```
%--------------------------------------------------------------------------
%
%           ADJUSTMENT THEORY I
%     Exercise 8: Adjustment Calculation - part III
%
%    Author          : Anastasia Pasioti
%    Version         : October 09, 2018
%    Last changes    : January 03, 2022
%
%--------------------------------------------------------------------------

clc;
clear all;
close all;

%--------------------------------------------------------------------------
%    Task 2 - Copper
%--------------------------------------------------------------------------
disp('Task 2 - Non-linear adjustment problem!')
```

```
Task 2 - Non-linear adjustment problem!
```

```

%--------------------------------------------------------------------------
%    Observations and initial values for the unknowns
%--------------------------------------------------------------------------
%Functional Model
%a = nthroot(V,3)
%m = V * p

%Vector of observations
L = [11.60, 15.15]'; %a,m
p = 8.93/1000;

%Standart errors
a_s = 0.05; %mm
m_s = 0.05; %g

%Number of observations
no_n = length(L);

%Initial values for the unknowns
V = 11.60^3;

%Test for same units of measurement
if (abs((V * p) - L(2))<L(2))
    disp('Meassurements seems to have the same units of measurement')
else
    error('Meassurements DONT seems to have the same units of measurement')
end
```

```
Meassurements seems to have the same units of measurement
```

```
%Number of unknowns
no_u = length(V);

%Redundancy
r = no_n-no_u;

%--------------------------------------------------------------------------
%  Stochastic model
%--------------------------------------------------------------------------

%VC Matrix of the observations
S_LL = diag([a_s,m_s]);

%Theoretical standard deviation
sigma_0 = 1;    %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^-12;
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) = nthroot(V,3); %a
     L_0(2) = V * p; %m

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

     %Design matrix with the elements from the Jacobian matrix J
     A(1,1) = 1 / (3*V^(2/3));%a
     A(2,1) = p; %m

     %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
    V = V+x_hat;

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

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

    %Vector of adjusted observations
    L_hat = L+res;

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

    %functional relatinships without the observations

    phi_X_hat(1) = nthroot(V,3);
    phi_X_hat(2) = V * p;

    %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));

%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));

%Tables
table(L,L_hat,s_L_hat,res,s_v,'VariableNames',...
{'L'...
'L_hat',...
's_L_hat',...
'v',...
's_v'}, ...
'RowNames',{'a [mm]' 'm [g]'})
```

ans = 2×5 table

	L	L_hat	s_L_hat	v	s_v
1 a [mm]	11.6000	11.9055	0.0805	0.3055	0.3055
2 m [g]	15.1500	15.0695	0.3055	-0.0805	0.0805

```
table(11.60^3,V,s_X,'VariableNames',...
{'X',...
'X_hat',...
's_X_hat'},...
'RowNames',{'V [Volume]'})
```

ans = 1×3 table

	X	X_hat	s_X_hat
1 V [Volume]	1.5609e+03	1.6875e+03	34.2161