TASK BRIEF

MPC Upskilling

Submission details:

Type:	Code, and PDF with results and discussion.
Length:	1-3 figures and 1-3 paragraphs for each of items (D), (E), and (F) described in the task details.
Weighting:	6% of the overall subject grade.
Due Date:	Friday 25 Aug 23:59 (i.e., end of semester week 5).
Instructions:	Upload your code and PDF to the LMS.

The following learning outcomes are demonstrated through this assessment activity:

- **ILO2)** Analyse and implement model-based methods for providing stability and performance, such as: PID, MPC, system identification, or adaptive control schemes.

PURPOSE STATEMENT

Thee purpose of this task is to give you hands-on experience with implementing MPC from scratch and then using that implementation to investigate some design trade-offs for a simple example. The reason for implementing MPC from scratch is similar to way we teach and learn many topics at university:

(1) First learn the fundamentals;
(2) Then apply the fundamentals "by hand" to examples that capture the essence of the topic;
(3) Then investigate how these fundamentals are incorporated into existing software, and "fact check" the software results against our own "by hand implementation";
(4) Finally, graduate to using existing software to address complex problems.

This task is part of **step 2 for an MPC teaching and learning journey**.

TASK DESCRIPTION

SUMMARY:
Develop your own MPC implementation for a simple linear time-invariant model where you:

(A) Construct the matrices that are necessary for passing an MPC optimization program to a general-purpose optimization solver.
(B) Pass those matrices to the solver and interrogate the results provided by the solver.
(C) Simulate closed-loop trajectories of the MPC policy.

Use this implementation to investigate the following trade-offs and parameter influence:

(D) Horizon length versus computation time and performance.
(E) Objective function weights on position versus velocity states, and on state weights versus action weights.
(F) Terminal objective options.

DETAILS:
For parts (A), (B), and (C) you can use the following step-by-step guide to fully complete these parts:
https://people.eng.unimelb.edu.au/pbeuchat/ai4r/appendix_01_mpc_implementation.html
Section 2 on that page provides the parameters of a **simple 2D point-mass model** that we strongly encourage you to use for this assignment.

For each of parts (D), (E), and (F) you are expected to provide:

- One or more graphs showing the results of your trade-off / parameter investigations.
- 1-3 paragraphs describing and discussing the trade-offs shown by your results.

The following provides more detail of what investigation is expected for each of these parts:

- **For part (D):** Create and discuss plots showing, for the example system being considered, how:
  - Horizon length influences computation time of the optimization solver.
  - Horizon length influences the trajectories predicted by MPC.
  - Horizon length influences performance.
- **For part (E):** Create and discuss plots showing the performance of 6 (or more) options for the the weightings of the Q and R matrices. Your weights should, as a minimum, investigate:
  - Within the Q matrix, the relative weights on a position state versus a velocity state.
  - The relative overall weight of the Q matrix versus the R matrix. To do this, use a scalar $\alpha \in [0, \infty)$ to adjust the relative weight as $s^T Q s + a^T (\alpha R)\, a$
  - **Hint:** the relative weight $\alpha = 0$ here is an important case to include in your results because it represents the actions having no cost (nor reward) in the objective function.
  - Explicitly discuss any intuition you gain about how adjustments to the weights influences the system performance.
- **For part (F):** Create and discuss plots showing how the choice of terminal objective effects the predicted trajectories and performance. Consider the following three terminal objective options:
  - Zero, i.e., P=0
  - Same as per-stage objective, i.e., P=Q
  - Ricatti solution, Section 2 of the step-by-step guide linked above provides the line of code for computing the Ricatti solution, i.e.:
    `P = scipy.linalg.solve_discrete_are(A,B,Q,R)`
  - **Hint:** a short MPC time horizon makes the difference more pronounced between these choices of terminal objective.

CRITERIA GUIDANCE

- All figures should:
  - Have a label on each axis.
  - Have a legend.
  - Have grid lines (unless these significantly clutter the figure).
- For comparing trade-offs with performance, one needs a metric for performance. However, there are many viable metrics you can consider for these parts. We recommend the following metrics:
  - **For parts (D) and (F):** the choice of the Q and R objective function matrices remains fixed at some value, hence use the performance metric of the sum over the simulation horizon of per-stage objective function, i.e.:

$$\sum_{k=0}^{N_{\mathrm{sim}}} \left(s_k^T \, Q \, s_k \; + \; a_k^T \, R \, a_k \right)$$

  - **For part (E):** the Q and R matrices are adjusted, hence you need to compare multiple metrics of performance to understand the trade-offs. You can use the following metrics as a starting point and add any additional metrics that you find to be insightful:
    - Overshoot past the x-axis and y-axis relative to the starting position.
    - Number of simulation time steps required to first get to within some small radius of the origin (small relative to the starting location).
    - Sum of the absolute magnitude of each element of the actions, as a proxy for the total

action resource/energy consumed, i.e.:

$$\sum_{k=0}^{N_{\mathrm{sim}}} \left(\|a_k\|_1 \right)$$

* Maximum change in action between subsequent time steps, as a proxy for aggressiveness of the policy, i.e.:

$$\max_{k=1,\dots,(N_{\mathrm{sim}}-1)} \| \, a_k - a_{k-1} \, \|_2$$

* Any other metric you wish to propose for quantitative assessment of the results that you observe.

- You are **strongly encouraged to code in Python** (even if you feel more comfortable in Matlab, C++, or another language), however, you are not penalized for coding in another language. The two main motivations for using Python are:
  - The step-by-step MPC implementation guide linked above is in Python.
  - The hands-on experiments later in semester will be in Python.
- You are allowed to use a system that is different from the 2D point-mass model provided in the step-by-step MPC implementation guide. Please discuss your alternative system choice with the teaching team to ensure it can satisfy the purpose of this task.

MARKING GUIDE

6% of the overall subject marks are awarded as **2% for each of parts (D), (E), and (F)**. The 2% for each of those parts is awarded as:

- 2% is awarded for figures and discussions that are clear, correct, insightful, and address the trade-offs specified.
- 1% is awarded for figures and discussions that are simplistic and/or ambiguous and/or incorrect.
- 0% is awarded for figures and discussions that do not address the respective part in any meaningful fashion.