CHEM-E7225/2023: Exercise 02

Task 1 (20 points). Consider the following uni-dimensional unconstrained optimisation problem

$$\min_{x \in \mathcal{R}} \frac{2x^2 - 4x + 11}{x^2 + 2}$$

1. Plot the objective function $f(x)$ and solve visually for the optimal value x^*;
2. Derive on paper the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ of the objective function. Can you find the values x^* for which $\nabla f(x^*) = 0$? If positive, comment on $\nabla^2 f(x^*)$;
3. What would the minimiser be, had we included inequality constraints $x \in [-20, -2]$,

$$\begin{aligned} \min_{x \in \mathcal{R}} \quad & \frac{2x^2 - 4x + 11}{x^2 + 2} \\ \text{subject to} \quad & -20 \leq x \leq -2 \end{aligned}$$

4. Implement code to formulate both these problems and then solve them for the optimal values x^*. Comment on the results of the optimisation;

   Hint: Use the code in the appendix as a template for solving these problems using CasADi's Opti Stack.

Task 2 (20 points). Consider the following two-dimensional constrained optimisation problem

$$\begin{aligned} \min_{x,y \in \mathcal{R}} \quad & \sin(y) e^{(1-\cos(x))^2} + \cos(x) e^{(1-\sin(y))^2} + (x-y)^2 \\ \text{subject to} \quad & (x+5)^2 + (y+5)^2 \leq 25 \end{aligned}$$

1. Plot the objective function $f(x, y)$ over the feasible set and solve for the optimal value (x^*, y^*);
2. Implement code to formulate this problem and then solve it for the optimal value (x^*, y^*). Show graphically and report the results when using 5 arbitrary chosen and different initial solutions. Choose one of the results and show graphically the intermediate solutions for each iteration of the solver;

   Hint: Use the matrices (`x_iter`, `y_iter`), as shown in the appendix, to plot the intermediate solutions.
3. Comment on the results of the optimisation.

Task 3 (30 points). Consider the constrained optimisation of the N-dimensional Rosenbrock function

$$\begin{aligned} \min_{x \in \mathcal{R}^{N+1}} \quad & \sum_{n=1}^{N} \left(100\left(x_{n+1} - x_n^2\right)^2 + (1 - x_n)^2\right) \\ \text{subject to} \quad & \sum_{n=1}^{N+1} (x_n - 1)^2 \leq 2 \end{aligned} \tag{1}$$

1. Implement code to formulate this problem for $N = 800$, then solve it from different initial solutions;
2. Comment on the results of the optimisation.

Task 4 (30 points). Consider the Three-Tank System from Homework 1 (Figure 1). As before, the process consists of three cylindrical tanks (T_i, $i = 1, 2, 3$) connected by two fixed valves (V_i, $i = 1, 2$), with an outflow valve V_0 for the last tank. The liquid levels (h_i, $i = 1, 2, 3$) in each tank are controlled by manipulating the incoming flow-rates to tanks T_1 and T_3 through the pumps P_1 and P_3, respectively. Here, the system is further equipped with a sensor arrangement measuring the levels of tanks T_1 and T_3, in real-time.

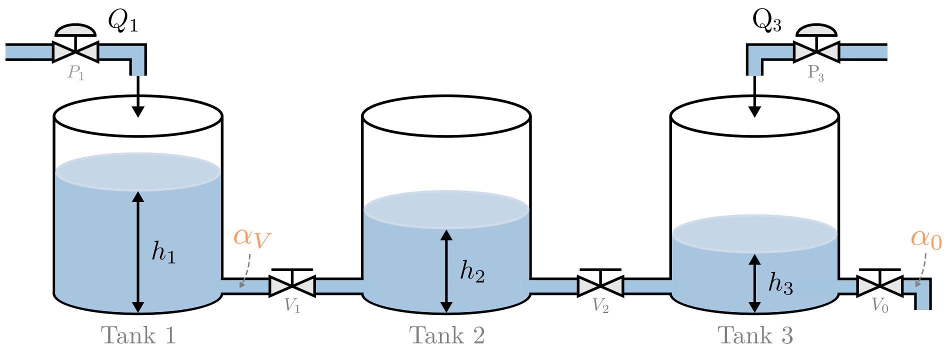

Figure 1: Three-Tank System: Process layout.

