```matlab
%---------------------------------------------------------------------------
%
%          ADJUSTMENT THEORY I
%    Exercise 12: Adjustment Calculation - part VII
%
%   Author         : Anastasia Pasioti
%   Version        : October 12, 2018
%   Last changes   : January 31, 2023
%
%---------------------------------------------------------------------------
% Group 6
%
% Fahd Sbahi / 0501211
% Nils Lang / 0494447
% Felix Nöding / 0501313
% Egor Grekhov / 0501111
% Corban Rosenauer / 0494441
% Zilong Liu / 0501297
%---------------------------------------------------------------------------

clc;
clearvars;
close all;
format long g;
%---------------------------------------------------------------------------
%   Task 1
%---------------------------------------------------------------------------
%---------------------------------------------------------------------------
%   Observations and initial values for the unknowns
%---------------------------------------------------------------------------
%Load files
dir = load('directions.txt');               %[gon]
coord = load('Points.txt');              %[m]     %Error-free

for i = 1:size(coord,1)
    eval(['y' num2str(coord(i,1)) '=' num2str(coord(i,2)) ';']);
    eval(['x' num2str(coord(i,1)) '=' num2str(coord(i,3)) ';']);
end

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

%Number of observations
no_n = length(L);

%Initial values for the unknowns
x3 = 250;
y3 = 500;
w3 = 1;
```

```
%Vector of initial values for the unknowns
X_0 = [x3 y3 w3]';

%Number of unknowns
no_u = length(X_0);

%Redundancy
r = no_n-no_u;

%---------------------------------------------------------------------------
%  Stochastic model
%---------------------------------------------------------------------------
s_dir = 0.001 * pi / 200;

%VC Matrix of the observations
S_LL = s_dir^2*eye(no_n);

%Theoretical standard deviation
sigma_0 = 0.001;     %a priori

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

%Weight matrix
P = inv(Q_LL);

%---------------------------------------------------------------------------
%  Adjustment
%---------------------------------------------------------------------------
%break-off conditions
epsilon = 10^-5;
delta = 10^-13;
max_x_hat = Inf;
Check2 = Inf;

%Number of iterations
iteration = 0;

while max_x_hat>epsilon || Check2>delta

     %Observations as functions of the approximations for the unknowns

     L_0(1,1) = direction(y3, x3, y1, x1, w3);
     L_0(2,1) = direction(y3, x3, y2, x2, w3);
     L_0(3,1) = direction(y3, x3, y4, x4, w3);
     L_0(4,1) = direction(y3, x3, y5, x5, w3);
     L_0(5,1) = direction(y3, x3, y6, x6, w3);

     %Vector of reduced observations
     l = L - L_0;
```

```
%Design matrix with the elements from the Jacobian matrix J
A(1,1) = dr_dx_from(y3,x3,y1,x1);
A(1,2) = dr_dy_from(y3,x3,y1,x1);
A(1,3) = -1;

A(2,1) = dr_dx_from(y3,x3,y2,x2);
A(2,2) = dr_dy_from(y3,x3,y2,x2);
A(2,3) = -1;

A(3,1) = dr_dx_from(y3,x3,y4,x4);
A(3,2) = dr_dy_from(y3,x3,y4,x4);
A(3,3) = -1;

A(4,1) = dr_dx_from(y3,x3,y5,x5);
A(4,2) = dr_dy_from(y3,x3,y5,x5);
A(4,3) = -1;

A(5,1) = dr_dx_from(y3,x3,y6,x6);
A(5,2) = dr_dy_from(y3,x3,y6,x6);
A(5,3) = -1;
%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 the normal equations
x_hat = Q_xx * n;

%Update
X_hat = X_0 + x_hat;
X_0 = X_hat;

x3 = X_hat(1);
y3 = X_hat(2);
w3 = X_hat(3);


 %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;
```

```
    %Objective function
    vTPv = v' * P * v;

    %Functional relationships without the observations

    phi_X_hat(1,1) = direction(y3, x3, y1, x1, w3);
    phi_X_hat(2,1) = direction(y3, x3, y2, x2, w3);
    phi_X_hat(3,1) = direction(y3, x3, y4, x4, w3);
    phi_X_hat(4,1) = direction(y3, x3, y5, x5, w3);
    phi_X_hat(5,1) = direction(y3, x3, y6, x6, w3);

    %Check 2
    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!
```

```
%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));        %[m]

%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));

