FITQUADMODEL Fits a quadratic model to the response data y using the columns of X as regressors. X has one or two columns; the same number of rows as X.

```
function [modelCoeffs, fh] = fitQuadModelFinal(X, y, showplot)

    narginchk(2, 3)
    if nargin < 3
        showplot = false;
    end

    validateattributes(X, {'double'}, {'real', '2d', 'nonempty'}, mfilename(), 'X', 1)
    validateattributes(y, {'double'}, {'real', 'column', 'nonempty'}, mfilename(), 'y', 2)
    validateattributes(showplot, {'logical'}, {'scalar'}, mfilename(), 'showplot', 3)

    if (size(X, 1) < 3)
        error('fitQuadModel:XTooSmall', 'X must have at least 3 rows.' )
    end

    if (size(X, 1) ~= numel(y))
        error('fitQuadModel:DimMismatch', ...
            'X and y must have the same number of observations.' )
    end
    if ~(ismember(size(X, 2), [1, 2]))
        error('fitQuadModel:TooManyXCols', ...
            'X must be a one or two column matrix.' )
    end
    if any( isinf( X(:) ) ) || any( isinf( y ) )
        error('F02_fitQuadModel:expectedNoninfinite', 'X and y must not contain infinite values
    end

    %Clean the data.
    [XClean, yClean] = removeNaNs(X, y);
    if isempty(yClean)
        error('fitQuadModel:AllNaNs', 'All observations contain NaNs.')
    end

    % Fit the model.
    modelCoeffs = fitModel(XClean, yClean);
    % Visualise the results.
    if showplot
        fh = visualiseResults(X, y, XClean, modelCoeffs);
    else
        fh = [];
    end % if

end  % end main function fitQuadMode
```

```
function [XClean, yClean] = removeNaNs(X, y)

    missingIdx = isnan(y) | any(isnan(X), 2);
    XClean = X(~missingIdx, :);
    yClean = y(~missingIdx);
```

```matlab
end % removeNaNs

function modelCoeffs = fitModel(XClean, yClean)

    n = size(XClean, 2);

    switch n
        case 1
            designMat = [ones(size(XClean)), XClean, XClean.^2];

        case 2
            x1 = XClean(:, 1); x2 = XClean(:, 2);
            designMat = [ones(size(x1)), x1, x2, x1.^2, x2.^2, x1.*x2];
    end % switch/case

    modelCoeffs = designMat\yClean;

end % fitModel

function fh = visualiseResults(X, y, XClean, modelCoeffs)

    n = size(XClean, 2);
    fh = figure;
    switch n
        case 1
            modelFun = @(c, x) c(1) + c(2)*x + c(3)*x.^2;
            scatter(X, y, 'bx')
            hold on
            plot(XClean, modelFun(modelCoeffs, XClean), 'r*')
            hold off
        case 2
            modelFun = @(c, x1, x2) c(1) + c(2)*x1 + c(3)*x2 + c(4)*x1.^2 + ...
                        c(5)*x2.^2 + c(6)*x1.*x2;
            scatter3(X(:,1), X(:,2), y, 'kx')
            hold on
            x1Vec = linspace(min(XClean(:,1)), max(XClean(:,1)));
            x2Vec = linspace(min(XClean(:,2)), max(XClean(:,2)));
            [X1Grid, X2Grid] = meshgrid(x1Vec, x2Vec);
            YGrid = modelFun(modelCoeffs, X1Grid, X2Grid);
            surf(X1Grid, X2Grid, YGrid, 'FaceColor', 'interp', 'EdgeAlpha', 0)
            hold off
    end

end % visualiseResults
```