CHEM-E7225 Advanced Process Control Homework Report

Jani Arponen - 769684

May 30, 2021

This document contains all exercise solutions for the Advanced Process Control course. All code is publicly available at https://github.com/jk-arp/apc-exercises until the course is graded.

Exercise 01

Consider a perfectly mixed continuous stirred tank chemical reactor (CSTR). Mass and energy balances lead to the following nonlinear dynamics

$$\begin{aligned}
\frac{dC_A(t)}{dt} &= \frac{F_{\text{in}}(C_{A_{\text{in}}} - C_A(t))}{\pi r^2 h} - k_0 \exp\left(-\frac{E}{RT(t)}\right) C_A(t) \\
\frac{dT(t)}{dt} &= \frac{F_{\text{in}}(T_{\text{in}} - T(t))}{\pi r^2 h} + \frac{-\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT(t)}\right) + \frac{2U}{r\rho C_p}(T_c(t) - T(t)) \\
\frac{dh(t)}{dt} &= \frac{F_{\text{in}} - F(t)}{\pi r^2} \\
\mathbf{x}(t) &= \begin{bmatrix} C_A(t) & T(t) & h(t) \end{bmatrix}^T, \quad \mathbf{u}(t) = \begin{bmatrix} F(t) & T_c(t) \end{bmatrix}^T
\end{aligned} \tag{1}$$

In the given source [1] for the exercise description, the reactor's temperature state $T(t)$ equation is slightly different (eq 1C in [1]) and I have used the source version going forward instead:

$$\frac{dT(t)}{dt} = \frac{F_{\text{in}}(T_{\text{in}} - T(t))}{\pi r^2 h} + \frac{-\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT(t)}\right) C_A(t) + \frac{2U}{r\rho C_p}(T_c(t) - T(t)) \tag{2}$$

Integration schemes for simulation

The main script from the provided `dynSim.zip` was modified to use the CSTR system by writing a helper function file `CSTRfun.m`. As a first test, the system was simulated using the provided steady state values producing the graphs in Figure 1. The `ode45` solution does not seem stable and thus an additional option to reduce the relative tolerance, shown in Figure 2. The tank level stays constant as $\dot{h}(t) = 0$ with the given parameters and steady state values. Different constant controls were used to test the integrators and results are seen in the below graphs of Figure 3.

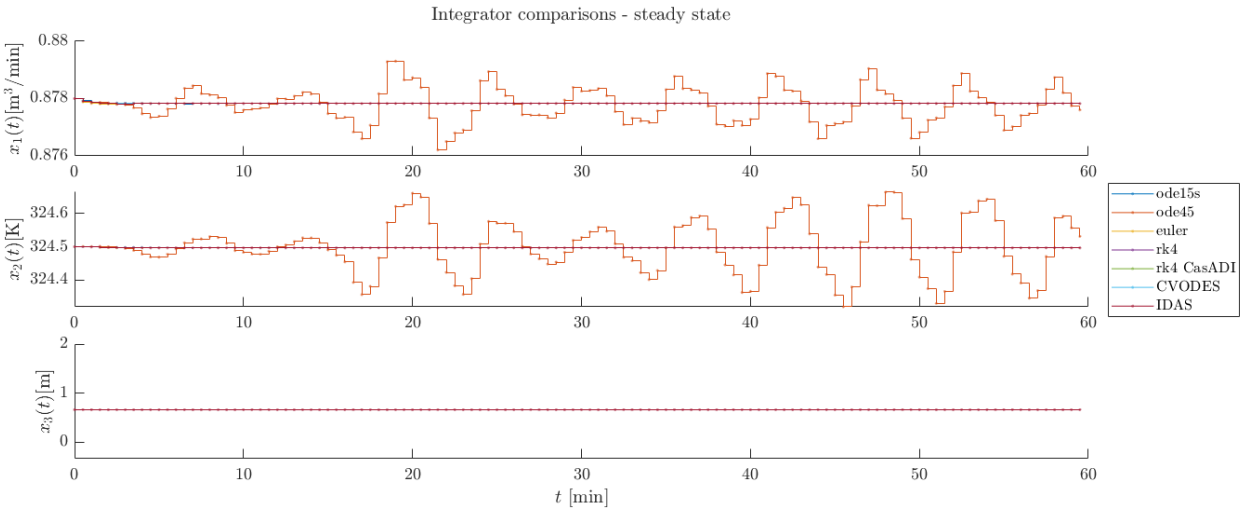


Figure 1: CSTR simulated at steady state control and state variable values.

