Contents


```
function [ Lmatrix, Amatrix,rho_f,Xs_f,Ys_f,Zs_f,P1_f,dtsL1_dtr_f,tAtoS_f]...
    = satLandP( satelliteNumberOrder,P1_f,navfiles,XA0,YA0,ZA0,ndaysf,nhoursf,nminutesf,nsecondsf)
```

Constants

```
c = 299792458; % speed of light (m/s)
mu = 3.986005e14; % universal gravitational parameter (m/s)^3
omega_e_dot = 7.2921151467e-5; % earth rotation rate (rad/s)
F = -4.442807633e-10; % s/m^1/2
```

Importing numbers from table 3 for a satellite

```
i = 1:8:112;
sat = navfiles(i(satelliteNumberOrder):i(satelliteNumberOrder)+8,:);
sat = transpose(cell2mat(sat));
sat = num2cell(sat);
% Imports numbers to all variables
[~,     af0,        af1,        af2,...
    ~,       crs,        change_n,         m0,...
    cuc,        ec,         cus,        sqrtA,...
    toe,        cic,        omega0,     cis,...
    i0,         crc,        w,          omegadot,...
    idot,       ~,          ~,          ~,...
    ~,          ~,          tgd,        ~,...
    ~]...
    =sat{:};
```

```
Error using satLandP (line 10)
Not enough input arguments.
```

1. Compute signal propagation time by (13)

```
tA_nom = seconds_in_week(ndaysf,nhoursf,nminutesf,nsecondsf); % 2 days, 1 hour, 14 minutes My Time
tAtoS_f = P1_f/c; % signal propagation time
```

2. Compute signal transmission time by (14)

```
tS_nom = tA_nom - P1_f/c;
```

3. Compute satellite clock correction dtsL1

by (24) and (25), neglect dtr

```
t_oc = toe; % I believe this is true, but not sure
change_tsv_f = af0 + af1*(tS_nom-t_oc)+af2*(tS_nom-t_oc)^2; % (25)
dtsL1 = change_tsv_f - tgd; % (24)
```

4. Compute ts using the correction from the step 3.

```
ts_f = tS_nom - dtsL1;
```

5. Compute eccentric anomaly (Table 2)

ek = mk + ec*sin(ek)

```
A = sqrtA^2;
n0 = sqrt(mu/A^3); % Computed mean motion
n = n0 + change_n;
tk = ts_f - toe;
tk = fixTk(tk); % if,then for table 2 of tk
mk = m0 + n*tk;
Ek = keplersEquation(mk,ec);
```

6. Compute dtr by (26) and ts by (15).

```
change_tr = F*ec*sqrtA*sin(Ek); %(26)
ts_with_dtr = ts_f - change_tr;
```

7. Compute satellite coordinates Xs, Ys, Zs, for time ts

- Table 2 Calculate rk

```
vk = atan2((sqrt(1-ec^2)*sin(Ek)/(1-ec*cos(Ek))),...
    ((cos(Ek)-ec)/(1-ec*cos(Ek))));
Phik = vk + w;
drk = crs*sin(2*Phik) + crc*cos(2*Phik);
rk = A*(1-ec*cos(Ek)) + drk;  % Corrected radius
% Calculate uk
duk = cus*sin(2*Phik) + cuc*cos(2*Phik);
uk = Phik + duk;
% Calculate ik
dik = cis*sin(2*Phik) + cic*cos(2*Phik);
ik = i0 + dik + idot*tk;
% Calculate omega's
omegak = omega0 + (omegadot-omega_e_dot)*tk - omega_e_dot*toe;
% Calculate xkp and ykp
```

```
xkp = rk*cos(uk);
ykp = rk*sin(uk);
% Calculate xk,yk,zk -> Xs, Ys, Zs for time ts
xk = xkp*cos(omegak) - ykp*cos(ik)*sin(omegak);
yk = xkp*sin(omegak) + ykp*cos(ik)*cos(omegak);
zk = ykp*sin(ik);
Xs_f = xk;
Ys_f = yk;
Zs_f = zk;
```

8. Compute satellite clock correction dtsL1 by (24) - (27)

```
dtsL1_dtr_f = change_tsv_f + change_tr - tgd; % (24)
```

9. Compute tropospheric correction T_A_to_s (tA)

10. Compute ionospheric correction I_A_to_s (tA)

11. Compute approximate distance rho_A0_to_s (tA) by (11).

```
rho_f = sqrt(...
    (Xs_f - XA0 + omega_e_dot * YA0 * tAtoS_f)^2 + ... % x^2
    (Ys_f - YA0 - omega_e_dot * XA0 * tAtoS_f)^2 + ... % y^2
    (Zs_f - ZA0)^2   ... % z^2
    );
% dtA = 0;
% rho_A_to_s = P1 + c*dtsL1_with_dtr - c*dtA; % (8) dtA =\= 0
```

12. Repeat steps 1 - 11 for all measured satellites.

13. Compute elements of vector L (19).

```
Lmatrix = P1_f - rho_f + c*dtsL1_dtr_f;
```

14. Compute elements of matrix A (20); a_x_to_s , a_y_to_s , a_z_to_s by (12)

```
Amatrix = 1/rho_f*[-(Xs_f - XA0),-(Ys_f - YA0),-(Zs_f - ZA0),rho_f];

end
```