TU Clausthal
Clausthal University of Technology
Institute of Chemical and Electrochemical Process Engineering

Project Work
Summer Term 2025

The aim of the project work is to improve student's understanding of the basic methods and algorithms discussed in the lecture and to apply these methods. The project is designed as an individual work and has to be submitted via **Stud.IP** by **10.08.2025** (**23:59:59**). Please submit your work as a **.zip file** containing the following:

- **Well-commented source code** that clearly explains your implementation.
- A **well-structured PDF document** in which you **answer all given questions**. If helpful, you may include **graphs or visualizations** in the PDF to support your discussion of the results. If you are unable to fully solve a programming task, please describe your approach, thought process, and partial solutions in detail. In general, your PDF should include a **critical discussion** of your results**, including any limitations**, **unexpected outcomes**, or **insights gained** during your work.
- If you used any **AI tool (e.g., ChatGPT)** during your work, please **include the transcript or link of the conversation** as well, as done in previous submissions.

Passing the project work is a **necessary to get the Bonus Points for the exam**. At least **42 points** out of 84 points are required to pass the project work.

-------------------------------------- Theory Questions --------------------------------------

Task 1 – Multiple Choice Questions (8P)

There are always 4 possible answers in the questions below. The correct answers (minimum one, maximum four) should be marked with a cross, and then each correct cross counts for 0.5 points. Answers that are not correct should not be marked, and also count for 0.5 points each. If no answer is marked, the corresponding task is considered as not completed!

a) Which of these conditions are sufficient for a strict local minimum x* of a twice continuously differentiable function F? (2P)

- ○ $F(x^* + \delta x) - F(x^*) > 0 \qquad \forall \delta x \in \mathbb{R}^n : \|\delta x\| < \varepsilon$
- [✓ ○ $F_{xx}(x^*)$ is pos. definit and $F_x(x^*) = 0$]
- ○ $F_{xx}(x^*) > 0$
- ○ $F_x^T(x^*) \cdot \delta x < 0 \qquad \forall \delta x \in \mathbb{R}^n : \|\delta x\| < \varepsilon$

b) Which statements about quadratic penalty functions are correct? (2P)

o The optimum of the surrogate function is always within the permissible range.

[✓ o The position of the optimum of the replacement function depends on the weighting factor.]

o The existence of the optimum of the replacement function is always guaranteed.

[✓ o Quadratic penalty functions can be applied to equality and inequality constraints.]

c) Which of these problems are clearly convex ones? (2P)

[✓ o Unconstrained problems with convex objective functions.]

o Problems with convex constraints.

o Constrained problems with convex objective functions.

o Quadratic problems.

d) Which of these steps are part of the active set algorithm? (2P)

[✓ o Checking the KKT conditions.]

o Decomposition of the variable vector $x^{(k)}$ into dependent and independent variables.

o The current variable vector $x^{(k)}$ is changed by means of random numbers $\delta x^{(k)}$.

[✓ o Determining the permissible search direction $v^{(k)}$.]

Task 2 – Gradient Methods (10P)

a) Which condition must be fulfilled by a search direction $v^{(k)}$? Also formulate this condition mathematically and proof that this is always fulfilled by the steepest descent method! (4P)

b) Formulate the subproblem that must be solved in each iteration and give two methods which can be used for that. Can the simplex method according to Nelder-Mead be used for this? Give a short explanation! (6P)

Task 3 – Newton Methods (10P)

a) Give two equivalent interpretations of the basic idea of the Newton-Raphson method, as well as its iteration rule! (4P)

b) List three major difficulties that can occur when using the Newton-Raphson method and explain how the quasi-Newton method addresses these difficulties! (3P)

c) Compare the quasi-Newton method with the steepest descent method using the criteria convergence speed (number of iterations), computational effort per iteration and robustness! (3 P)

------------------------------------ Programming Part ------------------------------------

In the programming part, you have the option to choose between two different tasks: **Task 4 <u>or</u> Task 5**. While Task 4 is a real-life chemical engineering example, Task 5 considers a more general problem and does not require any specific background knowledge. Both tasks yield the same number of points (56P), so **only one needs to be completed.**

Task 4 – Optimal operation of a methanation reactor with subsequent feeding into the natural gas grid (56P)