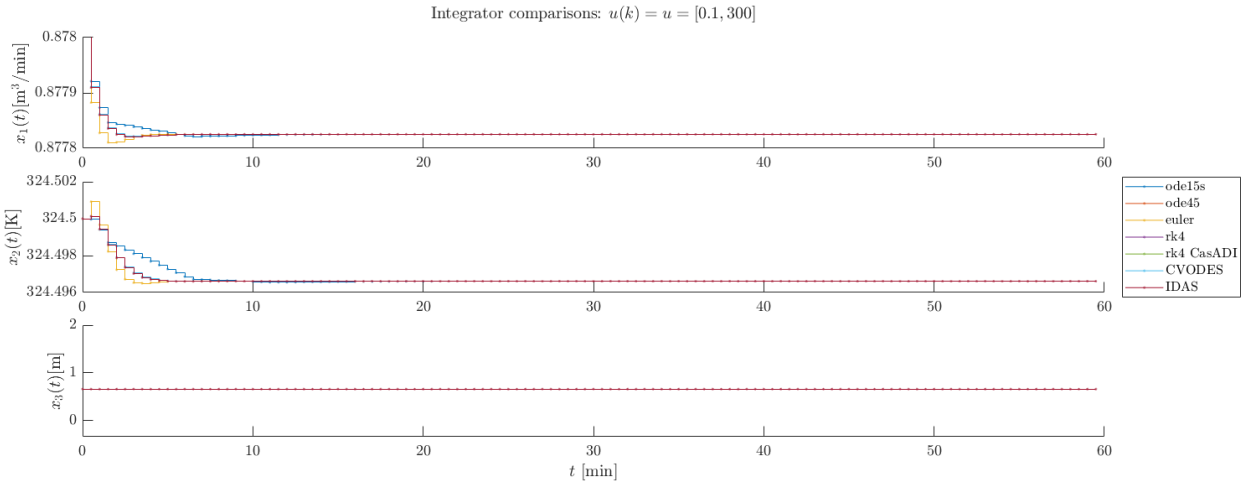
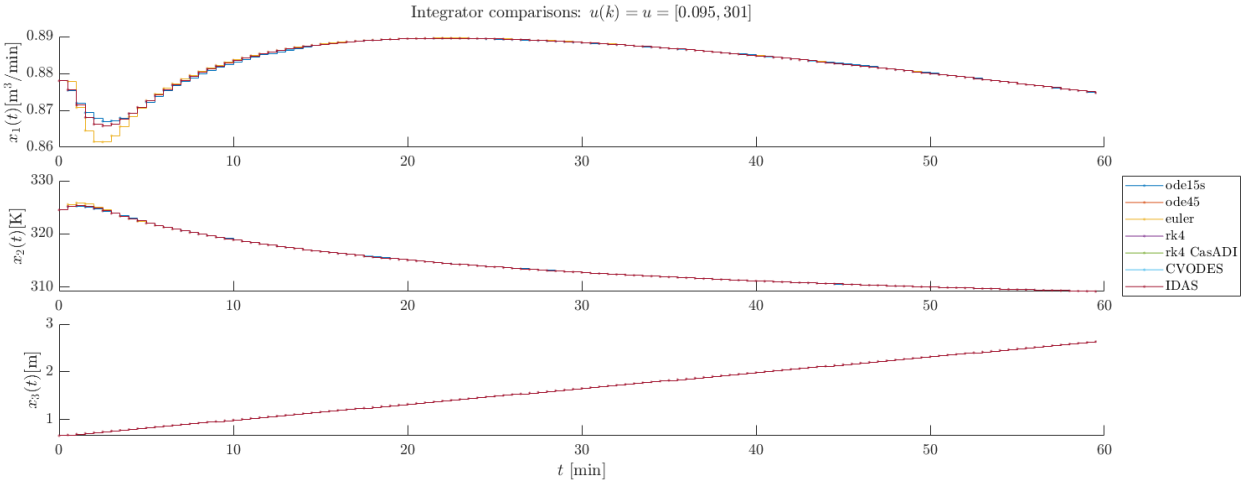
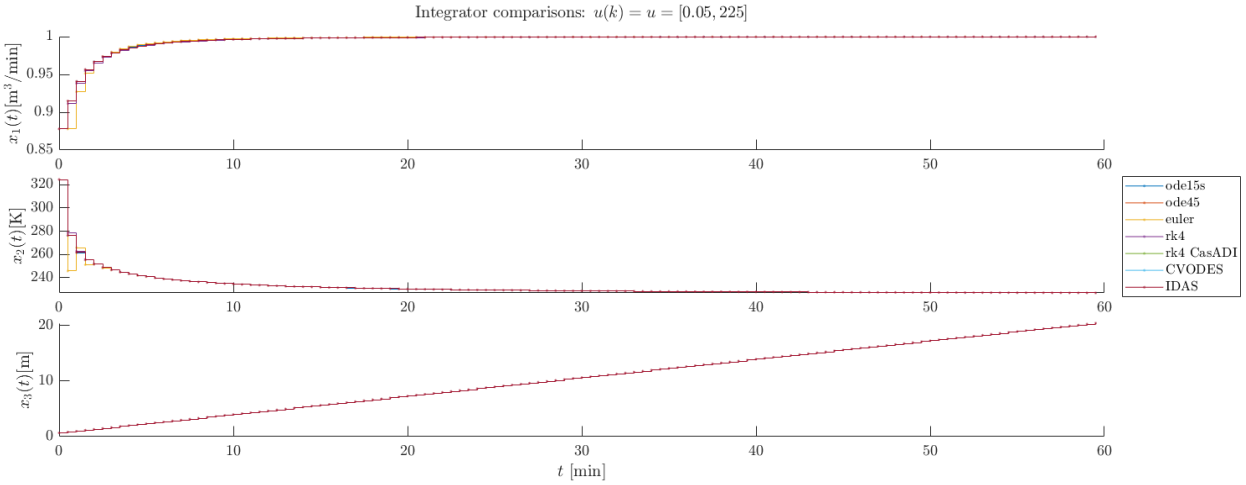


