

```

function [elevation_ts, distance_ts, T0] =
generate_passage_singleorbit(h0, iK, phi0, alpha0, Nsat, lambdaT, etaT,
dt, Tmax)

% INPUT
% h0: height of the orbit [km]
% iK: orbit inclination [degrees]
% phi0: longitude at t=0 of the ascending node of the orbit [degrees]
% alpha0: angular position of the 1st satellite at t = 0 [degrees]
% Nsat: number of satellites in the orbit
% lambdaT: Ground user latitude [degrees]
% etaT: Ground user longitude [degrees]
% dt: Sampling time [seconds]
% Tmax: duration of the time window [seconds]

% OUTPUT
% elevation_ts: matrix that contains in the rows the elevation time
series
% for all satellites in the orbit [degrees]
% distance_ts : matrix that contains in the rows the distance time series
% for all satellites in the orbit [km]
% T0: orbital period [s]

mu = 3.986e5; % Gravitational constant [km^3 s^-2]
Re = 6371; % Earth radius [km]

R0 = Re + h0; % Orbit radius [km]

v0 = sqrt(mu / R0); % Satellite Radial speed [km/s]
w0 = v0 / R0; % Satellite Angular speed [rad / s]
T0 = 2*pi / w0; % Satellite Orbital period [s]
wEd = 1 / 24/10; % Earth angular speed [degree/s]

alpha_sep = 2*pi / Nsat;
alpha0_rad = pi / 180 * alpha0;
alpha_0_vec = (0:(Nsat-1)) * alpha_sep + alpha0_rad;
alpha_0_vec = alpha_0_vec(:);

t_vec = 0:dt:Tmax; %Time axis [s]

alpha = w0 * t_vec + alpha_0_vec; % Satellite position in the orbit [rad]
app_vec = sind(iK) * sin(alpha);
theta = acosd(app_vec); % Polar coordinate theta [degrees]
phi_rad = atan2(cosd(iK)*sin(alpha), cos(alpha)); % Polar coordinate phi
[radiants, wrapped]
% Unwrapping of the polar coordinate phi
phi = 180 / pi * unwrap(phi_rad, [], 2) ; % convert from radiants to
degrees smoothly with no wraps(errors)

lambdaS = 90 - theta; % Satellite latitude [degrees]
etaS = phi + phi0 - wEd * t_vec; % Satellite longitude [degrees]
etaS = mod(etaS, 360);
etaS(etaS>180) = etaS(etaS>180) - 360 ;

% gamma is the central angle between GU and Sub-Satellite Point
gamma = acos(sind(lambdaT) * sind(lambdaS) + cosd(lambdaT) *
cosd(lambdaS) .* cosd(etaS-etaT));

```

```
elevation_ts = atan2d(cos(gamma) - Re / R0, sin(gamma)); % Elevation time  
series [degrees]
```

```
distance_ts = sqrt(Re^2 * (sind(elevation_ts)).^2 + h0^2 + ...  
2*h0*Re) - Re * (sind(elevation_ts)); % Distance time series [km]
```