In order to guarantee a certain methane concentration, which is required for feeding into the natural gas grid, the product gas from a biogas plant must be further processed in a continuously operated isothermal ideal tubular reactor. The methanation reaction that takes place in the reactor

$$\mathrm{CO_2} + 4\,\mathrm{H_2} \leftrightarrow \mathrm{CH_4} + 2\,\mathrm{H_2O},$$

converts excess carbon dioxide, an essential component of biogas, and hydrogen (produced separately by water electrolysis) into methane and water. The reaction rate r of this reaction can be calculated with the following equation:

$$r = k * p_{\mathrm{CO_2}}^{0.5} * p_{\mathrm{H_2}}^{0.5} * \left(1 - \frac{p_{\mathrm{CH_4}} * p_{\mathrm{H_2O}}^{2}}{p_{\mathrm{CO_2}} * p_{\mathrm{H_2}}^{4} * K_{\mathrm{eq}}}\right) * \frac{1}{Q^2} \qquad [r] = \mathrm{mol/(g_{cat}\ s)}$$

Here the variables p_i represent the partial pressures (in bar) of the corresponding species i (i = CH_4, H_2O, H_2, CO_2), k is the (temperature-dependent) rate constant

$$k = 3.46 * 10^{-4} * exp\left(\frac{77500}{R} * \left(\frac{1}{T_{\mathrm{ref}}} - \frac{1}{T}\right)\right) \qquad [k] = \mathrm{mol/(g_{cat}\ s\ bar)}$$

and Q describes the dependence of the reaction rate r on the adsorption of the species on the catalyst surface.

$$Q = 1 + K_{\text{mix}} * p_{CO_2}^{0.5} + K_{H_2} * p_{H_2}^{0.5} + \frac{K_{OH} * p_{H_2O}}{p_{H_2}^{0.5}} \qquad [Q] = 1$$

The temperature-dependent adsorption constants can be calculated as follows:

$$K_{\text{mix}} = 0.88 * exp\left(\frac{-10000}{R}\left(\frac{1}{T_{\text{ref}}} - \frac{1}{T}\right)\right) \qquad [K_{\text{mix}}] = \text{bar}^{-0.5}$$

$$K_{H_2} = 0.44 * exp\left(\frac{-6200}{R}\left(\frac{1}{T_{\text{ref}}} - \frac{1}{T}\right)\right) \qquad [K_{H_2}] = \text{bar}^{-0.5}$$

$$K_{OH} = 0.5 * exp\left(\frac{22400}{R}\left(\frac{1}{T_{\text{ref}}} - \frac{1}{T}\right)\right) \qquad [K_{OH}] = \text{bar}^{-0.5}$$

The temperature dependence of the equilibrium constant K_{eq} can be described by the following approach:

$$K_{\text{eq}} = 137 * T^{-3.998} * exp\left(\frac{158700}{R*T}\right) * 1.01325^{-2} \qquad [K_{\text{eq}}] = \text{bar}^{-2}$$

Knowing the reaction rate r and the gas velocity v (in m/s), the concentration profiles (c in mol/m^3) over the axial length of the reactor (coordinate x in m), can be calculated using the following ODE system:

$$\frac{dc_{CH_4}}{dx} = \frac{0.1}{v} * r * \rho_{\text{cat}} * (1 - \varepsilon)$$

$$\frac{dc_{H_2O}}{dx} = 2 * \frac{0.1}{v} * r * \rho_{\text{cat}} * (1 - \varepsilon)$$

$$\frac{dc_{H_2}}{dx} = -4 * \frac{0.1}{v} * r * \rho_{\text{cat}} * (1 - \varepsilon)$$

$$\frac{dc_{CO_2}}{dx} = \frac{-0.1}{v} * r * \rho_{\text{cat}} * (1 - \varepsilon)$$

To obtain the partial pressures (in bar) for the calculation of the reaction rate, the following relationship applies:

$$p_i = \frac{c_i}{\sum_{i=1}^{4} c_i} * p \qquad [p] = \text{bar}$$

<u>Necessary Model - Parameter:</u>

$R = 8.314\,\text{J/mol}$	$\varepsilon = 0.4$	$T_{\text{ref}} = 555\text{ K}$
$\rho_{\text{cat}} = 1.8 \cdot 10^6\,\text{g/m}^3$	$p = 15$ bar	

a) **Calculate** the axial concentration profiles resulting from the given ODE system for a methanation reactor with a length of 1.5 m. The reactor is operated at a constant temperature of 523 K and a gas velocity of 1 m/s. Use the following initial conditions for solving the problem:

$$c_{\mathrm{CH_4}}(x=0) = 0\,\mathrm{mol/m^3}$$
$$c_{\mathrm{H_2O}}(x=0) = 0\,\mathrm{mol/m^3}$$
$$c_{\mathrm{H_2}}(x=0) = 10\,\mathrm{mol/m^3}$$
$$c_{\mathrm{CO_2}}(x=0) = 2.5\,\mathrm{mol/m^3}$$