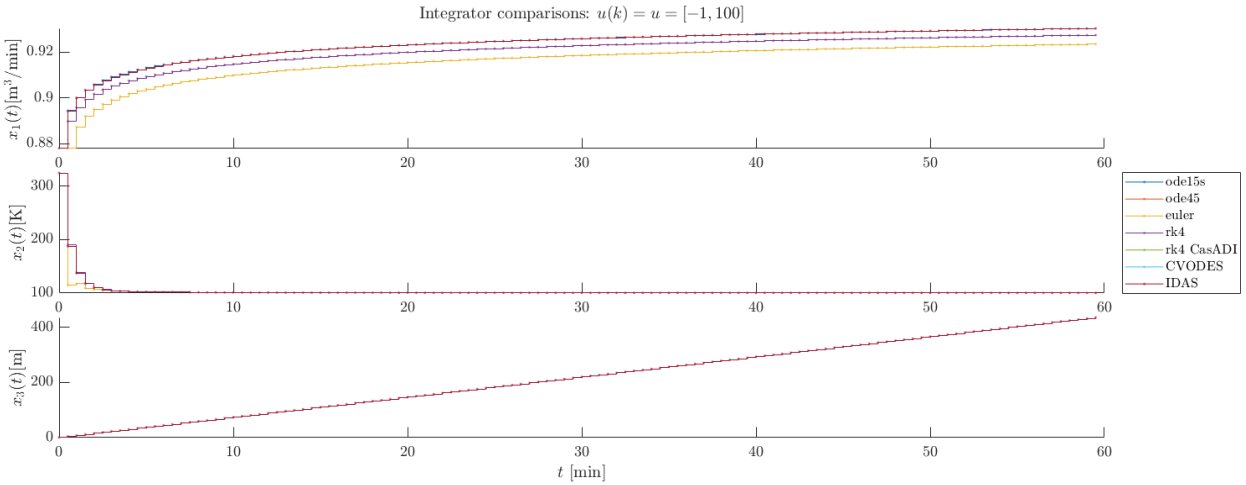
Figure 2: CSTR simulated at steady state control and state variable values and lower relative tolerance for the `ode45` solver.



(a)



(b)



(c)

Figure 3: CSTR simulated under different constant controls.

Exercise 02

Task 1

Consider the following uni-dimensional unconstrained optimisation problem

$$\min_{x \in \mathcal{R}} \frac{x^2 - 5x + 6}{x^2 + 1} \tag{3}$$

1. Plot the objective fucntion $f(x)$ and solve visually for the optimal value x^*

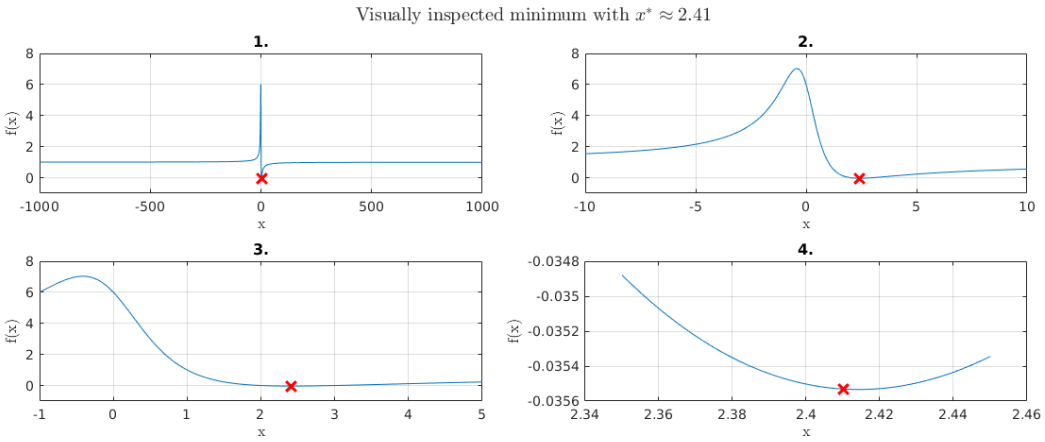


Figure 4: The objective function plotted for different ranges of x in the subfigures 1-4 to attempt a visual solution for x^*.

2. Derive on paper the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ of the objective function

We can first obtain the gradient by the quotient rule:

$$\begin{aligned} f(x) &= \frac{x^2 - 5x + 6}{x^2 + 1} \\ \nabla f(x) &= \frac{(2x - 5)(x^2 + 1) - (x^2 - 5x + 6)(2x)}{(x^2 + 1)^2} \\ &= \frac{2x^3 - 5x^2 + 2x - 5 - 2x^3 + 10x^2 - 12x}{(x^2 + 1)^2} \\ &= \frac{5(x^2 - 2x - 1)}{(x^2 + 1)^2} \end{aligned} \tag{4}$$

and similarly for the Hessian:

$$\begin{aligned}
\nabla f(x) &= \frac{5(x^2-2x-1)}{(x^2+1)^2} \\
\nabla^2 f(x) &= \frac{5(2x-2)(x^2+1)^2 - 5(x^2-2x-1)4x(x^2+1)}{(x^2+1)^4} \\
&= \frac{10x^3 - 10x^2 + 10x - 10 - 20x^3 + 40x^2 + 20x}{(x^2+1)^3} \\
&= \frac{10(-x^3+3x^2+3x-1)}{(x^2+1)^3}
\end{aligned} \tag{5}$$

