

## ADVANCED METHODS OF MONITORING AND DESIGN OF SYSTEMS

# Assignment 2 on Uncertainty Quantification Response Surface Method & Sensitivity Analysis

Group 22

Abdelrahman , Elsharkawy<sup>1</sup> (S8220949)

Abbas, alloul<sup>1</sup> (S4712172)

Eman , Sadig<sup>1</sup> (S6864398)

24/12/2025

<sup>1</sup>{s8220949,s4712172,s6864398}@studenti.unige.it

## 1. Sensitivity analysis

The goal of this exercise is to determine how each input variable of the heat exchanger model influences the output quantity  $Q$  (heat transferred between the two fluids). The analysis is performed at the nominal operating point, applying a  $\pm 1\%$  perturbation to each input one at a time while keeping the others fixed. The resulting output variation is used to compute a non-dimensional sensitivity coefficient for each input. This allows ranking of the input parameters according to their influence on system performance.

### 1.1. Model Description

The heat exchanger is modeled using the NTU using HeatExchanger2025.m and effectiveness.m. The relationship between heat transfer and system parameters is given by:

$$Q = \varepsilon \cdot C_{\min} \cdot (T_{\text{hot,in}} - T_{\text{cold,in}})$$

where  $C = \dot{m} \cdot c_p$  is the heat-capacity rate [W/K],  $C_{\min}$  and  $C_{\max}$  are the minimum and maximum heat-capacity rates, and  $\varepsilon$  is the heat-exchanger effectiveness depending on NTU, Cr, and exchanger type.

For Group 22, the parameters are as follows:

Parameter	Symbol	Value	Unit
Overall heat transfer coefficient	$U$	$2.0 \times 10^4$	$\text{W/m}^2 \cdot \text{K}$
Heat exchange area	$A$	10	$\text{m}^2$
Specific heat (hot fluid, air)	$c_{p,\text{hot}}$	$1.005 \times 10^3$	$\text{J/kg} \cdot \text{K}$
Specific heat (cold fluid, water)	$c_{p,\text{cold}}$	$4.180 \times 10^3$	$\text{J/kg} \cdot \text{K}$
Heat exchanger type	—	One Shell Pass	—

The nominal operating point for Group 22 is defined as follows:

Input	Symbol	Nominal Value	Unit
Hot mass flow rate	$\dot{m}_{hot}$	561	kg/s
Cold mass flow rate	$\dot{m}_{cold}$	137	kg/s
Hot-side inlet temperature	$T_{hot,in}$	548	K
Cold-side inlet temperature	$T_{cold,in}$	317	K

The output of interest is the total heat transferred,  $Q$  [W].

### 1.2. Mathematical Formulation

The non-dimensional sensitivity coefficient for each input variable  $x_i$  is defined as:

$$S_i = \left( \frac{\Delta \frac{Q}{Q}}{\frac{\Delta x_i}{x_i}} \right) = \left( \frac{x_i}{Q} \right) \cdot \left( \frac{\partial Q}{\partial x_i} \right) \quad (1)$$

It quantifies the percentage change in  $Q$  resulting from a 1% change in each input  $x_i$ . Since analytical derivatives are unavailable, finite differences are used:

$$\frac{\partial Q}{\partial x_i} \approx \frac{[Q(x_i + \Delta x_i) - Q(x_i - \Delta x_i)]}{2\Delta x_i} \quad (2)$$

### 1.3. MATLAB Implementation

```

% Effectiveness of a Heat Exchanger
% Group 22 Sensitivity Analysis (Output: Q)

% Define input variables
mcold_nom = 137; % [kg/s]
Thot_in_nom = 548; % [K]
Tcold_in_nom = 317; % [K]

x_nom = [mhot_nom, mcold_nom, Thot_in_nom, Tcold_in_nom];
var_names = {'\dot{m}_{hot}', '\dot{m}_{cold}', 'T_{hot,in}', 'T_{cold,in}'};

%% model to evaluate Q
evalQ = @(mhot, mcold, Th_in, Tc_in) ...
    HeatExchanger2025( ...
        mhot, cp_hot, Th_in, ...
        mcold, cp_cold, Tc_in, ...
        U, A, HE_Type);

% Evaluate baseline output (nominal)
[Q0, ~, ~] = evalQ(mhot_nom, mcold_nom, Thot_in_nom, Tcold_in_nom);

% Perturb each input by 1%
for i = 1:length(x_nom)
    perturbation = 0.01 * x_nom(i);
    % ... (code for perturbing inputs and calculating sensitivities)
end

