1.

Model: $z = a\ x + b\ y + c$

→ z : output , x:input1 ,y:input2

We want to solve (a , b , c).

→

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$Y \quad = \quad A \quad X$$

→ $X = (A^T A)^{-1} A^T\ Y$

Matlab code:

```
%% for HW5-1
data = xlsread('HW5-1.xls');
A = ones(50,3);
for i = 1:50
    A(i,1) = data(i,1);
    A(i,2) = data(i,2);
end
Y = data(:, 3);
x = (((A.')*A)\(A.'))*Y;
disp(x); % Show model coefficients(a,b,c) for z = 
ax+by+c
```

2.

$$ {}^S_E\hat{\boldsymbol{q}} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} $$

$$ {}^E\hat{\boldsymbol{d}} = \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix} = [0\ 0\ 0\ \text{-}9.8] $$

→

$$ \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) = \begin{bmatrix} 2d_x(\frac{1}{2} - q_3^2 - q_4^2) + 2d_y(q_1q_4 + q_2q_3) + 2d_z(q_2q_4 - q_1q_3) - s_x \\ 2d_x(q_2q_3 - q_1q_4) + 2d_y(\frac{1}{2} - q_2^2 - q_4^2) + 2d_z(q_1q_2 + q_3q_4) - s_y \\ 2d_x(q_1q_3 + q_2q_4) + 2d_y(q_3q_4 - q_1q_2) + 2d_z(\frac{1}{2} - q_2^2 - q_3^2) - s_z \end{bmatrix} $$

$$ \boldsymbol{J}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}) = \begin{bmatrix} 2d_yq_4 - 2d_zq_3 & 2d_yq_3 + 2d_zq_4 & -4d_xq_3 + 2d_yq_2 - 2d_zq_1 & -4d_xq_4 + 2d_yq_1 + 2d_zq_2 \\ -2d_xq_4 + 2d_zq_2 & 2d_xq_3 - 4d_yq_2 + 2d_zq_1 & 2d_xq_2 + 2d_zq_4 & -2d_xq_1 - 4d_yq_4 + 2d_zq_3 \\ 2d_xq_3 - 2d_yq_2 & 2d_xq_4 - 2d_yq_1 - 4d_zq_2 & 2d_xq_1 + 2d_yq_4 - 4d_zq_3 & 2d_xq_2 + 2d_yq_3 \end{bmatrix} $$

$$ \nabla \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) = \boldsymbol{J}^T({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}})\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) $$

→

$$ {}^S_E\boldsymbol{q}_{k+1} = {}^S_E\hat{\boldsymbol{q}}_k - \mu \frac{\nabla \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})}{\left\| \nabla \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) \right\|}, \ k = 0, 1, 2 \ldots n $$

Matlab:

```
clc;
q = [1 0 0 0];
acc = xlsread('HW5-2.xls');
deltaT = 0.001;
iter = 1000;
qua_predict = zeros(100,4);
%Update loop
for j = 1:100
    for i = 1:iter
        %Normalize the accelerometer measurement
        norm = sqrt(acc(j,1)*acc(j,1) +
acc(j,2)*acc(j,2) + acc(j,3)*acc(j,3));
        a_x = acc(j,1)/norm;
        a_y = acc(j,2)/norm;
        a_z = acc(j,3)/norm;
        % Compute the objective function
        f_1 = -2*(q(2)*q(4) - q(1)*q(3)) - a_x;
        f_2 = -2*(q(1)*q(2) + q(3)*q(4)) - a_y;
        f_3 = -2*(0.5 - q(2)*q(2) - q(3)*q(3)) - a_z;
        f = [f_1;f_2;f_3];
        % Compute Jacobian
        J = [2*q(3) -2*q(4) +2*q(1) -2*q(2); -2*q(2) -
2*q(1) -2*q(4) -2*q(3); 0 4*q(2) 4*q(3) 0];
        %Compute gradient
        G = J'*f;
        %Normalize the gradient
        norm = sqrt(G(1,1)*G(1,1) + G(2,1)*G(2,1) +
G(3,1)*G(3,1) + G(4,1)*G(4,1));
        G = G /norm;
        %Find the quaternion
        q = q - deltaT * G';
        %Normalize the quaternion
        norm = sqrt(q(1)*q(1) + q(2)*q(2) + q(3)*q(3)
+ q(4)*q(4));
        q = q /norm;
        % Check error function
        %fp(i) = f_3;
```

```
    end
    % Your result!
    qua_predict(j,1) = q(1);
    qua_predict(j,2) = q(2);
    qua_predict(j,3) = q(3);
    qua_predict(j,4) = q(4);
    %Check your accleration
    quat1 = quaternion(q(1),q(2),q(3),q(4));
    [a,b,c,d] = parts(quat1);
    quat1inv=quatinv([a,b,c,d]);
    quat1inv=quaternion(quat1inv);
    quat2 = quaternion(0,0,0,-9.8);
    %%Project gravity to current frame
    quat2=quat1inv*quat2*quat1;
    [A(j,:),B(j,:),C(j,:),D(j,:)] = parts(quat2);
end
M=[B,C,D];
xlswrite("acc.xlsx",M);
```