3. Can you derive on paper the value x^* such that $\nabla f(x) = 0$? If positive, comment on $\nabla^2 f(x^*)$

We can easily solve for x^* by setting the gradient to 0:

$$\begin{aligned}
\nabla f(x) &= 0 \\
\frac{5(x^2-2x-1)}{(x^2+1)^2} &= 0 \\
x^2 - 2x - 1 &= 0 \\
\implies x^* &= 1 \pm \sqrt{2}
\end{aligned} \tag{6}$$

We now have two candidates for the minimiser and from Figure 4 it is clear that one of them is the minimiser and one the maximiser, since the gradient is equal to zero at both of these points. We can determine which is which by evaluating the Hessian at these points:

$$\begin{aligned}
\nabla^2 f(1+\sqrt{2}) &\approx 0.3033 \\
\nabla^2 f(1-\sqrt{2}) &\approx -10.3033
\end{aligned} \tag{7}$$

The first order optimality condition states that the gradient must be zero at optimal - minimal or maximal - values. The second order optimality condition can then be used to differentiate between them, by evaluating the Hessian at the optimal value(s). If the Hessian is positive semi-definite, then the optimal value is a minimiser - as is the case here with $x^* = 1 + \sqrt{2}$.

4. What would the minimiser be, had we included inequality constraints $x \in [0, 4]$

$$\begin{aligned}
&\min_{x \in \mathcal{R}} \frac{x^2 - 5x + 6}{x^2 + 1} \\
&\text{subject to } 4 \geq x \geq 0
\end{aligned} \tag{8}$$

The minimiser lies inside the constrained region and thus the addition of these inequality constraints would not affect the results.

5. Implement code to formulate both these problems and then solve them for the optimal values x^*

Below is a snippet of the full code for this task, where `opti` is used to formulate and solve the optimization problems.

```
%% 5. Use opti from CasADI to solve the optimization
opti = casadi.Opti();
x = opti.variable();
opti.minimize( (x^2 -5*x + 6)/(x^2 + 1) );
opti.solver('ipopt');

sol_uncons = opti.solve();

opti.subject_to( x <= 4 );
opti.subject_to( x >= 0 );
sol_constr = opti.solve();

xstar_uncons = sol_uncons.value(x)
xstar_constr = sol_constr.value(x)
```

6. Comment on the chosen solver and on the results of the optimisation.

I decided to use `ipopt` as the solver for these problems as it was used in earlier exercises in the course and in many of the examples. Both the unconstrained and constrained solutions returned a value of approximately 2.4142, which was expected and also equal to the minimizer $x^* = 1 + \sqrt{2}$ solved earlier.

Task 2

Consider the following two-dimensional constrained optimisation problem

$$\begin{aligned} &\min_{x,y \in \mathcal{R}} \; -20e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(\cos 2\pi x + \cos 2\pi y)} + e + 20 \\ &\text{subject to } x^2 + y^2 \leq 3 \end{aligned} \tag{9}$$

1. Plot the objective function $f(x, y)$ with the feasible set and solve for the optimal value (x^*, y^*)

Below, in Figure 5 are three different views of the objective function and the feasible set. The optimal value $(x^*, y^*) = (0, 0)$ was solved visually to start with.
Taking the gradient of the objective function we get:

$$\nabla f(x,y) = \begin{bmatrix} \pi e^{0.5(\cos 2\pi x + \cos 2\pi y)} \sin 2\pi x + 2x \frac{e^{-0.2\sqrt{0.5(x^2+y^2)}}}{\sqrt{0.5(x^2+y^2)}} \\ \pi e^{0.5(\cos 2\pi x + \cos 2\pi y)} \sin 2\pi y + 2y \frac{e^{-0.2\sqrt{0.5(x^2+y^2)}}}{\sqrt{0.5(x^2+y^2)}} \end{bmatrix} \tag{10}$$

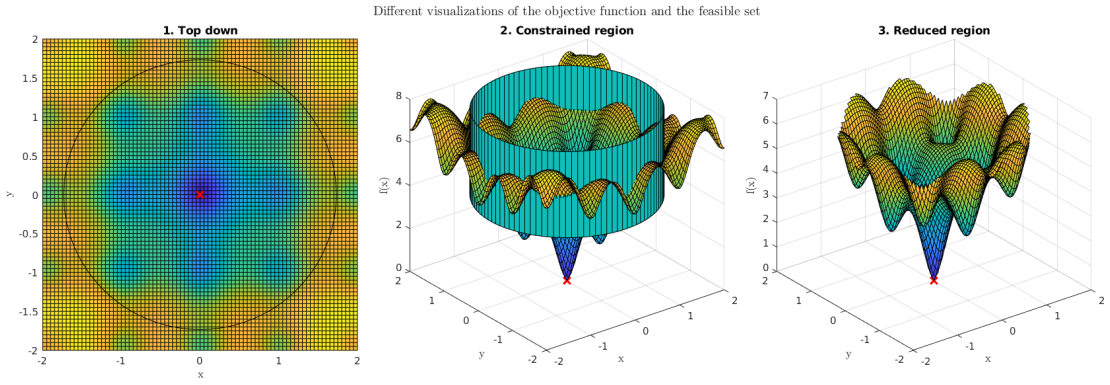


Figure 5: The objective function plotted with different visualizations of the feasible set.

