```
clear; clc; close all;

raw_report = readmatrix("path/to/file.xls");

% data analysis of the deviations of a 11x11 grid of circles printed with
% baxter

% all values are measured in mm

% initialize arrays
x_nom_raw = zeros(121,1); % nominal x-position of circle center
y_nom_raw = zeros(121,1); % nominal y-position of circle center
x_meas_raw = zeros(121,1); % measured x-position of circle center
y_meas_raw = zeros(121,1); % measured y-position of circle center
x_dev_raw = zeros(121,1); % deviation of measured x-position from nominal x-position
y_dev_raw = zeros(121,1); % deviation of measured y-position from nominal y-position
d_meas_raw = zeros(121,1); % measured diameter of circles
d_dev_raw = zeros(121,1); % deviation of measured diameter from nominal diameter

% fill arrays
for i = 1:121
    x_nom_raw(i) = raw_report(i*4-1, 3);
    y_nom_raw(i) = -raw_report(i*4, 3); % negative to switch from right hand to left ha
    x_meas_raw(i) = raw_report(i*4-1, 2);
    y_meas_raw(i) = -raw_report(i*4, 2); % negative to switch from right hand to left h
    x_dev_raw(i) = raw_report(i*4-1, 4);
    y_dev_raw(i) = -raw_report(i*4, 4);
    d_meas_raw(i) = raw_report(i*4-2, 2);
    d_dev_raw(i) = raw_report(i*4-2, 4);
end

% initialise good measurements
x_nom = zeros(1,1); % nominal x-position of circle center
y_nom = zeros(1,1); % nominal y-position of circle center
x_meas = zeros(1,1); % measured x-position of circle center
y_meas = zeros(1,1); % measured y-position of circle center
x_dev = zeros(1,1); % deviation of measured x-position from nominal x-position
y_dev = zeros(1,1); % deviation of measured y-position from nominal y-position
d_meas = zeros(1,1); % measured diameter of circles
d_dev = zeros(1,1); % deviation of measured diameter from nominal diameter

% initialise outliers
x_nom_out = zeros(1,1); % nominal x-position of circle center
y_nom_out = zeros(1,1); % nominal y-position of circle center
x_meas_out = zeros(1,1); % measured x-position of circle center
y_meas_out = zeros(1,1); % measured y-position of circle center
x_dev_out = zeros(1,1); % deviation of measured x-position from nominal x-position
y_dev_out = zeros(1,1); % deviation of measured y-position from nominal y-position
d_meas_out = zeros(1,1); % measured diameter of circles
d_dev_out = zeros(1,1); % deviation of measured diameter from nominal diameter

d_avg = mean(d_meas_raw);
```

```
% filter outliers
for i = 1:121
    if ((4.975 < d_meas_raw(i)) & (d_meas_raw(i) < 5.075))
        x_nom = [x_nom x_nom_raw(i)];
        y_nom = [y_nom y_nom_raw(i)];
        x_meas = [x_meas x_meas_raw(i)];
        y_meas = [y_meas y_meas_raw(i)];
        x_dev = [x_dev x_dev_raw(i)];
        y_dev = [y_dev y_dev_raw(i)];
        d_meas = [d_meas d_meas_raw(i)];
        d_dev = [d_dev d_dev_raw(i)];
    else
        x_nom_out = [x_nom_out x_nom_raw(i)];
        y_nom_out = [y_nom_out y_nom_raw(i)];
        x_meas_out = [x_meas_out x_meas_raw(i)];
        y_meas_out = [y_meas_out y_meas_raw(i)];
        x_dev_out = [x_dev_out x_dev_raw(i)];
        y_dev_out = [y_dev_out y_dev_raw(i)];
        d_meas_out = [d_meas_out d_meas_raw(i)];
        d_dev_out = [d_dev_out d_dev_raw(i)];
    end
end

% remove obsolete first zero
x_nom = x_nom(2:end);
y_nom = y_nom(2:end);
x_meas = x_meas(2:end);
y_meas = y_meas(2:end);
x_dev = x_dev(2:end);
y_dev = y_dev(2:end);
d_meas = d_meas(2:end);
d_dev = d_dev(2:end);

x_nom_out = x_nom_out(2:end);
y_nom_out = y_nom_out(2:end);
x_meas_out = x_meas_out(2:end);
y_meas_out = y_meas_out(2:end);
x_dev_out = x_dev_out(2:end);
y_dev_out = y_dev_out(2:end);
d_meas_out = d_meas_out(2:end);
d_dev_out = d_dev_out(2:end);

% nominal circle diameter is the same for all measurments
d_nom = 5;
```

Plotting

```
s = scatter(x_meas, y_meas)
hold on
scatter(x_meas_out, y_meas_out, 'rx')
hold on
xlabel('x [mm]')
ylabel('y [mm]')
```

```
axis([-60, 60, -60, 60])
s.SizeData = 85;
hold off

plot(d_meas_raw)
hold
plot([1, 121], [5.075, 5.075], 'r')
hold on
plot([1, 121], [4.975, 4.975], 'r')
hold on
axis([0 122 4.8 6.2])
xlabel('Circle no.')