Plot the **concentration profiles** and **determine** the concentration of carbon dioxide (CO_2) at the reactor outlet! (22P)

b) For feeding into the natural gas grid, the carbon dioxide content of the product gas ($x_{CO_2} = \frac{c_{CO_2}}{\sum_{i=1}^{4} c_i}$) should not exceed 1%. To meet this requirement, both the reactor temperature T and the gas velocity v can be modified. In order to guarantee a high productivity of the reactor, the methane throughput

$\dot{n}_{\mathrm{CH_4}} = c_{\mathrm{CH_4}} * v$ $[\dot{n}]$ = mol/s

is to be **maximized** simultaneously. **Formulate** the complete **optimization problem** (give the mathematical formulation: objective function, optimization variables, constraints, dimensions), **calculate the optimal values for T and v**, **check** the carbon dioxide content of the product gas x_{CO_2} and **give the maximum methane flow rate**. You can use your own or MATLAB/Python/CasADi built-in algorithms (do not use search methods) for solving the optimization problem. The starting point for the optimization should be the solution from Task 4 a). Finally, **validate your results** by creating a **2D contour plot**, with temperature T and gas velocity v on the axes, and showing different levels of the objective function (methane flow rate) as contour lines. (24P)

c) Which other, technically suitable parameter would you use to achieve an even higher methane throughput? **Briefly explain your choice** and **determine** the result with **another optimization calculation**. Compare your result to the solution of task b) and discuss it critically. (10P)

Task 5 – Optimal packing of 11 squares (56P)

11 identical **small squares**, each with a **side length of 1**, should be placed **inside one larger square** with the **side length b**. Your task is to find the **smallest side length b** that fits all 11 squares. The small squares can be rotated arbitrarily and can touch each other, but they should not overlap. To give you an idea of how a possible solution to this problem might look like: The figure below shows the optimal packing of 10 small squares.

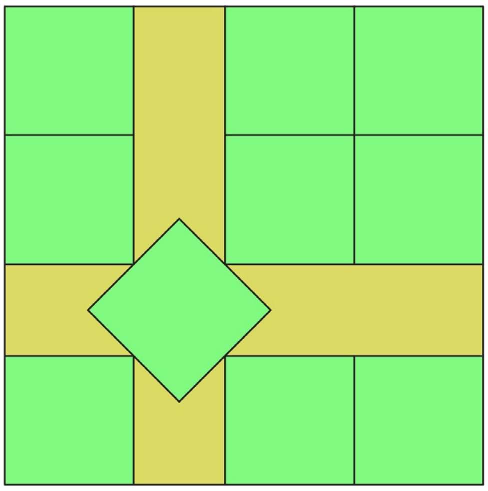
Figure 1: Example - Optimal packing of 10 small squares.

a) **Formulate** the corresponding optimization problem. Especially think about a suitable formulation of the constraints (give the mathematical formulation: objective function, optimization variables, constraints, dimensions). (12P)

b) **Perform** a mathematical optimization to **minimize the side length b** using MATLAB, Python, or CasADi. Clearly **comment** your code and **describe your approach** in a **step-by-step** manner (As a small hint: Consider starting with a simple script, such as placing the 11 small squares without overlapping. Then create the larger square that fits the smaller ones exactly). **Choose a optimization algorithm** for this problem and **discuss** your choice. Finally, **plot** your solution / your arrangement of the squares and **interpret** your result. (28P)

c) Do you expect that your solution of task b) is a **local or a global** optimum? **Perform a global optimization** to proof your expectation. (16P)

Contact
Lukas Gottheil, M.Sc.
gottheil@icvt.tu-clausthal.de

Submission per .zip file on Stud.IP
Deadline: 10.08.2025 (23:59:59)

Task 2 – Gradient Methods (10P)

a) Which condition must be fulfilled by a search direction $v^{(k)}$? Also formulate this condition mathematically and proof that this is always fulfilled by the steepest descent method! (4P)

b) Formulate the subproblem that must be solved in each iteration and give two methods which can be used for that. Can the simplex method according to Nelder-Mead be used for this? Give a short explanation! (6P)

a) The descent condition must be fulfilled by a search direction $v^{(k)}$.

$$(v^{(k)})^T F_x^{(k)} < 0$$

$\rightsquigarrow$ Steepest descent method: $v^{(k)} = -F_x^{(k)}$

$$(v^{(k)})^T F_x^{(k)} = (-F_x^{(k)})^T F_x^{(k)} = -\|F_x^{(k)}\| < 0 .$$

b) The subproblem that must be solved at each iteration is given by

$$\min_{\alpha > 0} F(x^{(k)} + \alpha^{(k)} v^{(k)})$$

where $x^{(k+1)} = x^{(k)} + \alpha^{(k)} v^{(k)}$

The subproblem consists of an unconstrained one dimensional optimization problem to determine stepsize $\alpha^{(k)}$ at each iteration.