We can make a noteworthy observation in the second term of the sum in both elements, namely the division by $\sqrt{0.5(x^2+y^2)}$, which indicates that the gradient does not exist at $(0,0)$ - the visually inspected optimal point. Without loss of generality we can set $y=0$ and take the limit

$$\begin{aligned}
&\lim_{x\to 0} \pi e^{0.5(\cos 2\pi x+\cos 0)} \sin 2\pi x + 2x\frac{e^{-0.2\sqrt{0.5x^2}}}{\sqrt{0.5x^2}} \\
=&\lim_{x\to 0} 2x\frac{e^{-0.2\sqrt{0.5x^2}}}{\sqrt{0.5x^2}} \\
\to&\lim_{x\to 0^-}(\cdots)\approx -2.8284 \\
\to&\lim_{x\to 0^+}(\cdots)\approx 2.8284
\end{aligned} \tag{11}$$

and we see the discontinuity. This may raise problems in the next step of coding the optimization problem.

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 16 randomly chosen and different initial solutions.

I decided to code the optimizer using the provided `simpleNewton.m` implementation of Newton's method. I modified the full step length to a fixed value of 0.2 as this seemed to produce better results at the cost of more iterations. Solutions can be seen in the below Figure 6 and a code snippet below that.

```
%% 2.1 Optimization using provided simpleNewton.m
N = 16; K = 10;
x0 = nan(N,2);
xt = nan(N,2,K+1); % trajectories
% pick initial points from uniform disk
r = sqrt(3)*sqrt(rand(N,1));
th = 2*pi*rand(N,1);
for n = 1:N
```

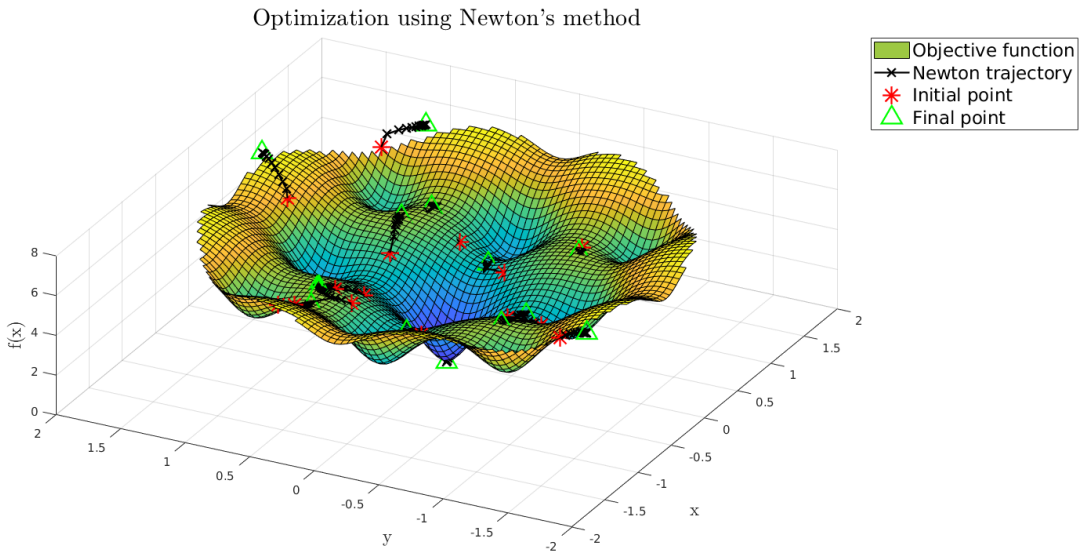


Figure 6: Optimization results using Newton's method with a fixed 0.2 step length.

```
    x0(n,:) = [r(n).*cos(th(n)); r(n).*sin(th(n))];
    xt(n,:,:) = simpleNewton(x0(n,:), jac, hes, K);
end
```

3. Comment on the chosen solver and on the results of the optimisation.

Newton's method performed reasonably though it needed some fine tuning for the step size and iteration count. From Figure 6 it is obvious that most of the solutions do not converge to the true optimal value, but rather get stuck in either saddle points or local minima and a "lucky" initial guess is needed for a global optimum. This is expected of Newton's method. An attempt was made for implementing the optimization with `ipopt` instead, but that did not seem to converge either (even with ludicrous iteration count).

Task 3

Consider the constrained optimisation of the N-dimensional Rosenbrock function

$$\begin{aligned} &\min_{x\in\mathcal{R}^{N+1}} \sum_{n=1}^{N} \left(100(x_{n+1}-x_n^2)^2 + (1-x_n)^2\right) \\ &\text{subject to } \sum_{n=1}^{N+1}(x_n-1)^2 \leq 2 \end{aligned} \tag{12}$$

