Note: these exercises are meant to provide a first introduction to Julia and present examples of the majority of the functions we will need through the course. You are **not** expected to know how to implement these, but to try to implement those yourself following the introduction. Doing so will set you to success for the upcoming assignments.

Problem 1.1: Extreme points of a univariate function

Plot the function $f(x) = 2x^4 - 5x^3 - x^2$ and characterize its stationary points.

Problem 1.2: Extreme points of a bivariate function

Plot contours of the function $f(x, y) = (y - x^2)^2 - x^2$ and characterize its stationary points.

Problem 1.3: Newton's method for a univariate problem

Consider the following unconstrained optimization problem where $f(x)$ is a univariate function:

$$\min. \quad f(x) \tag{1}$$

Solve the problem (1) with different functions $f(x)$ using Newton's method. In the univariate case, Newton's method starts with an initial starting point $x_0 \in \mathbb{R}$ and updates the solution as follows:

$$x_{n+1} = x_n - f''(x_n)^{-1} f'(x_n)$$

Try different starting points and observe if the method converges to a stationary point or diverges without producing a solution. Plot the functions $f(x)$ and show the path taken by Newton's method. You can try, for example, the following functions and starting points:

$f(x) = x^4 - x^3 - 8x^2$	try different values for x_0
$f(x) = x^5 - 6x^3 - 2x^2$	$x_0 = 1$
$f(x) = \arctan(x)$	$x_0 = 1$
$f(x) = (1/4)x^4 - x^2 + 2x$	$x_0 = 1.5$

Problem 1.4: Newton's method for a bivariate problem

Consider the following unconstrained optimization problem

$$\min. \quad (x_1 - 2)^4 + (x_1 - 2x_2)^2 \tag{2}$$

Let $f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$ denote the objective function. Solve the problem (2) using Newton's method. Newton's method starts with an initial starting point $x_0 \in \mathbb{R}^2$ and updates the solution as follows:

$$x_{n+1} = x_n - \nabla^2 f(x_n)^{-1} \nabla f(x_n)$$

Try different starting points and observe if the method converges to a stationary point or diverges without producing a solution. Plot the contours of $f(x)$ in (x_1, x_2) plane and show the path taken by Newton's method.

Problem 1.5: Pooling Problem

Consider the following example of the pooling problem presented in Figure 1. This problem arises, for example, in gas transportation and oil refinery blending problems.

$\underline{\overline{q}}_s$: value of property q at node $s \in S$
$\overline{q}_t$: upper bound of property q at node $t \in T$
c_i: unit cost at node $s \in S$ (or unit revenue at node $t \in T$)
$\overline{b}_i$: upper bound of flow to node $i \in N$

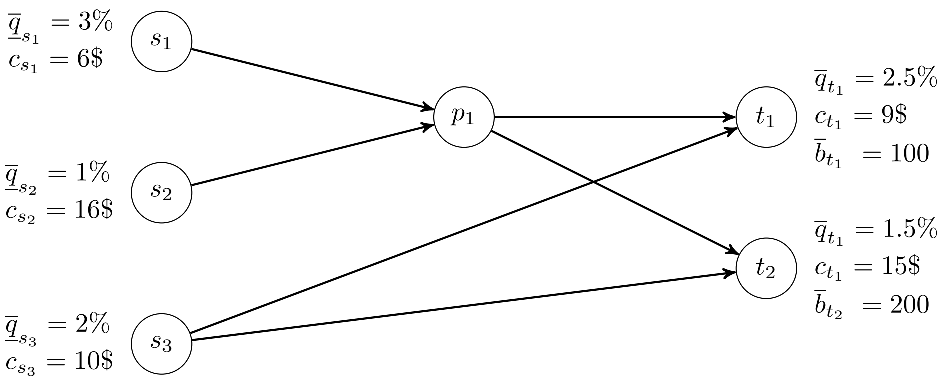

Figure 1: Pooling problem network.

