ELEC9123: Design Task F
(AI for Telecommunication System)
Reinforcement Learning for Trajectory Design in UAV-aided Telecommunication Systems

Co-designed by Mahnoor Anjum

Term T2, 2025

1 Background

This project investigates the application of Artificial Intelligence (**AI**), specifically reinforcement learning, to tackle complex optimisation problems in modern telecommunication systems. As wireless connectivity expands and the demand for reliable service increases in remote or disaster-affected areas, aerial base stations (ABS) mounted on unmanned aerial vehicles (UAVs) have emerged as an innovative solution.

In this task, you will model a UAV-based ABS that can fly over a remote area to provide wireless coverage to users on the ground. The primary challenge is to intelligently optimise both the UAV's flight path, referred to as a trajectory, and its signal transmission strategy to ensure maximum data delivery to users. This requires designing an AI engine capable of making real-time decisions in dynamic environments, which is crucial for the future of smart, adaptive communication networks.

This project is highly relevant today, as AI-driven, energy-efficient mobile communication systems are becoming essential for emergency response, rural connectivity, and the development of next-generation 6G networks. By participating in this task, you'll gain practical experience in applying an **AI engine** to real-world wireless networking challenges, preparing you for a future in intelligent and sustainable telecommunications.

1.1 Optimization

Optimization is the process of determining the best set of decision variables which maximize a given objective function while satisfying all the problem-specific requirements. Formally, an optimization problem can be stated as:

$$\begin{aligned} \text{(P1:)} \quad & \underset{\mathbf{x}\in\mathbb{R}^n}{\text{maximize}} \quad f(\mathbf{x}) \\ & \text{subject to} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m, \\ & \qquad\qquad\quad h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p. \end{aligned}$$

In (P1), $\mathbf{x}$ denotes the design variable. We observe that the search space for $\mathbf{x}$ is also described in the problem i.e., $\mathbf{x} \in \mathbb{R}^n$. This indicates that $\mathbf{x}$ is an n-element vector, which can only have real-values. The function $f(\mathbf{x})$ denotes the objective of the problem. The objective function of an optimization problem has the following characteristics:

- The objective function must directly, or indirectly, take design variables as inputs.
- The objective function must output a scalar value[1] i.e., $f(x) \in \mathbb{R}$.

The m functions $g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m$, denote the inequality constraints of the problem[2]. Similarly, the p functions $h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p$ denote the equality constraints of (P1). An optimization problem is considered solved a candidate assignment of the decision variables is computed such that it meets every one of the problem's constraints and yields the highest[3] achievable objective value.

1.2 Problem Formulation

In this section, we will explore the steps required to cast any real-world problem into a mathematically sound optimization problem. We will discuss two different real-world problems to understand problem formulation:

- (R1:) Fruit basket optimization: You have $30 to spend on a fruit basket. You wish to buy the maximum amount of each fruit that the budget allows.

[1] Multi-objective optimization is out of scope for this project.
[2] It is the standard notation to write inequality constraints with 0 on the right-hand-side.
[3] The term highest is chosen relative to (P1). Optimization problems encompass both maximization and minimization formulations.

- (R2:) Traveling salesperson problem: You're a salesperson who must visit N different towns and then return home. You wish to save fuel and only travel once to each town via the optimal path.

Table 1: Analysis of the two simple optimization problems

Aspect	**Fruit basket optimization**	**Traveling salesperson problem**
Objective	Maximize the number of each fruit	Minimize total tour length
Design variables	# of bananas, # of apples, # of strawberries, etc.	Visit order of the different towns.
Constraints	Budget \$30	Visit each city at least once and return to start
Goal	Spend up to \$30 to get as much fruit as possible of each kind	Visit all N cities in a closed loop of minimum length