1. Implement code to formulate this problem for $N = 8$, then solve it from different initial solutions

I decided to implement this problem using `ipopt` and `opti`. Evaluating the optimizer at different initial solutions was done by polling K points from the $N+1$ dimensional hypersphere defined by the inequality constraint function [2]. Code snippet below.

```
%% Task 3
N = 8;
opti = casadi.Opti();
x = opti.variable(N + 1);
% build function and constraints
ys = {};cs = {};
for n = 1:N
    ys{end + 1} = 100*(x(n + 1) -x(n)^2)^2 + (1 -x(n))^2;
    cs{end + 1} = (x(n) -1)^2;
end
cs{end + 1} = (x(N + 1) -1)^2;
% optimize
opti.minimize( sum([ys{:}]) );
opti.subject_to( sum([cs{:}]) <= 2 );
opti.solver('ipopt');
% sol = opti.solve();
```

```
%% Find optimal value
K = 25;
% pull initials from uniform N+1 dimensional hypersphere
% https://se.mathworks.com/matlabcentral/fileexchange/9443-random-points-in-an-n-
    dimensional-hypersphere?s_tid=answers_rc2-3_p6_MLT
x0 = randsphere(N+1,K,sqrt(2)) + 1;
xf = NaN(N+1,K);
for k = 1:K
    opti.set_initial(x,x0(:,k));
    sol = opti.solve();
    xf(:,k) = sol.value(x);
end
```

2. Solve for the optimal value x^* and comment on the chosen solver and on the results of the optimisation

All of the randomly chosen initial points resulted in an optimal point $\mathbf{x}^* = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$, which is the well known global minima of this n-dimensional extension of the Rosenbrock system [3].

Exercise 03

Consider the following optimal control problem for the CSTR system from Exercise 01 (eq 1 and 2), where the control variable boundaries are expressed as element wise inequalities

$$\begin{aligned} &\min_{\substack{x(0\rightsquigarrow t_f)\\ u(0\rightsquigarrow t_f)}} E(\mathbf{x}(t_f)) + \int_0^{t_f} L(\mathbf{x}(t),\mathbf{u}(t))dt \\ &\text{subject to } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)|\theta_{\mathbf{x}}), \quad t\in[0,T] \\ &\qquad \mathbf{u}_{\max} \le \mathbf{u}(t) \le \mathbf{u}_{\min}, \qquad t\in[0,T] \\ &\qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad (t=0) \end{aligned} \tag{13}$$

With terminal stage - and running costs

$$\begin{aligned} E(\mathbf{x}(t)) &= \frac{1}{2}\left((\mathbf{x}(T)-\mathbf{x}_{\text{ref}})^T\mathbf{Q}_{t_f}(\mathbf{x}(T)-\mathbf{x}_{\text{ref}})\right) \\ L(\mathbf{x}(t),\mathbf{u}(t)) &= \frac{1}{2}\left[(\mathbf{x}(t)-\mathbf{x}_{\text{ref}})^T\mathbf{Q}(\mathbf{x}(t)-\mathbf{x}_{\text{ref}}) + (\mathbf{u}(t)-\mathbf{u}_{\text{ref}})^T\mathbf{R}(\mathbf{u}(t)-\mathbf{u}_{\text{ref}})\right] \\ \mathbf{Q}_{t_f},\mathbf{Q} &\succeq 0, \quad \mathbf{R}\succ 0 \end{aligned} \tag{14}$$

The state equations and parameters are the same as used in Exercise 01. The reference state and control values are the steady state ones from Exercise 01 and the initial state is:

$$\mathbf{x}_0 = \begin{bmatrix} 0.9 & 0 & 0 \\ 0 & 1.1 & 0 \\ 0 & 0 & 1.0 \end{bmatrix} \mathbf{x}_{\text{ref}} \tag{15}$$

The upper- and lower-bounds of control are:

$$\begin{aligned} \mathbf{u}_{\min} &= 0.85\mathbf{u}_{\text{ref}} \\ \mathbf{u}_{\max} &= 1.15\mathbf{u}_{\text{ref}} \end{aligned} \tag{16}$$

Discrete-time optimal control formulation

Discretizing the system is simple when the control is assumed constant over the discretization interval:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)|\theta_{\mathbf{x}}) \qquad t\in[0,T] \\ \mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \int_t^{t+\Delta t} \mathbf{f}(\mathbf{x}(\tau),\mathbf{u}(\tau)|\theta_{\mathbf{x}})d\tau \\ &\approx \mathbf{x}(t) + \int_t^{t+\Delta t} \mathbf{f}(\mathbf{x}(\tau),\mathbf{u}_k(\tau)|\theta_{\mathbf{x}})d\tau \\ \mathbf{u}_k(\tau) &= \mathbf{u}(t=k\Delta t)\ \forall\ \tau\in[k\Delta t,(k+1)\Delta t) \\ \mathbf{x}(t+\Delta t) &\approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t),\mathbf{u}_k(t)|\theta_{\mathbf{x}})\Delta t \\ k\Delta t &\triangleq t, \quad (k+1)\Delta t \triangleq t+\Delta t, \quad K\Delta t \triangleq T \\ \rightarrow \mathbf{x}_{k+1} &\approx \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k|\theta_{\mathbf{x}})\Delta t \qquad k\in[0,1,\cdots K] \end{aligned} \tag{17}$$

The code for the sequential and simultaneous solutions was based on the code provided during the course in `toyOCP.zip` and further edited to fit the problem description. A final time of $t_f = 10$ minutes and a total of $K = 250$ steps was chosen through experimentation leading to a timestep of $\Delta t \approx 10ms$. A smaller step count may have sufficed, but numerical issues with the sequential approach seemed to be easier to deal with using a smaller timestep.