The problem considered here is defined on a directed graph $G = (N, A)$ as follows. We have a set of source (or production) nodes $S = \{s_1, s_2, s_3\}$, a set of intermediate pooling nodes $P = \{p_1\}$, and a set of target (or demand) nodes $T = \{t_1, t_2\}$. The node set is thus

$$N = S \cup P \cup T$$

Each source node $s \in S$ has a unit cost c_s to purchase oil and each target node $t \in T$ has a unit value c_t which represents a revenue for receiving oil. Each node $i \in N$ has a property value q_i which corresponds to oil sulfur content in this example. These property values at source nodes $s \in S$ are constants $q_s = \underline{\overline{q_s}}$ and at target nodes $t \in T$ they have upper bounds $q_t \leq \overline{q}_t$ representing required specifications (in this case maximum sulfur content) for the oil to be commercialised. The property values q_p at pooling nodes $p \in P$ are unknown, according to the information, we can derive loose value bounds as $q_{p_1} \in [0, \infty]$, $q_{t_1} \in [0, 2.5]$, and $q_{t_2} \in [0, 1.5]$.

The arcs A represent pipelines transporting oil between the nodes. From each source node $s \in S$, crude oil with a property value q_s flows to target nodes $t \in T$ either directly or via pooling nodes $p \in P$. When two or more oil streams with different properties flow to a pooling node $p \in P$ or a target node $t \in T$, the properties q_p or q_t of the oil at that node change due to blending.

The objective is to maximise the total profit by purchasing oil at the source nodes $s \in S$ and selling it at the target nodes $t \in T$. We can use the following variables to formulate the problem:

$$\begin{aligned} x_{ij} &\geq 0 && \text{amount of oil flowing through each arc } (i,j) \in A \\ q_i &\geq 0 && \text{propery (sulfur content) at each node } i \in N \end{aligned}$$

Let us further define

$$N_i^- = \{j \in N : (j,i) \in A\} \quad \text{and} \quad N_i^+ = \{j \in N : (i,j) \in A\}$$

The problem can be formulated as follows.

$$\begin{aligned}
\max_{x,q}. \quad & \sum_{t \in T} c_t \sum_{j \in N_t^-} x_{jt} - \sum_{s \in S} c_s \sum_{j \in N_s^+} x_{sj} && && (3)\\
\text{subject to} \quad & \sum_{j \in N_p^-} x_{jp} = \sum_{j \in N_p^+} x_{pj}, && \forall p \in P && (4)\\
& \sum_{j \in N_t^-} x_{jt} \leq \overline{b}_t, && \forall t \in T && (5)\\
& \sum_{j \in N_p^-} q_j x_{jp} = q_p \sum_{j \in N_p^+} x_{pj}, && \forall p \in P && (6)\\
& \sum_{j \in N_t^-} q_j x_{jt} = q_t \sum_{j \in N_t^-} x_{jt}, && \forall t \in T && (7)\\
& q_t \leq \overline{q}_t, && \forall t \in T && (8)\\
& q_s = \underline{q}_s, && \forall s \in S && (9)\\
& q_i \geq 0, && \forall i \in P \cup T && (10)\\
& x_{ij} \geq 0, && \forall (i,j) \in A && (11)
\end{aligned}$$

The objective (3) maximises the profit given by revenue minus cost. (4) maintains flow balance, and (5) defines upper bounds of flow to target nodes. (6) and (7) determine the property values at pooling nodes and target nodes, respectively. (8) imposes upper bounds for property values at target nodes and (9) sets the initial property values at source nodes.

The problem is non-convex due to the constraints (6) and (7) which are called *bilinear*. Typically, there are more pools and more than one property q_i for each node i. This can be modeled by introducing a set K of different properties so that q_i^k denotes the value of property k at node i.

Model and solve the problem (3)-(11) with `Julia` using `JuMP` using the data shown in Figure 1. This can be done using, for example, the non-linear programming solver `Ipopt`, and by trying different initial (starting) values for the unknown property values $q_i, i \in P \cup T$ with the `JuMP` function `set_start_value(...)`.