```
function [delTauG] = getIonoDelay(ionodata,fc,rRx,rSv,tGPS,model)
% getIonoDelay : Return a model-based estimate of the ionospheric delay
% experienced by a transionospheric GNSS signal as it
% propagates from a GNSS SV to the antenna of a terrestrial
% GNSS receiver.
%
% INPUTS
%
% ionodata ------- Structure containing a parameterization of the
% ionosphere that is valid at time tGPS. The structure is
% defined differently depending on what ionospheric model
% is selected:
%
% broadcast --- For the broadcast (Klobuchar) model, ionodata
% is a structure containing the following fields:
%
% alpha0 ... alpha3 -- power series expansion coefficients
% for amplitude of ionospheric delay
%
% beta0 ... beta3 -- power series expansion coefficients
% for period of ionospheric plasma density cycle
%
%
% Other models TBD ...
%
% fc ------------- Carrier frequency of the GNSS signal, in Hz.
%
% rRx ------------ A 3-by-1 vector representing the receiver antenna position
% at the time of receipt of the signal, expressed in meters
% in the ECEF reference frame.
%
% rSv ------------ A 3-by-1 vector representing the space vehicle antenna
% position at the time of transmission of the signal,
% expressed in meters in the ECEF reference frame.
%
% tGPS ----------- A structure containing the true GPS time of receipt of
% the signal. The structure has the following fields:
%
% week -- unambiguous GPS week number
%
% seconds -- seconds (including fractional seconds) of the
% GPS week
%
% model ---------- A string identifying the model to be used in the
% computation of the ionospheric delay:
%
% broadcast --- The broadcast (Klobuchar) model.
%
% Other models TBD ...
%
% OUTPUTS
%
% delTauG -------- Modeled scalar excess group ionospheric delay experienced
% by the transionospheric GNSS signal, in seconds.
%
%+------------------------------------------------------------------------+
% References: For the broadcast (Klobuchar) model, see IS-GPS-200F
% pp. 128-130.
%
%+========================================================================+
figure(),
```

```
[X,Y,Z]=sphere;
Re= 6378137 ; %Earth Radius in meters
surf(X*Re,Y*Re,Z*Re);
xlabel('m');
ylabel('m');
zlabel('m');
axis equal,
hold on,

plot3(rRx(1),rRx(2),rRx(3),'o');
hold on,
plot3(rSv(1),rSv(2),rSv(3),'*')


% A is left as rad instead of semi-circle unit since they are only used for trig calculations
[E, A, r_lla, s_lla]=findElevationAzimuthAngleANDLLA(rRx,rSv);
E = E/pi; % semi-circle
phi_u    = r_lla(1)/pi; % semi-circles
lambda_u = r_lla(2)/pi; % semi-circles

alpha    = [ionodata.broadcast.alpha0,ionodata.broadcast.alpha1,ionodata.broadcast.alpha2,ionodata.broadcast.alpha3];
beta     = [ionodata.broadcast.beta0,ionodata.broadcast.beta1,ionodata.broadcast.beta2,ionodata.broadcast.beta3];

psi   = 0.0137/(E+0.11)-0.22; % semi-circle
phi_i = phi_u+psi*cos(A); % semi-circle

if phi_i > 0.416
    phi_i = 0.416;
elseif phi_i < -0.416
    phi_i = -0.416;
end

lambda_i = lambda_u + psi*sin(A)/cos(phi_i*pi); % semi-circle
t        = 4.32*10^4*lambda_i +tGPS.seconds;

if t >= 86400
    t = t-86400;
elseif t<= -86000
    t = t+86400;
end

phi_m    = phi_i + 0.064*cos((lambda_i-1.617)*pi); % Multiply by pi to remove semicircle unit
F        = 1 + 16*(0.53 -E)^3;
PER      = sum(beta.*phi_m);

if PER < 72000
    PER = 72000;
end

x      = 2*pi*(t-50400)/PER;
AMP    = sum(alpha.*phi_m);

if AMP < 0
    AMP = 0;
end

if abs(x) < 1.57
    delTauG = F*(5*10^-9+AMP*(1-x^2/2+x^4/24)); % sec
elseif abs(x) >= 1.57
    delTauG = F*5*10^-9; % sec
end
```