%--------------------------------------------------------------------------
%   Task 2
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   Observations and initial values for the unknowns
%--------------------------------------------------------------------------
%Load files
dir = load('directions.txt');                   %[gon]
coord = load('Points.txt');             %[m]     %Error-free

for i = 1:size(coord,1)
    eval(['y' num2str(coord(i,1)) '=' num2str(coord(i,2)) ';']);
    eval(['x' num2str(coord(i,1)) '=' num2str(coord(i,3)) ';']);
end

%Vector of observations

L_ang(1) = (dir(3,3)-dir(2,3))*pi/200; %[rad]
L_ang(2) = (dir(4,3)-dir(3,3))*pi/200; %[rad]
L_ang(3) = (dir(5,3)-dir(4,3))*pi/200; %[rad]
L_ang(4) = (dir(1,3)-dir(5,3))*pi/200; %[rad]
L_ang = L_ang';

%Number of observations
no_n = length(L_ang);

%Initial values for the unknowns
x3 = 200;
y3 = 500;
w3 = 0;

%Vector of initial values for the unknowns
X_0_ang = [x3 y3]';

%Number of unknowns
no_u = length(X_0_ang);

%Redundancy
r = no_n-no_u;

%--------------------------------------------------------------------------
%  Stochastic model
%--------------------------------------------------------------------------
s_dir = 0.001 * pi / 200;
```

```
%VC Matrix of the observations
S_LL_dir = s_dir^2*eye(5);
F = [0 -1 1 0 0;
     0 0 -1 1 0;
     0 0 0 -1 1;
     1 0 0 0 -1];
S_LL = F * S_LL_dir * F';   % Variance-covariance propagation


%Theoretical standard deviation
sigma_0 = 1e-5;     %a priori

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

%Weight matrix
P = inv(Q_LL);

%--------------------------------------------------------------------------
%  Adjustment
%--------------------------------------------------------------------------
%break-off conditions
epsilon = 10^-5;
delta = 10^-13;
max_x_hat = Inf;
Check2 = Inf;

%Number of iterations
iteration = 0;

while max_x_hat>epsilon || Check2>delta

     %Observations as functions of the approximations for the unknowns

     L_0_ang(1,1) = direction(y3, x3, y4, x4, w3) - direction(y3, x3, y2, x2, w3);
     L_0_ang(2,1) = direction(y3, x3, y5, x5, w3) - direction(y3, x3, y4, x4, w3);
     L_0_ang(3,1) = direction(y3, x3, y6, x6, w3) - direction(y3, x3, y5, x5, w3);
     L_0_ang(4,1) = direction(y3, x3, y1, x1, w3) - direction(y3, x3, y6, x6, w3);

     %Vector of reduced observations
     l = L_ang - L_0_ang;

     %Design matrix with the elements from the Jacobian matrix J
     A_ang(1,1) = der_ang_x(y3,x3,y4,x4,y2,x2);
     A_ang(1,2) = der_ang_y(y3,x3,y4,x4,y2,x2);
     A_ang(2,1) = der_ang_x(y3,x3,y5,x5,y4,x4);
     A_ang(2,2) = der_ang_y(y3,x3,y5,x5,y4,x4);
     A_ang(3,1) = der_ang_x(y3,x3,y6,x6,y5,x5);
     A_ang(3,2) = der_ang_y(y3,x3,y6,x6,y5,x5);
```

```
    A_ang(4,1) = der_ang_x(y3,x3,y1,x1,y6,x6);
    A_ang(4,2) = der_ang_y(y3,x3,y1,x1,y6,x6);


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

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

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

    %Solution of the normal equations
    x_hat = Q_xx * n;

    %Update
    X_hat_ang = X_0_ang + x_hat;
    X_0_ang = X_hat_ang;

    x3 = X_hat_ang(1);
    y3 = X_hat_ang(2);


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

    %Vector of residuals
    v_ang = A_ang * x_hat - l;

    %Vector of adjusted observations
    L_hat_ang = L_ang + v_ang;

    %Objective function
    vTPv = v_ang' * P * v_ang;

    %Functional relationships without the observations

    phi_X_hat_ang(1,1) = direction(y3, x3, y4, x4, w3) - direction(y3, x3, y2, x2,
w3);
    phi_X_hat_ang(2,1) = direction(y3, x3, y5, x5, w3) - direction(y3, x3, y4, x4,
w3);
    phi_X_hat_ang(3,1) = direction(y3, x3, y6, x6, w3) - direction(y3, x3, y5, x5,
w3);
    phi_X_hat_ang(4,1) = direction(y3, x3, y1, x1, w3) - direction(y3, x3, y6, x6,
w3);

    %Check 2
    Check2 = max(abs(L_hat_ang - phi_X_hat_ang));
```

```
    %Update number of iterations
    iteration = iteration+1;

end

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