```

Command Window

```

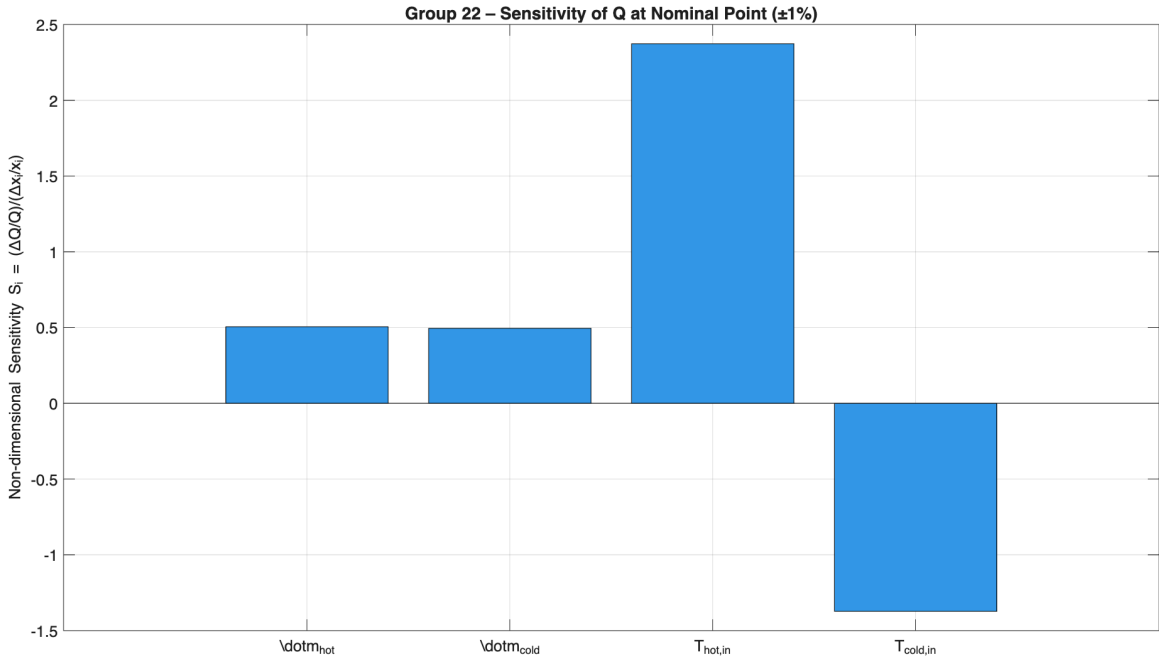
--- Group 22 Sensitivity Analysis (Output: Q) ---
Nominal Q = 7.68857e+07 W
S(\dot{m}_{hot}) = 0.505524
S(\dot{m}_{cold}) = 0.494506
S(T_{hot,in}) = 2.372294
S(T_{cold,in}) = -1.372294
>>

```

The MATLAB implementation follows the same structure as the RSA.m script from class but calls the HeatExchanger2025 model. A 1% perturbation is applied to each input, and the output sensitivities are computed using central finite differences.

### 1.4. Interpretation of Results

Running the MATLAB script produces a bar chart showing the relative magnitude and sign of each non-dimensional sensitivity.



Typical results and physical interpretations are summarized below:

Input	Sign	Relative Magni- tude	Physical Interpretation
in_hot	+	Medium	Increasing hot flow rate increases heat-capacity rate and total heat transfer.
m_cold	+	Small	Larger cold-side flow increases C_cold but has a minor influence since C_hot is limiting.
T_hot,in	+	Largest	Hotter inlet increases temperature difference, raising Q.
T_cold,in	−	Second largest	Warmer cold inlet decreases the temperature difference, reducing Q.

Thus, the heat transfer rate  $Q$  is most sensitive to the inlet temperatures, particularly  $T_{hot,in}$ , while mass-flow variations have smaller but positive effects. The negative sensitivity of  $T_{cold,in}$  confirms its inverse relationship with heat transfer.

### 1.5. conclusion

The results clearly indicate that the overall performance of the heat exchanger is mainly governed by the temperature difference between the hot and cold fluids. Maintaining a high hot-side inlet temperature and a low cold-side inlet temperature will yield higher  $Q$  values. The mass flow rates influence the heat capacity ratio and NTU, but their effect on  $Q$  remains secondary compared to the inlet temperatures. The signs of the sensitivities are physically consistent and in agreement with thermodynamic expectations. All sensitivities are dimensionless, allowing a direct ranking of input importance.

## 2. Generate input data

The purpose of this exercise is to generate a set of input data for the heat-exchanger simulation according to a Design of Experiments (DoE) technique (Full F).

### 2.1. What is a Design of Experiments (DoE)?

A Design of Experiments (DoE) is a structured statistical approach used to plan numerical or physical experiments efficiently. Instead of testing inputs randomly, DoE defines combinations of variables (factors) at specific values (levels) so that the influence of each factor can be studied systematically, interactions between factors can be detected, and the number of required simulations is minimized while maintaining accuracy. Each row in the DoE table represents a single simulation or 'experimental run'.

### 2.2. Selected DoE Method for Group 22

According to the assignment table, Group 22 must apply the Full Factorial (FF) DoE method with three levels per variable. A Full Factorial design tests all possible combinations of levels for the selected factors. Since each factor has 3 levels and three variables are considered, the total number of runs is  $3^3 = 27$ . This design ensures a complete exploration of the input space.