To formulate a problem, we first analyze the problem statement and obtain an overview of the design variables, objectives, and constraints. For (R1) and (R2), we provide an analysis in Table 1. Based on this analysis, the fruit basket optimization problem can be formally written as follows:

$$\begin{aligned} \text{(R1:)} \quad & \underset{t,\, \mathbf{x}\in\mathbb{R}^F}{\text{maximize}} \quad t \\ & \text{subject to} \quad x_i \geq t, \quad i = 1, \ldots, F, \\ & \qquad\qquad \sum_{i=1}^{F} p_i\, x_i \leq B, \\ & \qquad\qquad x_i \geq 0, \quad i = 1, \ldots, F, \end{aligned}$$

where F is the number of fruit types, x_i is the quantity of fruit i, $\mathbf{x} = [x_1, x_2, \ldots, x_F]^T \in \mathbb{R}^F$ is the vector of fruit count, p_i is the cost per unit of fruit i, B is the total budget, and t is the minimum quantity across all fruits. Maximizing t will maximize the minimum number of fruit bought of a particular type. This will then maximize the total number of fruit of each type bought by the \$30 budget. Similarly, the traveling salesperson problem is recast as follows:

$$\begin{aligned} \text{(R2:)} \quad & \underset{\{x_{ij}\}}{\text{minimize}} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}\, x_{ij} \\ & \text{subject to} \quad \sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, \ldots, N, \\ & \qquad\qquad \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, \ldots, N, \\ & \qquad\qquad x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, N, \end{aligned}$$

where, N denotes the total number of towns including the starting and ending location, and d_{ij} represents the distance of traveling directly from town i to town j. The binary decision variable $x_{ij} \in \{0, 1\}$ takes the value 1 if the salesperson visits town j immediately after town i, and 0 otherwise.

The conventional strategy of tackling problems (R1) and (R2) is to derive relaxed convex approximations of these problems which can be solved using efficient standard solvers e.g., *CVX Toolbox*. However, conventional methods can incur prohibitive computational costs with high dimensional design variable spaces. Furthermore, in large scale problems, search space can grow factorially or exponentially i.e., $O(n!)$ or $O(2^n)$, which can render conventional methods infeasible. To mitigate this, reinforcement learning is emerging as a promising technique as it can cast an optimization problem as a sequential decision making problem. In reinforcement learning, a mathematical agent learns a policy via sampled interactions and reward signals, which scales efficiently to high dimensional spaces without requiring convex relaxations.

1.3 Reinforcement Learning

Formally, reinforcement learning is a type of machine learning in which a mathematically modeled agent learns to make optimized decisions by interacting with an environment and receiving rewards or penalties for its actions. More interestingly, reinforcement learning is a branch of artificial intelligence. We can think of a reinforcement learning agent as a dog. We want to teach the dog a new trick. To do so we will:

- Define a goal. In this case, the dog needs to learn how to do a "roll over" when you whistle.
- Set a reward. While training the dog, if it does a "roll over" when you whistle, you give it a treat. If it does not do a "roll-over", you do not give it a treat.

- Define an action. In this case, you whistling would be the action. If the dog does the “roll-over" at any other action e.g., claps, etc., you do not give the dog a treat.

In reinforcement learning, tasks are formalized as Markov Decision Processes (MDPs), which provide the mathematical framework for sequential decision making. In the following, we will learn how an MDP is formulated for a reinforcement learning task.

1.4 Markov Decision Processes

An MDP is formulated by utilizing states, actions, and rewards related to the specific environment:

1. **Episode**: An episode is one complete interaction between the agent and its environment. It begins in an initial state s_0 and ends when a terminal condition is reached. The terminal condition can be defined by the episode length L.
2. **State** (s_t): Defines the agent's current observation of the environment at time t. This information is available to the agent and assists in taking optimized actions at particular timesteps. All the information needed to make the next decision is encoded in the state.
3. **Action** (a_t): Describes the action chosen by the agent in state s_t. After applying this action to the environment, the environment can transition into the next state. The goal of the agent is to optimize the actions taken in each state.
4. **Reward** (r_t): Defines the feedback relative to a particular state-action pair. This is a scalar quantity received immediately after taking a_t in state s_t and quantifies the quality of that step. The goal of the agent is to maximize the reward obtained in each step throughout the episode.
5. **Next State** (s_{t+1}): Describes the new observation that the agent receives after executing a_t in state s_t and receiving r_t.