ylabel('Diameter [mm]')
hold off
```

Try different models

```
[fitresult, gof] = createFits(x_nom, y_nom, x_meas, y_meas); % function definition belo

%% read out parameters and corresponing confidence intervals for different fits

% fit model xm = qx00 + qx10*xn
f1qx00 = fitresult{1}.qx00
f1qx10 = fitresult{1}.qx10
f1xr2 = gof(1).rsquare
f1xSSE = gof(1).sse
confint(fitresult{1}) % confidence intervals, first column corresponds to first paramet

% fit model ym = qy00 + qy10*yn
f5qy00 = fitresult{5}.qy00
f5qy10 = fitresult{5}.qy10
f1yr2 = gof(5).rsquare
f1ySSE = gof(5).sse
confint(fitresult{5})

% fit model xm = qx00 + qx10*xn + qx01*yn
f2qx00 = fitresult{2}.qx00
f2qx10 = fitresult{2}.qx10
f2qx01 = fitresult{2}.qx01
f2xr2 = gof(2).rsquare
f2xSSE = gof(2).sse
confint(fitresult{2})

% fit model ym = qy00 + qy10*yn + qy01*xn
f6qy00 = fitresult{6}.qy00
f6qy10 = fitresult{6}.qy10
f6qy01 = fitresult{6}.qy01
f2yr2 = gof(6).rsquare
f2ySSE = gof(6).sse
confint(fitresult{6})
```

```
% fit model xm = qx00 + qx10*xn + qx01*yn + qx12*xn*yn^2
f4qx00 = fitresult{4}.qx00
f4qx10 = fitresult{4}.qx10
f4qx01 = fitresult{4}.qx01
f4qx12 = fitresult{4}.qx12
f4xr2 = gof(4).rsquare
f4xSSE = gof(4).sse
confint(fitresult{4})

% fit model ym = qy00 + qy10*yn + qy01*xn + qy12*yn*xn^2
f8qy00 = fitresult{8}.qy00
f8qy10 = fitresult{8}.qy10
f8qy01 = fitresult{8}.qy01
f8qy12 = fitresult{8}.qy12
f4yr2 = gof(8).rsquare
f4ySSE = gof(8).sse
confint(fitresult{8})
```

Solve for nominal values

```
syms q_x00 q_x10 q_x01 q_y00 q_y10 q_y01 xm ym xn yn;
assume(q_x00,'real'); assume(q_x10,'real'); assume(q_x01,'real');
assume(q_y00,'real'); assume(q_y10,'real'); assume(q_y01,'real');
assume(xn,'real'); assume(xm,'real');
assume(yn,'real'); assume(ym,'real');

% We chose the linear two variable fit model as the best option
eqx = eq(xm, q_x00 + q_x10*xn + q_x01*yn)
eqy = eq(ym, q_y00 + q_y10*yn + q_y01*xn)

sol = solve([eqx, eqy], [xn, yn])

% verify solution by substituting back ing
simplify(q_x00 + q_x10*sol.xn + q_x01*sol.yn)
simplify(q_y00 + q_y10*sol.yn + q_y01*sol.xn)

% see if matlab can solve with third degree terms as well
syms q_x12 q_y12;
assume(q_x12,'real'); assume(q_y12,'real');

eqxp = eq(xm, q_x00 + q_x10*xn + q_x01*yn + q_x12*xn*yn^2)
eqyp = eq(ym, q_y00 + q_y10*yn + q_y01*xn + q_y12*yn*xn^2)

solp = solve([eqxp, eqyp], [xn, yn])
solp.xn(1)
solp.yn(1)
% no success
```

Fit function definition

```
function [fitresult, gof] = createFits(x_nom, y_nom, x_meas, y_meas)
%CREATEFITS(X_NOM,Y_NOM,X_MEAS,Y_MEAS)
```

```
% Fits the calibration data to a range of polynomials to evaluate
% which model suits the purpose best.