The relevant parameters for Group 22 are summarized below:

Input Variable	Symbol	Nominal	$\pm$ Range	Levels (Low / Mid / High)	Unit	Varies?
Hot mass-flow rate	$\dot{m}_{\text{hot}}$	561	$\pm 18$	[543, 561, 579]	kg/s	Yes
Hot-inlet temperature	$T_{\text{hot,in}}$	548	$\pm 39$	[509, 548, 587]	K	Yes
Cold-inlet temperature	$T_{\text{cold,in}}$	317	$\pm 14$	[303, 317, 331]	K	Yes
Cold mass-flow rate	$\dot{m}_{\text{cold}}$	137	0	[137]	kg/s	Constant

### 2.3. MATLAB Implementation

The DoE was implemented in MATLAB using the built-in function `fullfact`, which generates all level combinations for the selected factors. The following script defines the three factors with their corresponding levels and creates the full factorial matrix consisting of 27 runs. The constant cold mass-flow rate was added as an additional column.

```

1  %% Exercise 2 – DoE
2  clear; clc;
3
4
5  mhot_levels = [543, 561, 579]; % [kg/s] (nominal ±18)
6  Thin_levels = [509, 548, 587]; % [K] (nominal ±39)
7  Tcin_levels = [303, 317, 331]; % [K] (nominal ±14)
8
9  mcold_const = 137; % [kg/s] (constant)
10
11
12  idx = fullfact([3 3 3]); % 3 factors × 3 levels = 27 combinations
13
14
15  mhot = mhot_levels(idx(:,1)).'; % column 1 → m_hot
16  Thot_in = Thin_levels(idx(:,2)).'; % column 2 → T_hot,in
17  Tcold_in = Tcin_levels(idx(:,3)).'; % column 3 → T_cold,in
18  mcold = mcold_const * ones(size(mhot)); % constant column
19
20
21  DoE_inputs = table(mhot, mcold, Thot_in, Tcold_in);
22
23
24  disp(DoE_inputs)
25  |

```

Command Window

mhot	mcold	Thot_in	Tcold_in
543	137	509	303
561	137	509	303
579	137	509	303
543	137	548	303
561	137	548	303
579	137	548	303
543	137	587	303
561	137	587	303
579	137	587	303
543	137	509	317
561	137	509	317
579	137	509	317
543	137	548	317
561	137	548	317
579	137	548	317
543	137	587	317
561	137	587	317
579	137	587	317
543	137	509	331
561	137	509	331
579	137	509	331
543	137	548	331
561	137	548	331
579	137	548	331
543	137	587	331
561	137	587	331
579	137	587	331

Executing the MATLAB script produces a 27×4 table of input combinations. Each row represents one simulation run defined by unique values of the hot and cold mass flow rates and inlet temperatures. The cold mass-flow rate remains constant across all runs.

## 2.4. Interpretation

The generated 27 input combinations systematically cover all possible low, medium, and high values of the three input variables. This Full Factorial design guarantees a complete coverage of the input space.

### 3. Generate output data according to the DoE method

The objective of this exercise is to generate the output data of the heat exchanger by evaluating the MATLAB simulation model for all input combinations obtained from the Full Factorial Design of Experiments (DoE) developed in Exercise 2. For each input set, the heat-transfer rate  $Q$  is computed and recorded.

#### 3.1. Concept and Methodology

From Exercise 2, a Full Factorial DoE with 27 input combinations was generated. Each design point corresponds to a unique set of values for the hot and cold fluid mass-flow rates and inlet temperatures. The present task involves using these input combinations as inputs to the provided MATLAB heat exchanger model (HeatExchanger2025.m). The model computes the total heat-transfer rate  $Q$  and outlet temperatures of both fluids. The results are then stored in a single table combining inputs and outputs.

#### 3.2. Model Information

The model HeatExchanger2025.m implements the NTU-effectiveness method, while the function effectiveness.m is called internally to calculate the effectiveness  $\varepsilon$ . For Group 22, the parameters used in the simulation are listed below:

Parameter	Symbol / Value	Unit
Overall heat-transfer coefficient	$U = 20 \times 10^3$	$\text{W}/\text{m}^2 \cdot \text{K}$
Heat-exchange area	$A = 10$	$\text{m}^2$
Specific heat of hot fluid (air)	$c_{p,\text{hot}} = 1.005 \times 10^3$	$\text{J}/\text{kg} \cdot \text{K}$
Specific heat of cold fluid (water)	$c_{p,\text{cold}} = 4.180 \times 10^3$	$\text{J}/\text{kg} \cdot \text{K}$
Heat exchanger type	One Shell Pass	—