An MDP is mathematically specified by the tuple

$$(s_t,\ a_t,\ r_t,\ s_{t+1}).$$

The agent learns by sampling these (s_t, a_t, r_t, s_{t+1}) transitions under a specific policy and using them to develop the optimal policy π^*. To further understand how an MDP is formulated for a problem, we provide sample MDPs for (R1) and (R2) as follows:

Fruit basket optimization (R1):

- **State**: $s_t = (b_t, \mathbf{x}_t)$ where $b_t \in [0, B]$ and $\mathbf{x}_t \in \mathbb{Z}^F$. This state provides information on the budget which is left, and the amount and type of fruit already bought in timestep t.
- **Action**: $a_t = (i_t, q_t)$ with $i_t \in \{1, \ldots, F\}$ denoting the index of a particular type of fruit, and $q_t \in \mathbb{Z}_{\geq 1}$ defining the quantity of the fruit which has to be bought in timestep t.
- **Reward**: $r_T = \min_i x_{i,T}$ i.e., the minimum amount of the fruit present in the basket at timestep t
- **Episode termination**: Episode terminates when the budget $b_t \leq 0$.

Traveling salesperson problem (R2):

- **State**: $s_t = \big(c_t, \mathbf{v}_t\big)$, where $c_t \in \{1, \ldots, N\}$ is the current town at time step t, and $\mathbf{v}_t = [v_{1,t}, \ldots, v_{N,t}]^\mathsf{T} \in \{0,1\}^N$ is the visited indicator vector with $v_{i,t} = 1$ if town i has already been visited.
- **Action**: $a_t = j$, where $j \in \{k \mid v_{k,t} = 0\}$ selects the next unvisited town. When all towns have been visited ($\sum_{i=1}^{N} v_{i,t} = N$), a predefined action $a_t =$ return takes the salesperson to the starting town.
- **Reward**: $r_t = -d_{ij}$, where d_{ij} is the distance from town i to town j. Maximizing $\sum_{t=0}^{T-1} r_t$ minimizes the total tour length.
- **Episode termination**: The episode terminates when all the towns have been visited and the salesperson comes back to the starting town.

1.5 System Model

This task focuses on downlink communication from an ABS to K communication users. The communication users are deployed in a three-dimensional rectangular grid such that a point in the coordinates is represented by (x, y, z) where $x \in [x_{min}, x_{max}]$, $y \in [y_{min}, y_{max}]$, and $z \in [z_{min}, z_h]$. The K users are randomly distributed on ground i.e., (x_k, y_k, z_{min}) is the location of the k^{th} user. The ABS is mounted on a UAV which:

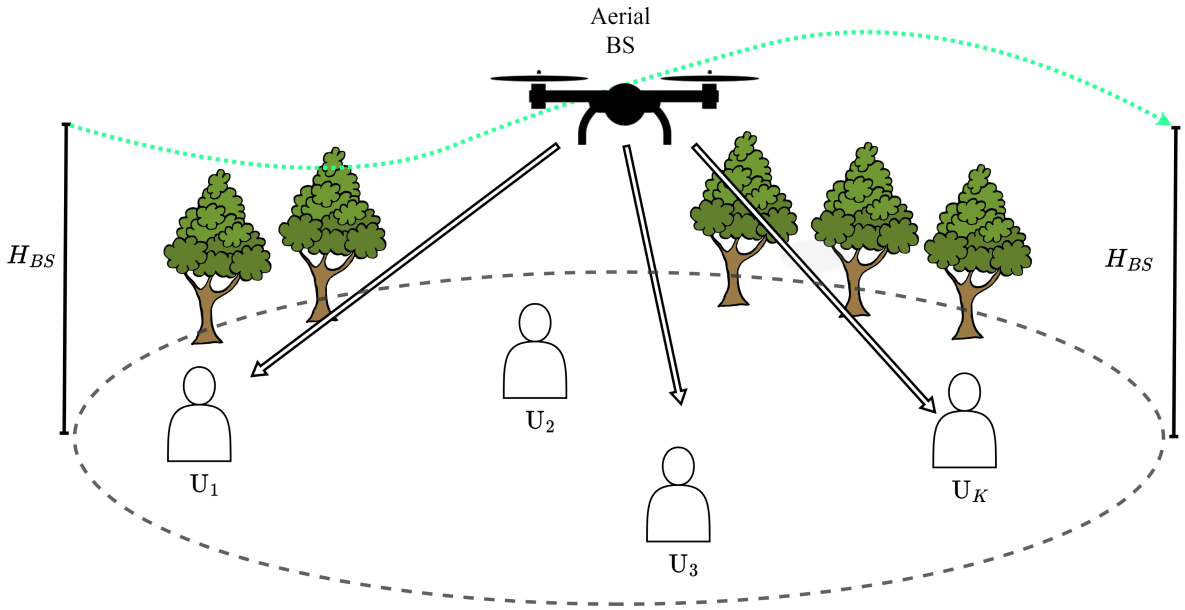

Figure 1: Illustration of the system model for a UAV-aided wireless system.

- Starts its flight at (x_0, y_0, z_h).
- Ends its flight at (x_L, y_L, z_h).
- Has a total flight time of 200 to 300 seconds.
- Has a total speed between $10 - 30$ meters per second.
- Can move in the x direction.
- Can move in the y direction.
- Has a fixed height of flight.

The ABS is equipped with N_t transmit antennas which enables multi-antenna transmission. The system model is visualized in Fig. 1.

1.6 Signal Model

In this section, we delineate a single-antenna transmission signal model. We also elaborate the performance metrics involved in information transfer through wireless links. The received signal from a single-antenna ABS[4] to a single-antenna communication user is formulated as:

$$y_k(t) = h_k x_k(t) + n(t), \tag{1}$$

here, $y_k \in \mathbb{C}$ is the complex baseband received signal at user k at time t; $x_k \in \mathbb{C}$ is the transmitted complex baseband signal for user k at time t with average power $P = \mathbb{E}[|x_k(t)|^2]$ where $\mathbb{E}[\cdot]$ is the expectation operator, $h_k \in \mathbb{C}$ is the complex channel gain for user k and $n \sim \mathcal{CN}(0, \sigma^2)$ is the additive white Gaussian noise sample of variance σ^2. The channel h_k is modeled as the line of sight (LoS) link between the transmit antenna and the receive antenna i.e.,

$$h_k = \sqrt{\frac{L_0}{d_k^{\eta}}} h_k^{LoS}, \tag{2}$$

where $L_0 = \left(\frac{\lambda}{4\pi}\right)^2$ with λ denoting the wavelength of operation, d_k is the distance between the transmit and receive antennas, and η is the path loss exponent of the environment. The LoS component $h_k^{LoS} = e^{j\frac{2\pi}{\lambda}d_k}$. The received signal at the user k is utilized to formulate the metric for communication efficiency; the signal-to-noise ratio (SNR) defined as

$$\text{SNR}_k(t) = \frac{\mathbb{E}\left[|h_k x_k(t)|^2\right]}{\mathbb{E}\left[|n(t)|^2\right]} = \frac{P}{\sigma^2}. \tag{3}$$

[4]Explore beamforming and multiple-input multiple-output systems to formulate the signal model for multi-antenna ABS.

SNR is a measure of how strong the desired signal is relative to the additive noise at the receiver. A higher SNR indicates that the signal power P dominates the noise variance σ^2. This consequently leads to more reliable symbol detection and higher achievable data rates. The effective normalized throughput of this communication link is formulated as follows:

$$R_k(t) = \log_2\big(1 + \mathrm{SNR}_k(t)\big) \quad \text{[bits per channel use]}. \tag{4}$$

2 Design Task Guide

The design task requirements are described in this section.