The process dynamics and measurement process are represented by the state-space model

$$\dot{x}(t) = f(x(t), u(t)|\theta_x); \tag{2a}$$
$$y(t) = g(x(t)|\theta_y), \tag{2b}$$

with state $x(t) = (h_1(t), h_2(t), h_3(t))$, controls $u(t) = \big(Q_1(t), Q_3(t)\big)$ and measurements $y(t) = (x_1(t), x_3(t))$. The state-dynamics are given by Eqs. (1a–1c) from Homework 1 with parameters $\theta_x = \big(S_T, S_V, \alpha_V, \alpha_0, g\big)$, and the measurement map simply selects $g(x(t)|\theta_y) = (x_1(t), x_2(t))$.

In this assignment we assume that the values of the flow coefficients (α_V, α_0) are not known, and thus have to be determined through *parameter fitting*. For this task, you will solve the optimisation

$$\min_{\theta_x \in \mathcal{R}^2} \quad \sum_{k=1}^{K} \left(y_k - y_k^{\text{data}}\right)^T \left(y_k - y_k^{\text{data}}\right) \tag{3a}$$
$$\text{subject to} \quad \theta_x \geq 0, \tag{3b}$$

$$\text{where} \quad x_{k+1} = F(x_k, u_k^{\text{data}}|\theta_x), \quad x_1 = (2, 8, 32), \qquad k = 1, \dots, K-1 \tag{3c}$$
$$y_k = g(x_k|\theta_y), \qquad k = 1, \dots, K \tag{3d}$$

with $\left(u^{\text{data}}, y^{\text{data}}\right)$ being some experimental input-output data obtained from the physical system. The integrator function $F(\cdot|\theta_x)$ is defined from some integration scheme (e.g., Range-Kutta 4th Order). The expressions in Eqs. (3c)–(3d) define how the vectors (x_k, y_k), for $k = 1, \dots, K$, are computed based on the decision variable $\theta_x \in \mathcal{R}_+^2$ (thus, the simulated outputs $(y_1, \dots, y_K)$ are all functions of parameters θ_x).

Adapt the starting code in the archive `E01_code.zip` for the following tasks:

1. Substitute the function files `ThreeTank.m` and `rk4fnc.m` with your solutions from Homework 1;
2. Complete the starter code `E02_main.m` to define the simulation and model parameters, the integrator function, and to build the objective function for the optimisation problem Eq. (3).
3. Execute the script to solve the problem and generate a simulation of the system using the fitted parameters. Report the simulation plots and comment on the results of the optimisation.

CasADi's Opti Stack tutorial

The **Opti Stack**[1] is a collection of helper functions from CasADi that allows us to construct nonlinear optimisation problems using the standard mathematical notation. Consider the optimization problem

$$\begin{aligned} \min_{x,y \in \mathcal{R}^N} \quad & (y - x^2)^2 \\ \text{subject to} \quad & x^2 + y^2 = 1 \\ & x + y \geq 1 \end{aligned}$$

This problem is constructed and solved through the following script:

```
opti = casadi.Opti();                    % Creates an Opti Stack structure

x = opti.variable();                     % Creates a scalar decision variable (x)
y = opti.variable();                     % Creates a scalar decision variable (y)

opti.minimize( (y - x^2)^2 );            % Define the objective function to be minimized
opti.subject_to( x^2 + y^2 == 1 )        % Defines an equality constraint
opti.subject_to( x + y >= 1 )            % Defines an inequality constraint

opti.solver('ipopt')                     % Chooses a solver (IPOPT, qpOASES, ...)

% Task 2: Callbacks to save the value of (x,y) at each iteration
x_iter = []; y_iter = [];
opti.callback(@(i) evalin('base',
        'x_iter = [x_iter opti.debug.value(x)];
         y_iter = [y_iter opti.debug.value(y)];' ))
% --

opti.set_initial([x  y], [-1 1])        % Sets an initial solution (default: 0)
opti.solve()                             % Executes the solver

x_sol = opti.value(x);                   % Retrieves the optimal solution of (x)
y_sol = opti.value(y);                   % Retrieves the optimal solution of (y)
```

The solver obtains the solution $(x^*, y^*) = (0.78615, 0.61803)$, associated to the optimal value $f(x^*, y^*) \approx 0$.

[1] Check the documentation in `https://web.casadi.org/docs/#document-opti` .