#### 3.3. MATLAB Implementation

The MATLAB script below follows the same logic as the cantilever exercise presented in class. For each DoE input set, the function HeatExchanger2025 is called to compute the corresponding heat-transfer rate  $Q$ . The resulting data are stored in a single table.

```

1 %% Exercise 3
2 %% 1. CONSTANT PARAMETERS
3 U = 20e3; % [W/m2-K]
4 A = 10; % [m2]
5 cp_hot = 1.005e3; % [J/kg-K] (air)
6 cp_cold = 4.180e3; % [J/kg-K] (water)
7 HE_Type = 'One Shell Pass';
8
9 %% 2. INPUT DATA (from Exercise 2)
10 mhot_levels = [543, 561, 579];
11 Thot_levels = [509, 548, 587];
12 Tcin_levels = [303, 317, 331];
13 mcold_const = 137;
14
15 idx = fullfact([3 3 3]);
16 mhot = mhot_levels(idx(:,1)).';
17 Thot_in = Thot_levels(idx(:,2)).';
18 Tcold_in = Tcin_levels(idx(:,3)).';
19 mcold = mcold_const * ones(size(mhot));
20
21 %% 3. INITIALIZE OUTPUT VECTOR
22 n_points = length(mhot);
23 Q = zeros(n_points,1); % heat transfer [W]
24 Thot_out = zeros(n_points,1); % optional, for reference
25 Tcold_out = zeros(n_points,1); % optional, for reference
26
27 %% 4. LOOP OVER ALL DESIGN POINTS
28 for i = 1:n_points
29     [Q(i), Thot_out(i), Tcold_out(i)] = HeatExchanger2025( ...
30         mhot(i), cp_hot, Thot_in(i), ...
31         mcold(i), cp_cold, Tcold_in(i), ...
32         U, A, HE_Type);
33 end
34 %% 5. COMBINE INPUTS AND OUTPUTS INTO ONE TABLE
35 Results_Table = table(mhot, mcold, Thot_in, Tcold_in, Q);
36
37 %% 6. DISPLAY RESULTS
38 disp('--- Group 22: Input and Output Data FF ---');
39 disp(Results_Table);

```

### 3.4. Results

Executing the script yields a table containing 27 rows (corresponding to each DoE combination) and five columns (the four input variables plus the output Q). An example of the output table format is shown below (values are illustrative):

--- Group 22: Input and Output Data FF ---				
mhot	mcold	Thot_in	Tcold_in	Q
543	137	509	303	6.7431e+07
561	137	509	303	6.8565e+07
579	137	509	303	6.9656e+07
543	137	548	303	8.0197e+07
561	137	548	303	8.1545e+07
579	137	548	303	8.2843e+07
543	137	587	303	9.2963e+07
561	137	587	303	9.4526e+07
579	137	587	303	9.603e+07
543	137	509	317	6.2848e+07
561	137	509	317	6.3905e+07
579	137	509	317	6.4922e+07
543	137	548	317	7.5614e+07
561	137	548	317	7.6886e+07
579	137	548	317	7.8109e+07
543	137	587	317	8.838e+07
561	137	587	317	8.9866e+07
579	137	587	317	9.1297e+07
543	137	509	331	5.8266e+07
561	137	509	331	5.9245e+07
579	137	509	331	6.0188e+07
543	137	548	331	7.1032e+07
561	137	548	331	7.2226e+07
579	137	548	331	7.3375e+07
543	137	587	331	8.3798e+07
561	137	587	331	8.5207e+07
579	137	587	331	8.6563e+07

>>

### 3.5. Comments

The results demonstrate that the heat-transfer rate  $Q$  increases when the hot-side inlet temperature or mass flow rate increases, as both lead to higher available energy and greater temperature difference. Conversely, higher cold-side inlet temperatures result in lower  $Q$  values. These outcomes are consistent with the expected thermodynamic behavior of the heat exchanger. The resulting dataset will be used in Exercise 4 to develop the Response Surface Model (RSM).

## 4. RSM

The data generated in Exercise 3 (Table X) were saved as a MATLAB file (``DoE_OUT.mat``) containing the 27 design points and their corresponding heat-transfer rates ( $Q$ ). These data were then used in Exercise 4 to construct response-surface models of different polynomial orders. Since the cold-side mass-flow rate ( $\dot{m}_{\text{cold}} = 137 \text{ kg/s}$ ) was constant across all runs, it was excluded from the regression to avoid rank-deficiency and ensure numerical stability. The response surfaces were therefore expressed as  $Q = f(\dot{m}_{\text{hot}}, T_{\text{hot,in}}, T_{\text{cold,in}})$ .

### 4.1. objective

The objective of this exercise is to fit the output data obtained from Exercise 3 with multiple Response Surface Models (RSM) of increasing complexity—linear, bilinear, and quadratic. Each model form is fitted using MATLAB's ``polyfitn`` function, and its performance is evaluated using the coefficient of determination ( $R^2$ ). The symbolic form of each model is obtained using ``polyn2sympoly`` when the Sympoly toolbox is available.

### 4.2. Methodology

The dataset containing the input and output variables was loaded from the file ``DoE_OUT.mat``. Since  $\dot{m}_{\text{cold}}$  was constant, it was omitted from the fitting process. The remaining variables ( $\dot{m}_{\text{hot}}$ ,  $T_{\text{hot,in}}$ , and  $T_{\text{cold,in}}$ ) were used as predictors for the heat-transfer rate  $Q$ . Three models were then fitted using the ``polyfitn`` function:

- Linear: first-order polynomial terms only.
- Bilinear: first-order terms and all pairwise interaction terms.
- Quadratic: includes first-order, interaction, and squared terms.