2.1 UAV Trajectory Modeling

We first model a single episode of the UAV trajectory as follows:

1. Simulate a 3D environment with dimensions $(x_{min}, y_{min}, z_{min})$ to (x_{max}, y_{max}, z_h) using the Python programming language.
2. Utilize the uniform random distribution to simulate K users in the XY-plane of the environment.
3. Model a UAV positioned at (x_0, y_0, z_h) at timestep $t = 0$.
4. Equip the UAV with N_t antennas organized in a uniform linear array along the Y-axis.
5. Compute the channel vectors $\mathbf{h}_k(t) \in \mathbb{C}^{N_t \times 1} \; \forall \; \{1, 2, \ldots, K\}$.
6. Initialize the transmit signal vectors.
7. Formulate the $SNR_k(t) \; \forall \; \{1, 2, \ldots, K\}$.
8. Utilize the $SNR_k(t)$ to obtain the throughput $R_k(t)$ for all users.
9. Compute sum throughput $R(t) = \sum_{i=0}^{t} \sum_{k=1}^{K} R_k(i)$.
10. Reposition the UAV at the next location.
11. Repeat steps 5 to 10 for episode length L.

The system parameters are shown in Table 2.

Table 2: Simulation parameters

Parameter	**Symbol**	**Value**
Minimum value of the x coordinate	x_{min}	0 m
Minimum value of the y coordinate	y_{min}	0 m
Minimum value of the z coordinate	z_{min}	0 m
Maximum value of the z coordinate	z_h	50 m
Maximum value of the x coordinate	x_{max}	100 m
Maximum value of the y coordinate	y_{max}	100 m
Path loss exponent	η	2.5
UAV start location	(x_0, y_0, z_h)	(0,0,50) m
UAV end location	(x_L, y_L, z_h)	(80,80,50) m
Number of users	K	2
Number of antennas	N_t	8
Transmit power budget	P_t	0.5 Watts
Noise power	σ^2	-100 dB
Episode length	L	200 to 300 seconds
Frequency of operation	f	2.4 GHz
UAV speed	v	10 to 30 m/s

2.2 Problem Definition

Following the environment definition, we start the analysis of our problem. First, we verbalize the goal of the problem which is to be solved in this task:

Goal: Maximize the sum throughput of one complete episode of the UAV-aided wireless system.

We now observe the decision variables which can be optimized to achieve the desired objective. We note that:

- The UAV trajectory can be optimized to provide higher sum throughput.
- The transmit signal can be designed at every step of the episode to provide higher throughput.

Therefore, we choose the following as the design variables for the task:

Design variables: UAV trajectory $\{(x_1, y_1, z_h), (x_2, y_2, z_h), \ldots, (x_{L-1}, y_{L-1}, z_h)\}$, transmit signal vectors $\{\{\mathbf{w}_k(t)\}_{k=1}^{K}\}_{t=0}^{L}$.

Now that we have defined the problem and its design variables, we discuss the constraints of the problem. The following must be adhered to during the course of the episode:

- The UAV must not fly out of the simulated 3D grid.
- The UAV must start at (x_0, y_0, z_h).
- The UAV must not gain or lose height during the trajectory optimization.
- The UAV must end at (x_L, y_L, z_h) when the episode concludes.
- The transmit signal power must not exceed the transmit power budget.

We formally define the problem:

$$
\begin{aligned}
\text{(T1:)} \quad & \underset{\{(x_1,y_1,z_h),\ldots,(x_{L-1},y_{L-1},z_h)\},\{\{\mathbf{w}_k(t)\}_{k=1}^{K}\}_{t=0}^{L}}{\text{maximize}} \quad \sum_{t=0}^{L}\sum_{k=1}^{K} R_k(t) \\
& \text{subject to} \quad \sum_{k=1}^{K} ||\mathbf{w}_k(t)||^2 \leq P_t \;\forall\, t \in \{0, 1, \ldots, L\}, \\
& \qquad x_i \geq x_{min} \;\forall\, i \in \{1, 2, \ldots, L-1\} \\
& \qquad x_i \leq x_{max} \;\forall\, i \in \{1, 2, \ldots, L-1\} \\
& \qquad y_i \geq y_{min} \;\forall\, i \in \{1, 2, \ldots, L-1\} \\
& \qquad y_i \leq y_{max} \;\forall\, i \in \{1, 2, \ldots, L-1\} \\
& \qquad (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \leq (v_{\max}\, \Delta t)^2, \;\forall\, i \in \{2, 3, \ldots, L-1\},
\end{aligned}
$$

