Computation of Receiver Position from Code Pseudoranges

Kevin Mead

April 13, 2014

1 Introduction

The task is to compute the coordinates of a GPS receiver at given time using P1-code pseudo ranges. My specific observation epoch is: 2004-02-02, 1 h 14 min 00 sec. The calculations were done with observation files and navigation files in RINEX format. The computation of this position is difficult because the satellite and receiver clocks are not synchronised, the satellite and receiver are both moving with varying velocities, and propagation times. There are many more problems that need to be accounted for like ionospheric and tropospheric corrections, but we have neglected them in this lab because we want a simple understanding before we take on the harder tasks.

2 Main Code

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
% match your time with observer time
[rowInObs,nOfRows] = findTimeInObsFunction( lov033b,time );
% Import P1 numbers from the matched time above
p1_numbers = importObsP1numbers('0lov033b.04o', rowInObs+1,rowInObs+nOfRows*2);
% Import satellite numbers from matched time
satelliteNumbers = importObsSatelliteNumbers('0lov033b.04o', rowInObs,rowInObs);
[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,nminutes,nseconds);
        count = count + 1;
    else
        fprintf('No data for Satellite%3d\n',cell2mat(satNumMatch(i)))
    end


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) est. 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 or lower
        if condition < 1e-4

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

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
```

3 Matlab Function: satLandP.m

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,nsec
ondsf)
```

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{:};
```

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
```

References