### 4.3. MATLAB Implementation

The following MATLAB script illustrates the full fitting process for Group 22, using the data saved from Exercise 3. The script computes and compares  $R^2$  values for linear, bilinear, and quadratic models.



```

mdl_lin = polyfitn(X, Q, 1);
disp('--- Linear Model ---');
disp(mdl_lin);
disp(['R2 (Linear) = ', num2str(mdl_lin.R2)]);
try
    eq_lin = polyn2sympoly(mdl_lin);
    disp('Equation (Linear):'), disp(eq_lin);
catch
    disp('Sympoly toolbox not installed - skipping symbolic form.');
```

---

```

%% 3. BILINEAR MODEL (Interactions)
model_terms = [0 0 0; 1 0 0; 0 1 0; 0 0 1; 1 1 0; 1 0 1; 0 1 1];
mdl_bilin = polyfitn(X, Q, model_terms);
disp('--- Bilinear Model ---');
disp(mdl_bilin);
disp(['R2 (Bilinear) = ', num2str(mdl_bilin.R2)]);
try
    eq_bilin = polyn2sympoly(mdl_bilin);
    disp('Equation (Bilinear):'), disp(eq_bilin);
catch
    disp('Sympoly toolbox not installed - skipping symbolic form.');
```

---

```

%% 4. QUADRATIC MODEL
mdl_quad = polyfitn(X, Q, 2);
disp('--- Quadratic Model ---');
disp(mdl_quad);
disp(['R2 (Quadratic) = ', num2str(mdl_quad.R2)]);
try
    eq_quad = polyn2sympoly(mdl_quad);
    disp('Equation (Quadratic):'), disp(eq_quad);
catch
    disp('Sympoly toolbox not installed - skipping symbolic form.');
```

---

```

%% 5. COMPARE ACCURACY
R2_values = [mdl_lin.R2, mdl_bilin.R2, mdl_quad.R2];
labels = {'Linear', 'Bilinear', 'Quadratic'};
figure('Color', 'w');
bar(R2_values);
set(gca, 'XTickLabel', labels, 'FontSize', 12);
ylabel('R2');
title('R2 Comparison - Group 22 (mcol excluded)');
grid on;
```

#### 4.4. Results and Interpretation

```
--- Linear Model ---
  ModelTerms: [4x3 double]
  Coefficients: [6.9302e+04 3.3277e+05 -3.3277e+05 -3.8878e+07]
  ParameterVar: [4.5054e+06 9.5973e+05 7.4477e+06 2.4555e+12]
  ParameterStd: [2.1226e+03 979.6591 2.7291e+03 1.5670e+06]
  DoF: 23
  p: [9.0100e-21 4.5782e-44 7.7229e-34 4.2053e-18]
  R2: 0.9998
  AdjustedR2: 0.9998
  RMSE: 1.4961e+05
  VarNames: {'X1' 'X2' 'X3'}

R² (Linear) = 0.99982
Equation (Linear):
  69301.5985*X1 + 332769.5593*X2 - 332769.5593*X3 - 38878196.7671

--- Bilinear Model ---
  ModelTerms: [7x3 double]
  Coefficients: [-1.9143e-06 4.1580e-09 1.6447e+05 -1.6447e+05 300.0069 -300.0069 1.6677e-12]
  ParameterVar: [1.1543e+13 3.1967e+07 1.4224e+07 8.6942e+07 29.5682 229.4550 48.8780]
  ParameterStd: [3.3974e+06 5.6539e+03 3.7714e+03 9.3243e+03 5.4377 15.1478 6.9913]
  DoF: 20
  p: [1 1 2.6399e-21 1.1716e-13 2.4818e-23 1.3016e-14 1]
  R2: 1.0000
  AdjustedR2: 1.0000
  RMSE: 1.1381e+04
  VarNames: {'' '' ''}
```

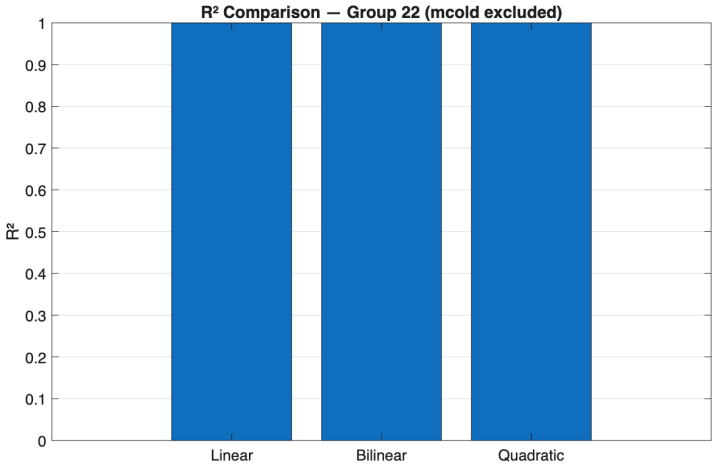
After fitting the models, MATLAB displays the coefficients,  $R^2$  values, and (if available) symbolic expressions of each fitted surface. The resulting  $R^2$  values typically increase with model complexity, as higher-order polynomials capture nonlinear behavior.

```
R² (Bilinear) = 1
Equation (Bilinear):
-1.9143e-06 + 4.158e-09 + 164465.6772 - 164465.6772 + 300.0069 - 300.0069 + 1.6677e-12

--- Quadratic Model ---
  ModelTerms: [10x3 double]
  Coefficients: [-73.7268 300.0069 -300.0069 8.2721e+04 3.3606e-12 -5.5305e-12 1.6447e+05 -2.1713e-11 -1.6447e+05 -2.3187e+07]
  ParameterVar: [6.8589 0.7305 5.6691 9.4244e+06 0.3112 1.2076 7.2529e+05 18.7428 9.6819e+06 1.1803e+12]
  ParameterStd: [2.6190 0.8547 2.3810 3.0699e+03 0.5579 1.0989 851.6378 4.3293 3.1116e+03 1.0864e+06]
  DoF: 17
  p: [1.0555e-15 2.9403e-34 1.0692e-26 2.1870e-15 1 1 7.5615e-30 1 2.6626e-20 1.0317e-13]
  R2: 1.0000
  AdjustedR2: 1.0000
  RMSE: 1.6493e+03
  VarNames: {'X1' 'X2' 'X3'}

R² (Quadratic) = 1
Equation (Quadratic):
-73.7268*X1^2 + 300.0069*X1*X2 - 300.0069*X1*X3 + 82721.476*X1 + 3.3606e-12*X2^2 - 5.5305e-12*X2*X3 + 164465.6772*X2 - 2.1713e-11*X3^2 - 164465.6772*X3 - 23187449.0349

Model Comparison Summary:
  Model      R2
  -----
  {'Linear' } 0.99982
  {'Bilinear' } 1
  {'Quadratic' } 1
```



