Contents

Computation of receiver's position

```
clear all;
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
time = [2,1,14,0]; % days, hours, minutes, seconds
junk = num2cell(time);
[nday,nhours,nminutes,nseconds] = junk{:};
clear junk;
```

Import observer file

```
lov033b = importObserverFileAsString('0lov033b.04o', 1, 5629);
% Import P1 numbers and satellite numbers
[rowInObs,nOfRows] = findTimeInObsFunction( lov033b,time ); % match your time with observer
p1_numbers = importObsP1numbers('0lov033b.04o', rowInObs+1,rowInObs+nOfRows*2); % Import P1
satelliteNumbers = importObsSatelliteNumbers('0lov033b.04o', rowInObs,rowInObs); % Import sa
[XA0,YA0,ZA0] = sampleFunction(lov033b); % Record Approximate Position
```

Import navigation file

```
navfiles = importNavigationFiles('0lov033b.04n');
```

Match satellite numbers with available satellites

```
satNumMatch = navfiles(1:8:96,1); % Order of satellite numbers import
sortedSatelliteNumbers = sortrows([satelliteNumbers',p1_numbers],1);
```

Main loop steps 1-14

Calculates variables needed for correction iterations

```
count = 1;
for i = 1:length(satNumMatch)
```

Steps 1-14 done inside

```
    if cell2mat(satNumMatch(i))==sortedSatelliteNumbers(count,1)
        [ Lmat(count,:), ...
            Amat(count,:),...
            rho(count,:),...
            Xs(count,:),Ys(count,:),Zs(count,:),...
            P1(count,:),...
            dtsL1_with_dtr(count,:),...
            tAtoS(count,:)]...
            = satLandP( i,sortedSatelliteNumbers(count,2),navfiles,XA0,YA0,ZA0,nday,nhours,n
        count = count + 1;
    else
        fprintf('No data for Satellite%3d\n',cell2mat(satNumMatch(i)))
    end
```

```
No data for Satellite 24
```

```
end
```

17. Repeat steps 11 -16 until the solution has converged.

The solution has converged if the condition is fulfilled

```
for i = 1:10

    changeX = (Amat'*Amat)\(Amat'*Lmat); % eq. (21)
    v(:,i) = -Amat*changeX + Lmat; % eq. (17)
    newXYZ = [XA0,YA0,ZA0] + changeX(1:3)'; % eq. (22) estimated coordinates
    newxyzcell = num2cell(newXYZ);
    [XA0,YA0,ZA0] = newxyzcell{:};
    clear newxyzcell;
    rho = sqrt(... % recompute rho
        (Xs - XA0 + omega_e_dot * YA0 * tAtoS).^2 + ... % x^2
        (Ys - YA0 - omega_e_dot * XA0 * tAtoS).^2 + ... % y^2
        (Zs - ZA0).^2   ... % z^2
        );
    Amat = [-(Xs - XA0)./rho,... % recompute matrix A
        -(Ys-YA0)./rho,...
        -(Zs-ZA0)./rho,...
        rho./rho];
    Lmat = P1 - rho + c*dtsL1_with_dtr; % recompute matrix L
```

17. Convergence condition

```
    if i>1 % check for convergence condition
        condition = abs(v(:,end)'*v(:,end)-v(:,end-1)'*v(:,end-1)); % condition must be 1e-5
        if condition < 1e-4

            fprintf('Convergence condition met = %d\n',condition);
```

```
Convergence condition met = 3.336069e-08
```

Lab 2

Find sigma and Q

```
            Q = inv((Amat'*Amat));
            sigma_0 = sqrt(v(:,end)'*v(:,end)/(length(Amat)-length(Q)));
            sigma_x = sigma_0*sqrt(Q(1,1));
            sigma_y = sigma_0*sqrt(Q(2,2));
            sigma_z = sigma_0*sqrt(Q(3,3));
            sigma_t = sigma_0*sqrt(Q(4,4))/c;
            fprintf('X = %7.3f, mX = %7.3f\n',XA0,sigma_x);
            fprintf('Y = %7.3f, mX = %7.3f\n',YA0,sigma_y);
            fprintf('Z = %7.3f, mX = %7.3f\n',ZA0,sigma_z);
            fprintf('T = %0.10f, mt = %0.10d\n',-changeX(4)'/c,sigma_t);
            return

        end
    end

end
```