where $v_{\max}$ is the speed of the UAV, and Δt is the resolution of one step of the episode. Note that the trajectory design starts at $t = 1$ and ends at $t = L - 1$. This is owing to the given condition that the UAV must start at (x_0, y_0, z_h) and end at (x_L, y_L, z_h).

2.3 Design Task Requirements

This section provides an overview of the steps which can be taken to obtain a solution to the problem (T1).

1. The first step of obtaining a reinforcement learning based solution is the formulation of an MDP (see Section 1.4). We must:
   - Explore if the solution can be divided into sub-problems for efficiency.
   - Utilize the problem definition to design an action space $\mathcal{A}$.
   - Design a state space $\mathcal{S}$ which completely and correctly captures the information of the environment.
   - Design a reward function which adheres to the constraints of the problem (T1).
2. After the formulation of the MDP, the environment should be modeled in Python. The signal, channel and system models must be verified using visualizations and plots.
3. Once the environment is modeled, the designer must explore different deep reinforcement learning algorithms e.g., proximal policy optimization (PPO), soft actor critic (SAC) and deep Q learning (DQN) which may be well suited for their MDP formulation. The choice of the algorithm highly depends on the nature of the action and state spaces i.e., if they are discrete, continuous or mixed.
4. Evaluation Metrics:
   The performance of your solution will be evaluated on the following fronts:
   - The model reward converges.
   - UAV reaches the end-location at the end of the trajectory.
   - UAV has a total flight time of 200 to 300 seconds.

- UAV has a total speed between 10 m/s to 30 m/s.
- Channel is modeled correctly.
- Pathloss is modeled correctly.
- Transmit signal is designed correctly.
- The solution is efficient with only the relevant information in the state space and action space.
- The reward signal sufficiently captures the objective and constraints of the problem.
- UAV reaches the locality of K users during its trajectory.
- UAV hovers around the locality of the K users.
- All K users are served during the trajectory of the UAV.

5. `Python` Implementation:
   To ensure your Python simulation and reinforcement learning implementation meets the expectations of this design task, please adhere to the following requirements as they are assessed during evaluation:
   - Task is implemented in `Python`. All primary functions and scripts must be clearly structured and stored as `.py` or `.ipynb` files.
   - The reinforcement learning models utilized for the solution must be from a standard library e.g., Stable-baselines3.
   - Your code should be clean, modular, and readable.
   - Include meaningful inline comments that explain the purpose of key operations.
   - Use consistent indentation and variable naming schemes for clarity.
   - A set of instructions must be provided on how to run your code.
6. Simulation Results and Analysis:
   To demonstrate the correctness and insights of your simulation results, you are required to generate the following plots and provide a brief but insightful analysis for each:
   - Signal power vs. transmitter-receiver distance: Plot the received signal power as a function of the transmitter-receiver distance between the ABS and the communication user for $\eta = 2, 2.5, 3, 3.5, 4$ (all curves in a single plot).
   - Signal power vs. transmit power: Plot the received signal power as a function of the transmit power budget for $K = 1, 2, 3, 4$ (all curves in a single plot).
   - Plot the sum throughput of a deterministic baseline UAV trajectory which takes the UAV directly from the start location to the end location during its flight time L.
   - Plot the individual throughput of a deterministic baseline UAV trajectory which takes the UAV directly from the start location to the end location during its flight time L.
   - Plot the convergence curves of the model for 2 different sets of user positions.
   - Plot the trajectories of 10 optimized UAV episodes and mark the locations where the UAV stays the longest using markers whose sizes are scaled to the corresponding dwelling times.
   - Provide bar-plots of the sum and individual throughputs of the optimized UAV and transmit signal models and compare them with:
     - **Benchmark trajectory with optimized transmit signal**: Has a deterministic flight path in which the UAV moves from the start to the end location. The transmit signal is optimized to maximize the communication performance.
     - **Benchmark trajectory with randomized transmit beamformers**: Has a deterministic flight path in which the UAV moves from the start to the end location. The transmit signal is randomly initialized.
     - **Optimized trajectory with randomized transmit beamformers**: Has the optimized path in which the UAV moves from the start to the end location using the deep reinforcement learning model. The transmit signal is randomly initialized.
7. To maintain academic integrity and transparency in your design task, you are required to:
   - Properly cite all external sources you have consulted or directly used in your work.