Sequential solution

Eliminating the state variables from the discretized system is done by writing the states as a sequence of controls, i.e.

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{x}_0 + \mathbf{f}(\mathbf{x}_0, \mathbf{u}_1|\theta_{\mathbf{x}})\Delta t \\ \mathbf{x}_2 &= \mathbf{f}(\mathbf{x}_0, \mathbf{u}_1|\theta_{\mathbf{x}})\Delta + \mathbf{f}(\mathbf{f}(\mathbf{x}_0, \mathbf{u}_1)|\theta_{\mathbf{x}}), \mathbf{u}_2|\theta_{\mathbf{x}}) \\ &\vdots \\ \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{f}(\cdots \mathbf{f}(\mathbf{x}_0, \mathbf{u}_1|\theta_{\mathbf{x}}), \cdots)\mathbf{u}_k|\theta_{\mathbf{x}}) \end{aligned} \tag{18}$$

The simulation results for the sequential solution can be seen in Figure 7. Through multiple different parameters, the control for the sequential solution always seemed to fail on numerical issues for when x_3 gets a value close to 0 thus blowing up the derivatives due to the volume calculation for the first two states. The regulator parameters used for the included graphs were:

$$\mathbf{Q}_{t_f} = 10^6\mathbf{Q},\ \mathbf{Q} = \mathbf{I}^{(3\times3)},\ \mathbf{R} = \mathbf{I}^{(2\times2)} \tag{19}$$

Simultaneous solution

The simulation results for the simultaneous solution can be seen in Figure 8. The simultaneous solution required adding a lower limit on the states such that the none of them could be zero or negative in addition to the bounds on the control variables. The regulator parameters used for the included graphs were:

$$\mathbf{Q}_{t_f} = 10^6\mathbf{Q},\ \mathbf{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ \mathbf{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{20}$$

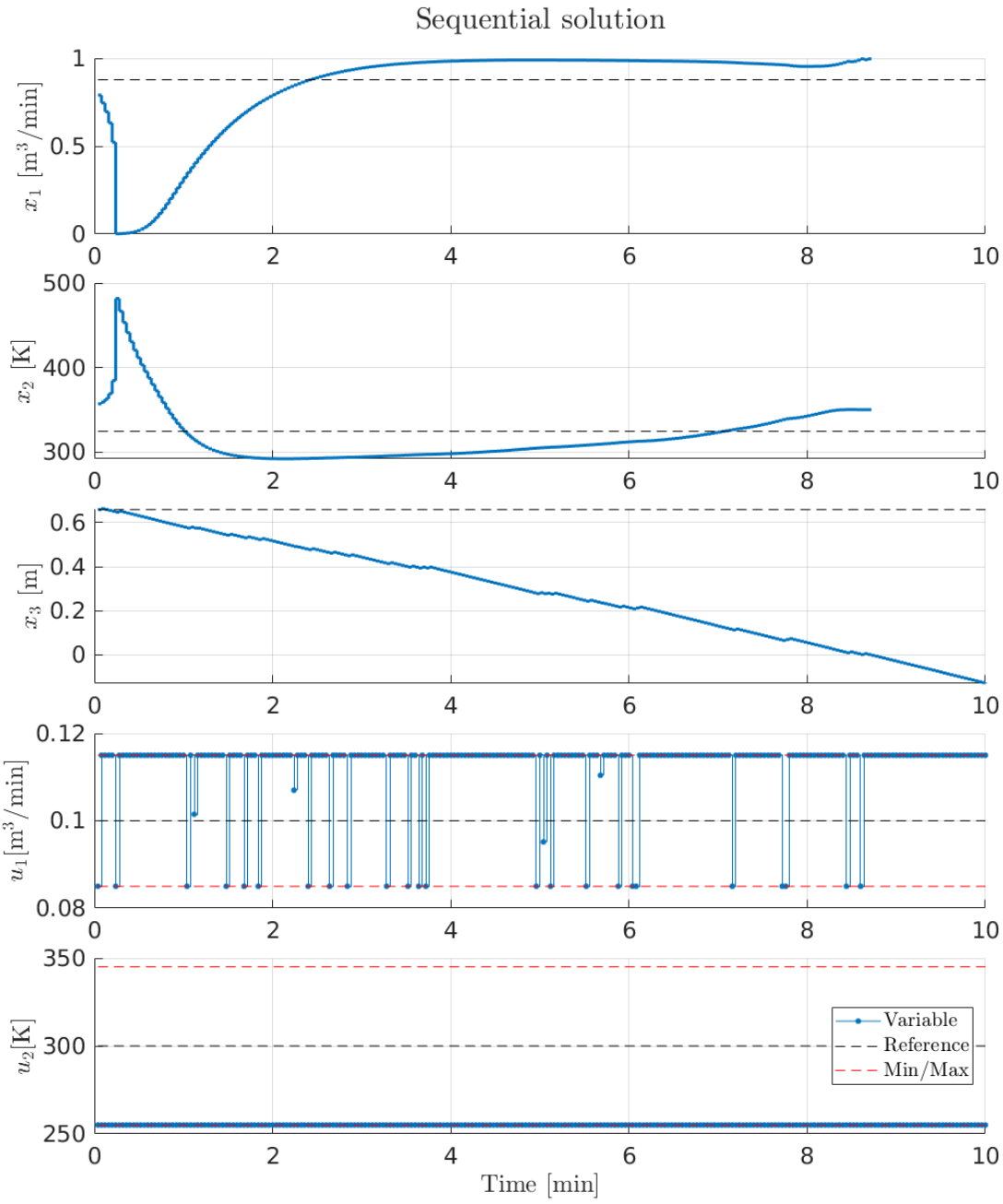


Figure 7: Solution to the optimal control problem using the sequential approach. Each state and control variable plotted separately with their respective reference and boundary values.