%  Output:
%      fitresult : a cell-array of fit objects representing the fits.
%      gof : structure array with goodness-of fit info.
%
%  See also FIT, CFIT, SFIT.

%  Auto-generated by MATLAB on 30-Mar-2022 14:24:26

%% Initialization.

% Initialize arrays to store fits and goodness-of-fit.
fitresult = cell( 8, 1 );
gof = struct( 'sse', cell( 8, 1 ), ...
    'rsquare', [], 'dfe', [], 'adjrsquare', [], 'rmse', [] );


%% Fit: 'x-00-10-cal-fit'.
[xData, yData, xzData] = prepareSurfaceData( x_nom, y_nom, x_meas );

% Set up fittype and options.
ft = fittype( 'qx00 + qx10*x + 0*y', 'independent', {'x', 'y'}, 'dependent', 'z' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.959743958516081 0.340385726666133];

% Fit model to data.
[fitresult{1}, gof(1)] = fit( [xData, yData], xzData, ft, opts );


%% Fit: 'x-00-10-01-cal-fit'.

% Set up fittype and options.
ft = fittype( 'qx00 + qx10*x + qx01*y', 'independent', {'x', 'y'}, 'dependent', 'z' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.765516788149002 0.317099480060861 0.950222048838355];

% Fit model to data.
[fitresult{2}, gof(2)] = fit( [xData, yData], xzData, ft, opts );


%% Fit: 'x-00-10-01-21-cal-fit'.

% Set up fittype and options.
ft = fittype( 'qx00 + qx10*x + qx01*y + qx21*x^2*y', 'independent', {'x', 'y'}, 'depend
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.585267750979777 0.223811939491137 0.751267059305653 0.890903252535

% Fit model to data.
[fitresult{3}, gof(3)] = fit( [xData, yData], xzData, ft, opts );
```

```
%% Fit: 'x-00-10-12-cal-fit'.

% Set up fittype and options.
ft = fittype( 'qx00 + qx10*x + qx01*y + qx12*x*y^2', 'independent', {'x', 'y'}, 'depend
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.794284540683907 0.311215042044805 0.528533135506213 0.654079098476

% Fit model to data.
[fitresult{4}, gof(4)] = fit( [xData, yData], xzData, ft, opts );


%% Fit: 'y-00-10-cal-fit'.
[xData, yData, yzData] = prepareSurfaceData( x_nom, y_nom, y_meas ); % change dependent

% Set up fittype and options.
ft = fittype( 'qy00 + qy10*y + 0*x', 'independent', {'x', 'y'}, 'dependent', 'z' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.254282178971531 0.814284826068816];

% Fit model to data.
[fitresult{5}, gof(5)] = fit( [xData, yData], yzData, ft, opts );


%% Fit: 'y-00-10-01-cal-fit'.


% Set up fittype and options.
ft = fittype( 'qy00 + qy10*y + qy01*x', 'independent', {'x', 'y'}, 'dependent', 'z' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.196595250431208 0.251083857976031 0.616044676146639];

% Fit model to data.
[fitresult{6}, gof(6)] = fit( [xData, yData], yzData, ft, opts );


%% Fit: 'y-00-10-01-21-cal-fit'.

% Set up fittype and options.
ft = fittype( 'qy00 + qy10*y + qy01*x + qy21*y^2*x', 'independent', {'x', 'y'}, 'depend
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.585264091152724 0.54972360829114 0.91719366382981 0.380445846975

% Fit model to data.
[fitresult{7}, gof(7)] = fit( [xData, yData], yzData, ft, opts );


%% Fit: 'y-00-10-01-12-cal-fit'.
```

```
% Set up fittype and options.
ft = fittype( 'qy00 + qy10*y + qy01*x + qy12*y*x^2', 'independent', {'x', 'y'}, 'depend
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.181847028302852 0.26380291652199 0.145538980384717 0.136068558708

% Fit model to data.
[fitresult{8}, gof(8)] = fit( [xData, yData], yzData, ft, opts );
end
```