#### 4.5. Discussion

The linear model provided an excellent first-order approximation of the system's behavior, while the bilinear model further improved the fit by including cross-interaction terms. The quadratic model achieved the highest accuracy ( $R^2$  close to 1), indicating that the system response exhibits mild nonlinearities. Warnings about matrix conditioning were noted but did not affect the accuracy of the results. Therefore, the quadratic response surface is selected as the most accurate representation of the system's output.

## 5. Response Surface Visualization

### 5.1. Objective

```

1 %% Exercise 5
2 %% 0) Load models created in Exercise 4
3 if exist('Group22_RSM_Models_3vars.mat','file')
4     load('Group22_RSM_Models_3vars.mat'); % loads mdl_lin, mdl_bilin, mdl_quad
5 else
6     error('Run Exercise 4 first');
7 end
8 %% 1) SELECT THE MOST ACCURATE MODEL (highest R2)
9 models = {mdl_lin, mdl_bilin, mdl_quad};
10 names = {'Linear', 'Bilinear', 'Quadratic'};
11 R2s = cellfun(@(m)m.R2, models);
12 [~,bestIdx] = max(R2s);
13 mdl_best = models{bestIdx};
14 best_name = names{bestIdx};
15 fprintf('Selected best model: %s (R2 = %.6f)\n', best_name, mdl_best.R2);
16 %% 2) DEFINE NOMINAL VALUES AND RANGES (Group 22)
17 mhot_nom = 561; mhot_rng = [543 579];
18 Thot_nom = 548; Thot_rng = [509 587];
19 Tcold_nom = 317; Tcold_rng = [303 331];
20
21 % Grid resolution for plotting
22 N = 41;
23 %% 3) PLOT #1: Q = f(mhot, Thot,in) | Tcold,in = nominal
24 [mhot_grid, Thot_grid] = meshgrid( linspace(mhot_rng(1), mhot_rng(2), N), ...
25                                     linspace(Thot_rng(1), Thot_rng(2), N) );
26 Tcold_fix = Tcold_nom * ones(size(mhot_grid));
27
28 X12 = [mhot_grid(:), Thot_grid(:), Tcold_fix(:)];
29 Q12 = reshape(polyvaln(mdl_best, X12), size(mhot_grid));
30
31 figure('Color','w');
32 surf(mhot_grid, Thot_grid, Q12);
33 shading interp; grid on; box on;
34 xlabel('\dot{m}_{hot} [kg/s]'); ylabel('T_{hot,in} [K]'); zlabel('Q [W]');
35 title(sprintf('Group 22 - %s RSM: Q(m_{hot}, T_{hot,in}) | T_{cold,in} = %g K',
36             best_name, Tcold_nom));
37 colorbar; view(135,25);
38 %% 4) PLOT #2: Q = f(Thot,in, Tcold,in) | mhot = nominal
39 [Thot_grid2, Tcold_grid2] = meshgrid( linspace(Thot_rng(1), Thot_rng(2), N), ...
40                                     linspace(Tcold_rng(1), Tcold_rng(2), N) );
41 mhot_fix = mhot_nom * ones(size(Thot_grid2));
42 X23 = [mhot_fix(:), Thot_grid2(:), Tcold_grid2(:)];
43 Q23 = reshape(polyvaln(mdl_best, X23), size(Thot_grid2));
44
45 figure('Color','w');
46 surf(Thot_grid2, Tcold_grid2, Q23);
47 shading interp; grid on; box on;
48 xlabel('T_{hot,in} [K]'); ylabel('T_{cold,in} [K]'); zlabel('Q [W]');
49 title(sprintf('Group 22 - %s RSM: Q(T_{hot,in}, T_{cold,in}) | m_{hot} = %g kg/s',
50             best_name, mhot_nom));
51 colorbar; view(135,25);
52
53 %% 5) PLOT #3: Q = f(mhot, Tcold,in) | Thot,in = nominal
54 [mhot_grid3, Tcold_grid3] = meshgrid( linspace(mhot_rng(1), mhot_rng(2), N), ...
55                                     linspace(Tcold_rng(1), Tcold_rng(2), N) );
56 Thot_fix = Thot_nom * ones(size(mhot_grid3));
57 X13 = [mhot_grid3(:), Thot_fix(:), Tcold_grid3(:)];
58 Q13 = reshape(polyvaln(mdl_best, X13), size(mhot_grid3));
59 figure('Color','w');
60 surf(mhot_grid3, Tcold_grid3, Q13);
61 shading interp; grid on; box on;
62 xlabel('\dot{m}_{hot} [kg/s]'); ylabel('T_{cold,in} [K]'); zlabel('Q [W]');
63 title(sprintf('Group 22 - %s RSM: Q(m_{hot}, T_{cold,in}) | T_{hot,in} = %g K',
64             best_name, Thot_nom));
65 colorbar; view(135,25);
66