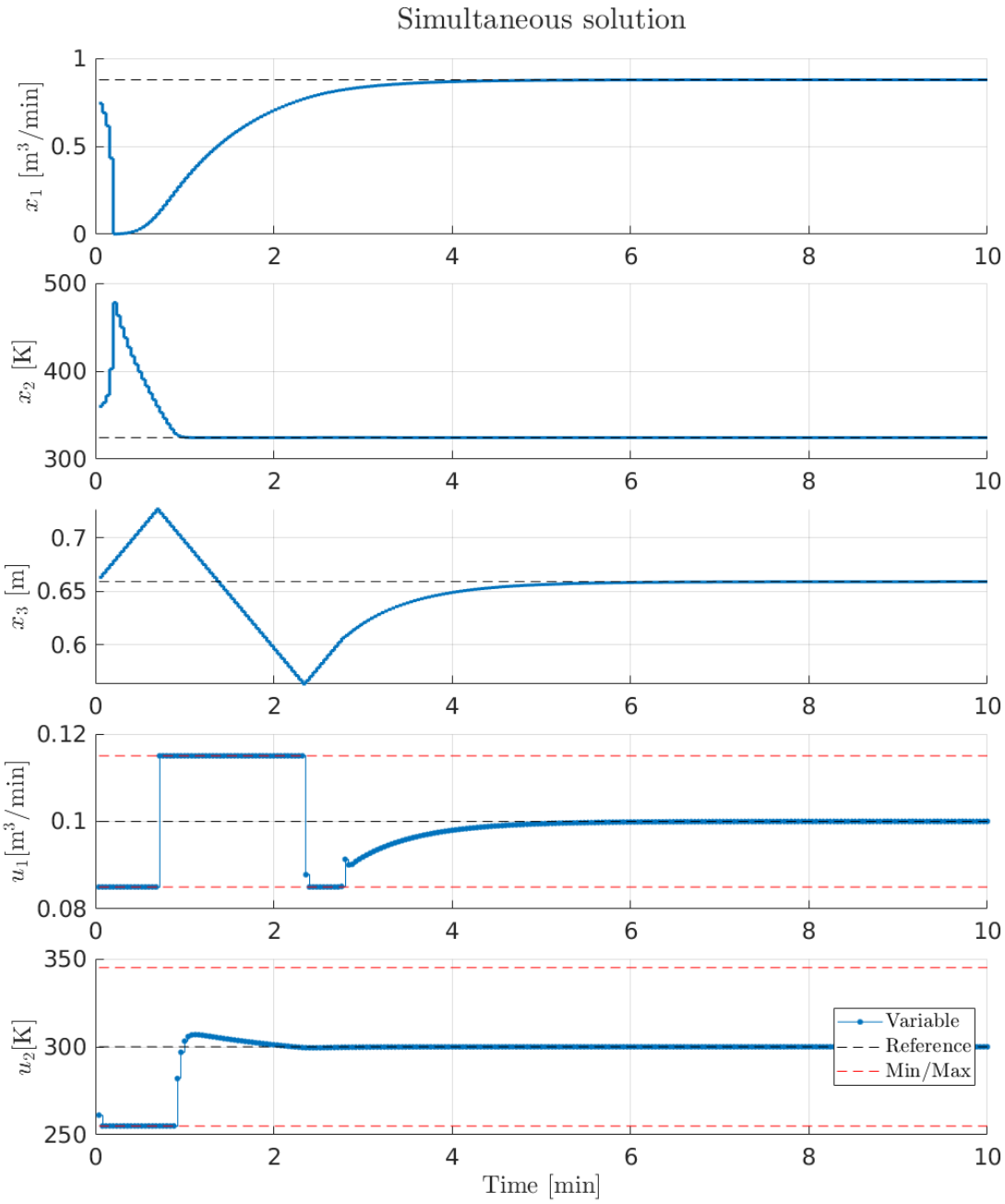


Figure 8: Solution to the optimal control problem using the simultaneous approach. Each state and control variable plotted separately with their respective reference and boundary values.

References

[1] J.B. Rawlings G. Pannocchia. "Disturbance models for offset-free MPC". In: *AIChE J.* 49(2): 426-437 (2003).

[2] Roger Stafford. *Random Points in an n-Dimensional Hypersphere*. Retrieved May 27, 2021. URL: https://www.mathworks.com/matlabcentral/fileexchange/9443-random-points-in-an-n-dimensional-hypersphere.

[3] Wikipedia contributors. *Rosenbrock function — Wikipedia, The Free Encyclopedia*. [Online; accessed 27-May-2021]. 2020. URL: https://en.wikipedia.org/w/index.php?title=Rosenbrock_function&oldid=993612771.