```matlab
%-------------------------------------------------------------------------
%
%          ADJUSTMENT THEORY I
%   Exercise 13: Adjustment Calculation - part VIII
%
%   Author         : Anastasia Pasioti
%   Version        : October 15, 2018
%   Last changes   : February 08, 2023
%
%-------------------------------------------------------------------------

clc;
clearvars;
close all;
format longG;

%-------------------------------------------------------------------------
%   Task 2
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Observations and initial values for unknowns
%-------------------------------------------------------------------------

% Load all files
dist = load('Distances_task2.txt');
dir = load('Directions_task2.txt');
fixedpoint = load('FixedPoints_task2.txt');      % y x 6 9
np = load('NewPoints_task2.txt');                % y x 1 15
coordinates = [fixedpoint; np];
coordinates2 = [fixedpoint; np];
dist_nums = dist(:,1:2);
dir_nums = dir(:,1:2);
u_nums = np(:,1);
w_nums = coordinates(:,1);


% Vector of observations
L = [dist(:,3); dir(:,3)*pi/200];

% Gauss-Krueger coordinates for control points [m]
fixed = [fixedpoint(:, 2) fixedpoint(:, 3)];

% Initial values for orientation unknowns
initial_w = zeros(length(coordinates),1); % w6 w9 w1 w15

% Initial values for unknowns
X_0 = [np(:,3); np(:,2); initial_w];

% X_0  = x100 x101 x102 x103 y100 y101 y102 y103 w1000 w2000 w3000 w100 w101 w102
w103
```

```
% Number of observations
no_n = length(L);

% Number of unknowns
no_u = length(X_0);

% Redundancy
r = no_n-no_u;

%---------------------------------------------------------------------------
%  Stochastic model
%---------------------------------------------------------------------------
% VC Matrix of the observations
s_dir = ones(length(dist),1)*0.001*pi/200;      %[rad]
s_dist = ones(length(dir),1)*0.001;             %[m]

S_dist = diag(s_dist.^2);
S_dir = diag(s_dir.^2);

S_LL = blkdiag(S_dist, S_dir);

% Theoretical standard deviation
sigma_0 = 1e-4;

% Cofactor matrix of the observations
Q_LL = 1/sigma_0^2*S_LL;

% Weight matrix
P = inv(Q_LL);

%---------------------------------------------------------------------------
%  Adjustment
%---------------------------------------------------------------------------

% break-off condition
epsilon = 1e-5;
delta = 1e-9;
max_x_hat = Inf;
Check2 = Inf;

% Number of iterations
iteration = 0;

% Initialising A
A = zeros(no_n, no_u);

% Iteration
while max_x_hat>epsilon || Check2>delta
```

```
    L_0 = create_L_0(dist_nums, dir_nums, coordinates, initial_w);

    % Vector of reduced observations
    l = L-L_0;

    % Design matrix with the elements from the Jacobian matrix J
    A = create_design_matrix(u_nums, w_nums, coordinates, dist_nums, dir_nums,
no_n,no_u, initial_w);

    % Normal matrix
    N = A'*P*A;

    % Vector of right hand side of normal equations
    n = A'*P*l;

    % Inversion of normal matrix / Cofactor matrix of the unknowns
    Q_xx = inv(N);

    % Solution of normal equations
    x_hat = Q_xx*n;

    % Adjusted unknowns
    X_hat = X_0+x_hat;


    % Update
    X_0 = X_hat;

    coordinates(length(fixedpoint)+1:length(coordinates),3) =
X_hat(1:length(u_nums));
    coordinates(length(fixedpoint)+1:length(coordinates),2) = X_hat(length(u_nums)
+1:2*length(u_nums));
    initial_w = X_hat(2*length(u_nums)+1:length(X_hat));

    % Check 1
    max_x_hat = max(abs(x_hat));

    % Vector of residuals
    v = A*x_hat-l;

    % Vector of adjusted observations
    L_hat = L+v;

    % Function
    vTPv = v'*P*v;

    % Functional relationships

    phi_X_hat = create_L_0(dist_nums, dir_nums, coordinates, initial_w);
```

```
    % Check 2
    Check2_full = L_hat - phi_X_hat;
    Check2 = max(abs(L_hat - phi_X_hat));

    % Update number of iterations
    iteration = iteration+1;

end

if Check2<=delta
    disp('Everything is fine!')
else
    disp('Something is wrong.')
end
```

```
Everything is fine!
```

```

% Convert to [gon]

X_hat(2*length(u_nums)+1:length(X_hat)) = X_hat(2*length(u_nums)
+1:length(X_hat))*200/pi;
for i = (2*length(u_nums)+1):length(X_hat)
    if X_hat(i) < 0
        X_hat(i) = X_hat(i) + 400;
    end
end

% Empirical reference standard deviation
s_0 = sqrt(vTPv/r);

% VC matrix of adjusted unknowns
S_XX_hat = s_0^2*Q_xx;

% Standard deviation of the adjusted unknowns
s_X = sqrt(diag(S_XX_hat));

% Cofactor matrix of adjusted observations
Q_LL_hat = A*Q_xx*A';

% VC matrix of adjusted observations
S_LL_hat = s_0^2*Q_LL_hat;

% Standard deviation of the adjusted observations
s_L_hat = sqrt(diag(S_LL_hat));

% Cofactor matrix of the residuals
Q_vv = Q_LL-Q_LL_hat;

% VC matrix of residuals
S_vv = s_0^2*Q_vv;
```

```
% Standard deviation of the residuals
s_v = sqrt(diag(S_vv));

% Results

table(X_hat, s_X, 'RowNames', {'x100' 'x101' 'x102' 'x103' 'y100' 'y101' 'y102'
'y103' 'w1000' 'w2000' 'w3000' 'w100' 'w101' 'w102' 'w103'})
```