To solve this problem we could use search methods such as the golden ratio search, with multiple evaluations of the function F. Furthermore, we could use the line search method with the wolfe conditions to provide sets of allowed step sizes.

In principle, the Nelder-Mead simplex method could be used for this. But since this method is designed for multidimensional problems it is considered slow and does not guarantee the wolfe conditions. So, it is not recommended for this kind of problems.

Task 3 – Newton Methods (10P)

a) Give two equivalent interpretations of the basic idea of the Newton-Raphson method, as well as its iteration rule! (4P)

b) List three major difficulties that can occur when using the Newton-Raphson method and explain how the quasi-Newton method addresses these difficulties! (3P)

c) Compare the quasi-Newton method with the steepest descent method using the criteria convergence speed (number of iterations), computational effort per iteration and robustness! (3 P)

a) The main idea of Newton-Raphson's method is to apply Newton's method to the optimality condition $F_x(x^*) = 0$.

(i) 1st solution

Linear approximation of $F_x(x^{(k)})$ at the current point $x^{(k)}$.

$$\tilde{F}_x(x^{(k)} + \Delta x^{(k)}) = F_x(x^{(k)}) + F_{xx}(x^{(k)})\,\Delta x^{(k)}$$

Finding zero of $\tilde{F}_x$ instead of F_x

$$\tilde{F}_x = 0 = F_x(x^{(k)}) + F_{xx}(x^{(k)})\,\Delta x^{(k)}$$

$$\Rightarrow \Delta x^{(k)} = -\left[F_{xx}(x^{(k)})\right]^{-1} F_x(x^{(k)})$$

(ii) 2nd solution

Approximation of the objective function as a quadratic function.

$$\tilde{F}(x^{(k)} + \Delta x^{(k)}) = F(x^{(k)}) + (\Delta x^{(k)})^T \cdot F_x(x^{(k)}) + \frac{1}{2} (\Delta x^{(k)})^T \cdot F_{xx}(x^{(k)}) \cdot (\Delta x^{(k)})$$

Then the optimality condition of approximation becomes:

$$\tilde{F}_{\Delta x}(\Delta x^{(k)}) = F_x(x^{(k)}) + F_{xx}(x^{(k)}) \cdot \Delta x^{(k)} = 0$$

b) Difficulties of Newton-Raphson

(i) Calculation of the Hessian matrix is very expensive for high-dimensional problems.

(ii) Inverting the Hessian matrix is also very expensive.

(iii) If the Hessian matrix is not positive definite, the Newton direction might not be a descent direction leading to divergence.

Solution is Quasi-Newton method.

(i) Uses gradient information to build an approximation of the Hessian matrix and its inverse.

(ii) Updates Hessian matrix with rank-1 or rank-2 updates, avoiding large matrix inversions from scratch.

(iii) Maintains positive-definiteness of the approximation ensuring descent directions.

c)

	nº of iterations	computational effort	robustness
Steepest descent			
	↓	↓	↑
Quasi Newton			

Task 5 – Optimal packing of 11 squares (56P)

11 identical **small squares**, each with a **side length of 1**, should be placed **inside one larger square** with the **side length b**. Your task is to find the **smallest side length b** that fits all 11 squares. The small squares can be rotated arbitrarily and can touch each other, but they should not overlap. To give you an idea of how a possible solution to this problem might look like: The figure below shows the optimal packing of 10 small squares.