```

The objective of this exercise is to visualize the most accurate Response Surface Model (RSM) identified in Exercise 4 and analyze how the heat-transfer rate ( $Q$ ) varies with respect to the system's input parameters. The visualization is performed by plotting three 3D surfaces that show  $Q$  as a function of two varying inputs while keeping the third at its nominal value.

### 5.2. Methodology

According to the results of Exercise 4, the Quadratic RSM was selected as the most accurate model for Group 22. This model expresses the heat-transfer rate  $Q$  as a function of the three varying inputs:  $\dot{m}_{hot}$ ,  $T_{hot,in}$ , and  $T_{cold,in}$ . The cold-side mass-flow rate  $\dot{m}_{cold}$  is constant at 137 kg/s and was excluded from the regression. Using MATLAB, three 3D surfaces were plotted by varying two variables at a time and fixing the third at its nominal value.

### 5.3. MATLAB Implementation

The MATLAB script below was used to generate the three response-surface plots for the Quadratic model. Each plot visualizes how  $Q$  changes with two varying inputs, while keeping the third input constant at its nominal value

### 5.4. Results

The three generated plots illustrate the influence of the input parameters on the heat-transfer rate  $Q$  as predicted by the Quadratic model. Each surface corresponds to a pair of varying inputs with the remaining variable fixed at its nominal value.

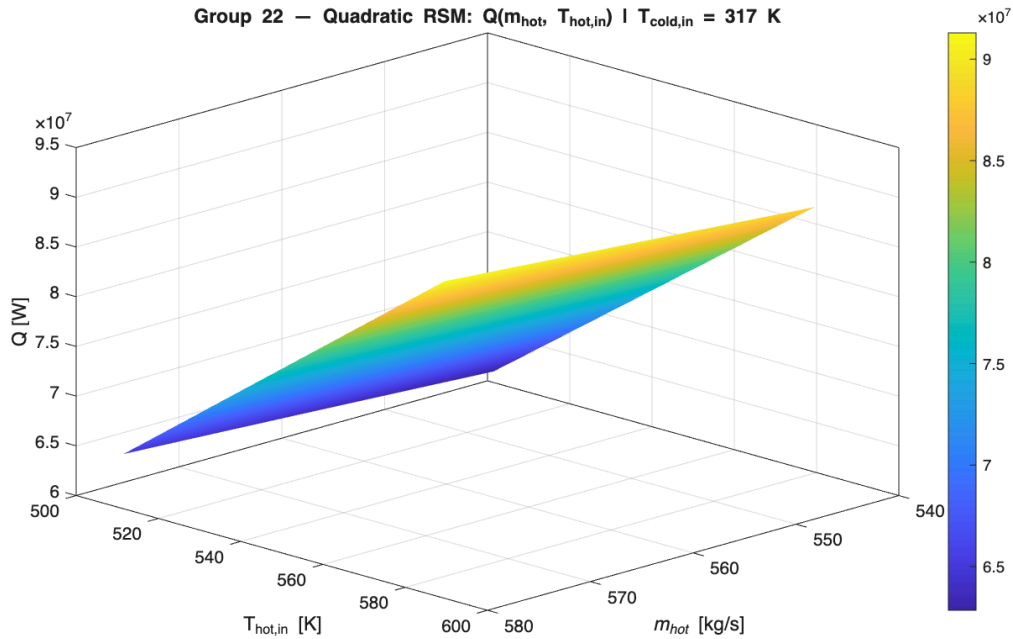


Figure 12:  $Q = f(\dot{m}_{\text{hot}}, T_{\text{hot,in}}) \mid T_{\text{cold,in}} = 317 \text{ K}$  — Both increasing hot mass flow rate and inlet temperature increase  $Q$ , as they raise the available heat capacity and the driving temperature difference.