ans = 15×2 table

	X_hat	s_X
1 x100	5820727.42269052	0.0154941327195254
2 x101	5820857.99051548	0.0386879713817922
3 x102	5820848.83488121	0.0187543925907365
4 x103	5820700.4290112	0.0130474231024254
5 y100	4590159.87178345	0.0101388084910928
6 y101	4589800.10118633	0.0210674961262621
7 y102	4590163.257201	0.0112741529387646
8 y103	4589956.94706474	0.0134693485462597
9 w1000	398.682310959361	0.000240470604518462
10 w2000	398.368428251891	0.000276925029799226
11 w3000	398.355489134244	0.000247853102793691
12 w100	398.470074571213	0.000164476973379866
13 w101	398.266611529727	0.000309743644808751
14 w102	398.767360629217	0.000189348113578543

⋮

```
table(L, v, L_hat, s_v, s_L_hat)
```

ans = 36×5 table

	L	v	L_hat	s_v	s_L_hat
1	201.941	-0.0230148566240063	201.917985143376	0.0179114739857134	0.0094796569798248

	L	v	L_hat	s_v	s_L_hat
2	175.94	0.00576707423236967	175.945767074232	0.0166749014077383	0.0115166166836548
3	106.177	-0.0106279759627461	106.166372024037	0.0155610899451895	0.0129821907440059
4	175.288	0.00352772483510707	175.291527724835	0.0154353595675867	0.0131314306842529
5	93.728	-0.0245210209853159	93.7034789790147	0.0178919866751334	0.00951638637366501
6	122.506	-0.00833838658431939	122.497661613416	0.0150591670392423	0.0135612051411454
7	201.941	-0.0230148566240063	201.917985143376	0.0179114739857134	0.0094796569798248
8	121.468	-0.00861954718997071	121.45938045281	0.0152770215766625	0.0133153073018319
9	207.826	-0.023623150839815	207.80237684916	0.0158540271013805	0.0126227818433467
10	93.727	-0.0235210209853253	93.7034789790147	0.0178919866751334	0.00951638637366501
11	222.323	-0.00283176306142713	222.320168236939	0.0155026241537843	0.013051951621981
12	175.287	0.00452772483511185	175.291527724835	0.0154353595675867	0.0131314306842529
13	175.942	0.00376707423236012	175.945767074232	0.0166749014077383	0.0115166166836548
14	121.466	-0.00661954718996116	121.45938045281	0.0152770215766625	0.0133153073018319