```
Everything is fine!
```

```
%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_ang = sqrt(diag(S_XX_hat));        %[m]

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

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

%Standard deviation of the adjusted observations
s_L_hat_ang = 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_ang = sqrt(diag(S_vv));

table(X_hat(1:2,:), X_hat_ang, s_X(1:2,:), s_X_ang, 'RowNames',{'x3' 'y3'},
'VariableNames',["X_hat", "X_hat_ang", "s_X", "s_X_ang"])
```

ans = 2×4 table

	X_hat	X_hat_ang	s_X	s_X_ang
1 x3	242.858489795327	242.85848979534	0.0043596029472373	0.00435960294760575
2 y3	493.696874627423	493.696874627512	0.0121312693890734	0.012131269405953

Conclusion: the results of both adjustments are the same within desired precision, which proves that all the computations were correct.

```
table(L, v, L_hat, s_v, s_L_hat, 'RowNames', {'r31' 'r32' 'r34' 'r35' 'r36'})
```

ans = 5×5 table

	L	v	L_hat	s_v	s_L_hat
1 r31	3.25012525499336	8.45337966877599e-06	3.25013370837303	5.9776055480953e-06	2.46471889537925e-05
2 r32	0.73046270346045	-6.81478321501533e-06	0.730455888677235	1.15513465914223e-05	2.25783543097677e-05
3 r34	1.32959898001921	3.75033181377967e-06	1.32960273035103	2.13388619627366e-05	1.37065189434752e-05
4 r35	1.81466402732613	2.11796350333525e-05	1.81468520696116	1.75081349428696e-05	1.83488665109002e-05
5 r36	2.44398786769324	-2.65685633008928e-05	2.44396129912994	1.88516362053508e-05	1.69655976550688e-05

```
table(L_ang, v_ang, L_hat_ang, s_v_ang, s_L_hat_ang, 'RowNames',{'alpha' 'beta'
'gamma' 'delta'})
```

ans = 4×5 table

	L_ang	v_ang	L_hat_ang	s_v_ang	s_L_hat_ang
1 alpha	0.599136276558763	1.05651149304862e-05	0.599146841673694	3.25408341669759e-05	1.50839482409515e-05
2 beta	0.485065047306917	1.74293031184671e-05	0.485082476610036	3.26572192952079e-05	1.48302869400101e-05
3 gamma	0.629323840367107	-4.77481983065219e-05	0.629276092168801	3.45875640709731e-05	9.49377661706135e-06
4 delta	0.806137387300121	3.50219431445808e-05	0.806172409243266	2.48107237540078e-05	2.59009530639443e-05