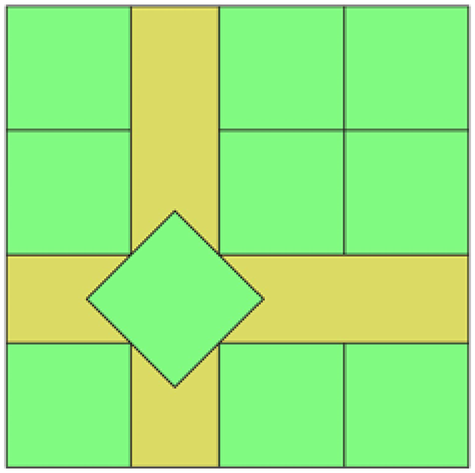

Figure 1: Example - Optimal packing of 10 small squares.

a) **Formulate** the corresponding optimization problem. Especially think about a suitable formulation of the constraints (give the mathematical formulation: objective function, optimization variables, constraints, dimensions). (12P)

b) **Perform** a mathematical optimization to **minimize the side length b** using MATLAB, Python, or CasADi. Clearly **comment** your code and **describe your approach** in a **step-by-step** manner (As a small hint: Consider starting with a simple script, such as placing the 11 small squares without overlapping. Then create the larger square that fits the smaller ones exactly). **Choose a optimization algorithm** for this problem and **discuss** your choice. Finally, **plot** your solution / your arrangement of the squares and **interpret** your result. (28P)

c) Do you expect that your solution of task b) is a **local or a global** optimum? **Perform a global optimization** to proof your expectation. (16P)

a) (i) Variables

→ $b \in \mathbb{R}$

→ For $i = 1, \dots, 11$:

↳ $c_i = (x_i, y_i) \in \mathbb{R}^2$ is the center of small square i

↳ $\theta_i \in \mathbb{R}$ is the rotation angle of small square i

Constants

(ii) The solution includes 2 steps: First, we approximate the squares as circles to get an initial guess and then close the gaps to get final solution.

$\textcircled{I}$ Circle approximation model

Let $p_{i,k} = c_i + R(\theta_i) d_k$, $k = 1, 2, 3, 4$, where $d_1 = (1/2, 1/2)$, $d_2 = (-1/2, 1/2)$, $d_3 = (-1/2, -1/2)$, $d_4 = (1/2, -1/2)$ and $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is the rotation matrix.

$$\min b$$

s.t.

$$0 \leq p_{i,k,x} \leq b \quad ; \; i = 1, \dots, 11 \; ; \; k = 1, \dots, 4$$

$$0 \leq p_{i,k,y} \leq b \quad ; \; i = 1, \dots, 11 \; ; \; k = 1, \dots, 4$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq 2 \quad ; \forall \, i < j$$

$$b \geq 1$$

$\rightsquigarrow$ Constraints 1 an 2 make sure the squares are always inside the big square. Constraint 3 is the circle approximation to avoid overlapping.

$\textcircled{\text{II}}$ Final model replacing the circle approximation

For every $i = 1, \dots, 11$, define two orthonormal edge directions $e_{i1} = (\cos\theta_i, \sin\theta_i)$ and $e_{i2} = (-\sin\theta_i, \cos\theta_i)$.

For a pair $i<j$ consider the four candidate separating axes $U_{ij} = \{e_{i1}, e_{i2}, e_{j1}, e_{j2}\}$.

For each $u \in U_{ij}$ compute:

$\rightarrow$ $\text{proj}(u) = \text{sa}((c_j - c_i)\, u)$, where $\text{sa}(t) := \sqrt{t^2 + \varepsilon}$

$\rightarrow$ $\rho_i(u) = \frac{1}{2}\left(\text{sa}(u \cdot e_{i1}) + \text{sa}(u \cdot e_{i2})\right)$

$\hookrightarrow$ $s_j(u) = \frac{1}{2}\left(sa(u \cdot e_{j1}) + sa(u \cdot e_{j2})\right)$

$\hookrightarrow$ $f_{ij}^{(u)} = proj_{ij}(u) - \left(s_i(u) + s_j(u)\right)$

Now, let

$$LSE_\alpha(\{f_K\}) = \frac{1}{\alpha} \log\left(\sum_{K=1}^{4} e^{\alpha f_K}\right)$$

Model :

$$\min b$$

s.t.

$$0 \leq p_{i,K,x} \leq b \quad ; \quad i = 1, \dots, 11 \quad ; \quad K = 1, \dots, 4$$

$$0 \leq p_{i,K,y} \leq b \quad ; \quad i = 1, \dots, 11 \quad ; \quad K = 1, \dots, 4$$

$$LSE_\alpha\left(f_{ij}^{(e_{i1})}, f_{ij}^{(e_{i2})}, f_{ij}^{(e_{j1})}, f_{ij}^{(e_{j2})} \right) \geq 0$$

$$b \geq 1$$

The added third constraint forces smoothly the squares axis together from the stage 1 solution used as initial guess.

→ Chatgpt proposed the stage 2 solution for getting the squares close after I got stuck with the stage 1 solution.

c) I expect the solution to be a local minimum because the problem is highly nonconvex and the solver tends to converge to the closest local minimum given a initial guess.