⋮

```
function A = create_design_matrix(u_nums, w_nums, coordinates, dist_nums, dir_nums, 
no_n, no_u, initial_w)

A = zeros(no_n,no_u);
```

```
C_dist = x_y_dist(dist_nums, coordinates);

C_dir = x_y_w_dir(dir_nums, coordinates, initial_w);

for i = 1:length(dist_nums) % distance Jacobian matrix
    for j = 1:length(u_nums)
        if dist_nums(i, 1) == u_nums(j)
            A_dist(i,j) = ds_dx_from(C_dist(i,3), C_dist(i,1), C_dist(i,4),
C_dist(i,2));
            A_dist(i,j+length(u_nums)) = ds_dy_from(C_dist(i,3), C_dist(i,1),
C_dist(i,4), C_dist(i,2));
        elseif dist_nums(i, 2) == u_nums(j)
            A_dist(i,j) = ds_dx_to(C_dist(i,3), C_dist(i,1), C_dist(i,4),
C_dist(i,2));
            A_dist(i,j+length(u_nums)) = ds_dy_to(C_dist(i,3), C_dist(i,1),
C_dist(i,4), C_dist(i,2));
        end
    end
end

for i = 1:length(dir_nums) % directions Jacobian matrix
    for j = 1:length(u_nums)
        if dir_nums(i, 1) == u_nums(j)
            A_dir(i,j) = dr_dx_from(C_dir(i,3), C_dir(i,1), C_dir(i,4), C_dir(i,2));
            A_dir(i,j+length(u_nums)) = dr_dy_from(C_dir(i,3), C_dir(i,1),
C_dir(i,4), C_dir(i,2));
        elseif dir_nums(i, 2) == u_nums(j)
            A_dir(i,j) = dr_dx_to(C_dir(i,3), C_dir(i,1), C_dir(i,4), C_dir(i,2));
            A_dir(i,j+length(u_nums)) = dr_dy_to(C_dir(i,3), C_dir(i,1),
C_dir(i,4), C_dir(i,2));
        end
    end
end

for i = 1:length(dir_nums) % orientation parameter w Jacobian matrix
    for j = 1:length(w_nums)
        if dir_nums(i, 1) == w_nums(j)
            A_w(i,j) = -1;
        end
    end
end


A(1:size(A_dist,1), 1:size(A_dist,2)) = A(1:size(A_dist,1), 1:size(A_dist,2)) +
A_dist;

A(size(A_dist,1)+1:size(A_dist,1)+size(A_dir,1), 1:size(A_dir,2)) =
A(size(A_dist,1)+1:size(A_dist,1)+size(A_dir,1),  1:size(A_dir,2):size(A_dir,2)) +
A_dir;
```

```
A(size(A_dist,1)+1:size(A_dist,1)+size(A_w,1),
size(A_dir,2)+1:size(A_dir,2)+size(A_w,2)) =
A(size(A_dist,1)+1:size(A_dist,1)+size(A_w,1),
size(A_dir,2)+1:size(A_dir,2)+size(A_w,2)) + A_w;
end

function C = x_y_dist(dist_nums, coordinates)

for i = 1:size(dist_nums, 1)
    for j = 1:size(dist_nums, 2)
        for k = 1:size(coordinates, 1)
            if dist_nums(i,j) == coordinates(k, 1)
                Y(i,j) = coordinates(k,2);
                X(i,j) = coordinates(k,3);
            end
        end
    end
end
C = [X Y];
end

function C = x_y_w_dir(G, coordinates, w)
for i = 1:size(G, 1)
    for j = 1:size(G, 2)
        for k = 1:size(coordinates, 1)
            if G(i,j) == coordinates(k,1)
                Y(i,j) = coordinates(k,2);
                X(i,j) = coordinates(k,3);
            end
            if G(i,1) == coordinates(k,1)
                W(i,1) = w(k);
            end
        end
    end
end
C = [X Y W];
end

function L_0 = create_L_0(dist_nums, dir_nums, coordinates, initial_w)

% Distances
L_0_dist = zeros(length(dist_nums),1);
C_dist = x_y_dist(dist_nums, coordinates);
for i = 1:length(dist_nums)
    L_0_dist(i) = distance(C_dist(i,3),C_dist(i,1),C_dist(i,4),C_dist(i,2));
end

% Directions
L_0_dir = zeros(length(dir_nums),1);
```

```
C_dir = x_y_w_dir(dir_nums, coordinates, initial_w);

for i = 1:length(dir_nums)
    L_0_dir(i) = direction(C_dir(i,3), C_dir(i,1), C_dir(i,4), C_dir(i,2),
C_dir(i,5));
end

L_0 = [L_0_dist; L_0_dir];

end
```