian_proto_mws2v(w,vol_vox,Gb,Ebx,Eby,Ebz);
    fprintf('    ● 積分方程式を解いて誘電コントラスト(モデル更新量)を算出.¥n');
    dS = mmSolve_linSys(J,dV,Vbc,prm_isol);
    dS=mu*dS;
    fprintf('    ● モデルを更新.¥n');
    dS_cmp = 1i*w*eps0*dS(1:N_vox) + dS(N_vox+1:N_vox*2);
    dpv(1:N_vox) = dpv(1:N_vox) + imag(dS_cmp)/(w*eps0);
    dpv(N_vox+1:N_vox*2) = dpv(N_vox+1:N_vox*2) + real(dS_cmp);
    % Box制約による解の修正
    for iV = 1:N_vox
        % 誘電率
        if dpv(iV) < prm_isol.range_epr(1)
            dpv(iV) = prm_isol.range_epr(1);
        end
        if dpv(iV) > prm_isol.range_epr(2)
            dpv(iV) = prm_isol.range_epr(2);
        end
        % 導電率1
        if dpv(N_vox+iV) < prm_isol.range_sgm(1)
            dpv(N_vox+iV) = prm_isol.range_sgm(1);
        end
        if dpv(N_vox+iV) > prm_isol.range_sgm(2)
            dpv(N_vox+iV) = prm_isol.range_sgm(2);
        end
        % 導電率2
        if dpv(N_vox+iV) < dpv(iV)*prm_isol.bound_er2sg(1)
            dpv(N_vox+iV) = dpv(iV)*0.03;
        end
        if dpv(N_vox+iV) > dpv(iV)*prm_isol.bound_er2sg(2)
            dpv(N_vox+iV) = dpv(iV)*0.03;
        end
    end
    pes(1:N_vox,5) = dpv(1:N_vox);          % 比誘電率
    pes(1:N_vox,6) = dpv(N_vox+1:N_vox*2);  % 導電率
    pes(1:N_vox,7) = pes(1:N_vox,6)./pes(1:N_vox,5)/(w*eps0);   % tanD
    mmFig_diProf1D_proto_mws(iItr,pes,pes_tru,dirPath);         % 誘電特性分布の１Ｄ表示
    mmFig_quality_factor(N_iter,iItr,pes,pes_tru);
    mmFig_3Dimage(iItr,pes,pes_tru,dirPath)

    fprintf('    ● すべての変数をMATファイルに保存 ¥n');
    save(sprintf('inv#%03d.mat',iItr),'Stm','Sbc','Vtm','pes','pes_tru','N_vox','dpv');

end
release(mws);

%% CSTStudioとの接続を解除
release(cst);

```