- If you have used any AI-based tools (e.g., ChatGPT, Copilot, Gemini, deepseek), you **MUST** clearly disclose this in your report.

3 Useful Facts

- A custom environment can be made using OpenAI's Gym library in Python. You must provide the state_space and action_space together with the reset(), step(), and render() functions. A sample environment is given at `https://gymnasium.farama.org/introduction/create_custom_env/`.
- Stable-baselines3 supplies reference implementations of modern deep reinforcement learning algorithms in Python. Startup guide is at `https://stable-baselines3.readthedocs.io/en/master/guide/install.html`.
- NumPy, short for Numerical Python, is the fundamental library for numerical and scientific computations in Python. It provides extensive support for large multi-dimensional arrays and matrices, and has an extensive collection of mathematical functions which operate on these arrays and matrices efficiently.
- Multiple-input multiple-output (MIMO) downlink systems employ beamforming to focus power in the spatial directions of intended communication users. A primer of beamforming is available at `https://ieeexplore.ieee.org/abstract/document/6832894`.
- Matplotlib is useful library for obtaining versatile visualizations.
- Real-time reward performance and training curves can be obtained using *tensorboard*. The details are available at `https://stable-baselines3.readthedocs.io/en/master/guide/tensorboard.html`.

4 Reference Materials

Though most of the required details that should be sufficient for completing this project are mentioned in this document itself, for further details, interested students are suggested to read the following:

[R1] A. Goldsmith, *Chapters 2 and 3 of Wireless communications*. Cambridge university press. 2005.

[R2] E. BjÃ¶rnson et. al., *Optimal Multiuser Transmit Beamforming: A Difficult Problem with a Simple Solution Structure [Lecture Notes]*. Weblink: `https://ieeexplore.ieee.org/abstract/document/6832894`

[R3] S. Ravichandiran *Deep Reinforcement Learning with Python* . Weblink: `https://github.com/PacktPublishing/Deep-Reinforcement-Learning-with-Python`

5 Attendance Requirements

- Student attendance in each laboratory session (Wednesday and Friday) will be **recorded**.
- In order to receive a grade for this design task, it is **mandatory** to attend all required lectures/labs, including the lab assessments.
- To ensure consistency and fairness among different simulation and hardware-focused students, all the lab-based questions will be answered or addressed only during the two three-hour weekly lab sessions.

6 Design Journal Submission

- It is mandatory to maintain a journal or report describing your design process, implementation code, and results, which must be documented via an MS Word or .pdf file.
- The design journal should include functioning and well-commented `Python` files containing your simulation which meets the requirements of the design task and must be submitted to the **Moodle Design Task X submission link**, where X is 1, 2, or 3 depending on whether you chose this task as your first, second, or third design task.
- Please ensure that the `Python` files are formatted according to the requirements listed in Section 2.3.
- Design task and your steps in obtaining them must be saved in your journal.
- Your journal should already have all the subparts required to showcase the necessary functionality, along with detailed comments explaining all the steps. You may be penalized if significant additions or changes are required to show functionality or demonstration.