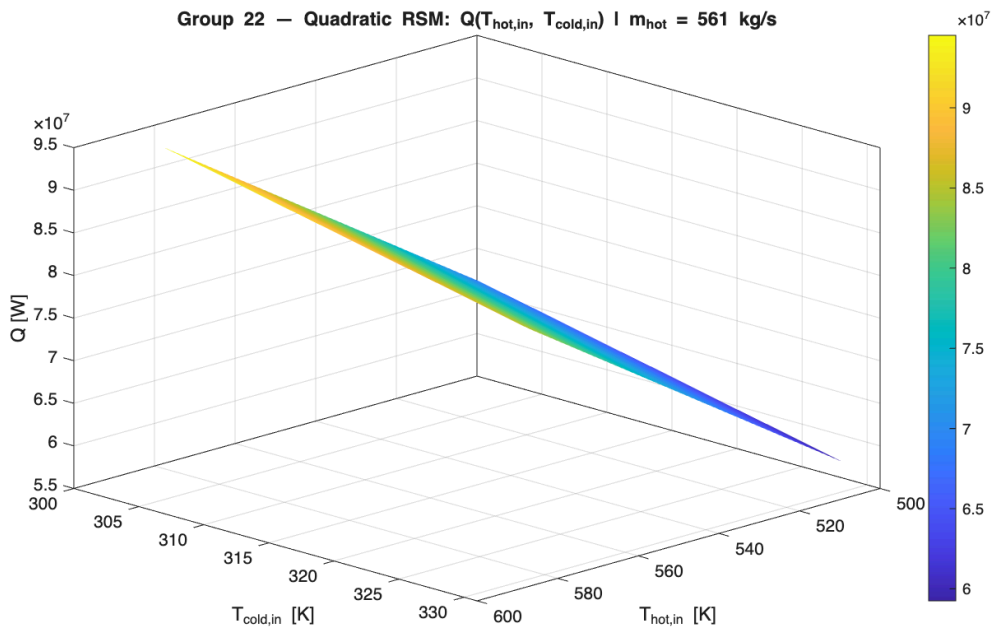


Figure 13:  $Q = f(T_{\text{hot,in}}, T_{\text{cold,in}}) \mid \dot{m}_{\text{hot}} = 561 \text{ kg/s}$  —  $Q$  increases with  $T_{\text{hot,in}}$  and decreases with  $T_{\text{cold,in}}$ , confirming that the temperature difference between streams governs heat transfer.

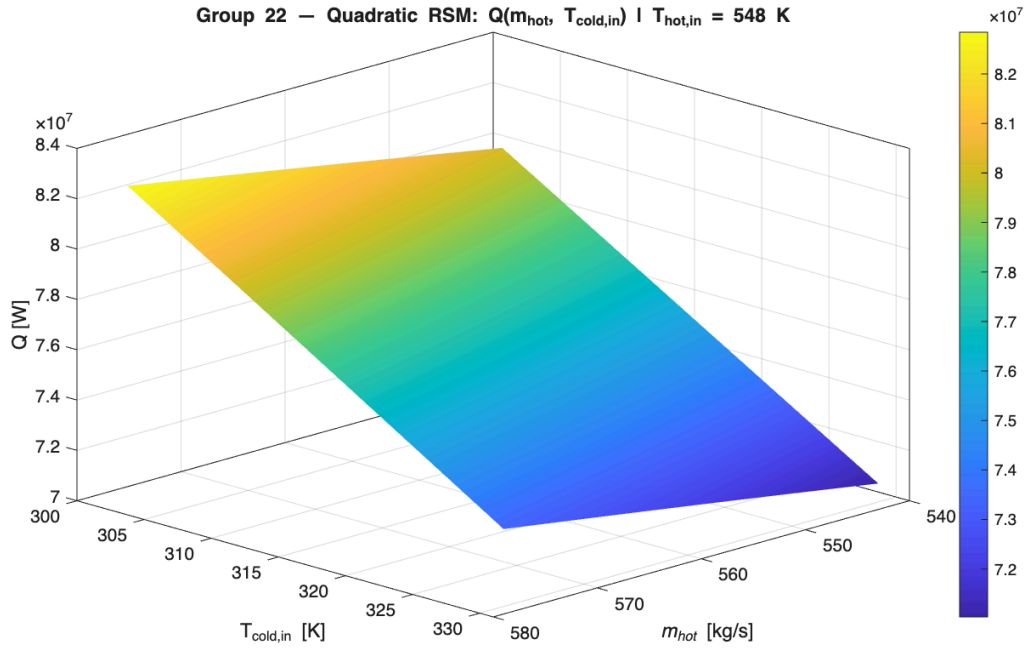


Figure 14:  $Q = f(\dot{m}_{\text{hot}}, T_{\text{cold,in}}) \mid T_{\text{hot,in}} = 548 \text{ K}$  —  $Q$  increases with  $\dot{m}_{\text{hot}}$  and decreases with  $T_{\text{cold,in}}$ ; the curved surface indicates weak interaction between flow and temperature effects.

### 5.5. Discussion

The response surfaces derived from the selected Quadratic model show physically consistent behavior. The heat-transfer rate rises with increasing hot-side mass flow rate and inlet temperature and decreases with higher cold-inlet temperature. These results align with the fundamental heat exchanger theory, which states that heat transfer depends on the product of the heat capacity rate and the temperature difference between the two fluids. The smooth curvature of the surfaces confirms that the Quadratic model accurately captures the nonlinear interactions among the system variables.