- **Submission File Format:**
  - The simulation files must be compressed into a '.zip' file containing all your codes. The name of this file must be **zID_LastName_DTF_2025.zip.** Here. the students are required to provide a README.txt text file explaining the role of each included file, and the instructions on how to run your code.
  - Your simulation files should already have all the subparts required to showcase the necessary functionality along with detailed comments explaining all the steps. You may be penalized if significant additions or changes are required to show functionality as per the mentioned requirements.
  - The brief portable document format (.pdf) report summarizing the outcomes of this design task and your steps in obtaining them must be saved as a '.pdf' file (readable text file). The name of this file must be: **zID_LastName_DTF_2025.pdf**

7 Design Task Schedule and Assessment Timeline

- Each design task must be completed over a span of six lab sessions. Term 2 is structured to accommodate three design tasks to be selected by each student out of the 7 available options, with the following schedule:
  - Task 1: Week 1 Wednesday lab session to Week 3 Friday lab session,
  - Task 2: Week 4 Friday lab session to Week 7 Wednesday lab session,
  - Task 3: Week 8 Wednesday lab session to Week 10 Friday lab session.
- Each task has a specific assessment session and submission deadline:
  - If this task is completed as Task 1:
    - Assessment: Wednesday lab session of Week 4,
    - Deadline: 11:59 PM, Tuesday, June 24, 2025.
  - If this task is completed as Task 2:
    - Assessment: Friday lab session of Week 7,
    - Deadline: 11:59 PM, Thursday, July 17, 2025.
  - If this task is completed as Task 3:
    - Assessment: Wednesday lab session of Week 11,
    - Deadline: 11:59 PM, Tuesday, August 12, 2025.

8 Assessment Criteria and Marking Rubric

- The assessment breakdown for each design task, showing different components involved, is shown in Table 3.

Table 3: The assessment breakdown for Design Task F

Deliverable	Percentage of Assignment Grade
Demonstration of Correct Working Design	20%
Explanation of the Implementation and Results	40%
Ability to Answer Questions	20%
Design Journal or Notebook	20%

- The Correct Working Design component, worth 20%, requires that **all objectives are accurately met** according to the details given in Section 2.3.
- The Explanation of Implementation and Results is evaluated based on the following four aspects:
  - **Clarity and explanation of plots** - how well the plots are presented and interpreted.
  - **Code efficiency and commenting** - the quality, readability, and documentation of the code.
  - **Working code and its explanation** - whether the code runs correctly and is clearly explained.
  - **Explanation of Results** - how well the outcomes are analysed and related to the design objectives.
- During the assessment, you will also need to demonstrate the functionality of the `Python` code, explain your design journal and your design to the lab demonstrator and answer any questions they may have about your design.

- The "Ability to Answer Questions" component is based on the quality of responses to the **two to four questions** based on the design task.
- The Design Journal component is evaluated based on four equally weighted areas. First, students must provide a brief **problem description** that clearly outlines the design task. Second, they should present their **understanding and logic**, explaining their approach and reasoning in a structured and coherent manner. Third, the journal should include a **demonstration of results**, showcasing the outcomes of their design work with appropriate interpretation. Finally, students must include **project management** details, such as weekly progress updates, a declaration of any AI tools used, and proper referencing of all sources.
- Overall, during the assessment, students will need to demonstrate the functionality of the `MATLAB` code, explain their design journal and design to the lab demonstrator, and answer any questions they may have about their design.
- All design tasks in this course **MUST** be completed individually. Copying from others is not permitted and may result in a **failing (UF) grade** [5], reinforcing the importance of academic integrity.
- Please note that design tasks must be submitted by the set deadline, and extensions will not be granted, except in cases of approved special consideration, which may allow students to redo the task during Weeks 11 and 12.
- **Passing Requirement and Hurdle**: To pass this course, you must:
  - Achieve an overall course mark of 50% or higher, where this overall mark is calculated as follows:
    - 30% from each of the three design tasks (totaling 90%),
    - 10% from the lab exam, which will be conducted during the Friday lab session of Week 11, and
  - Pass at least two out of your three design tasks. This means you can fail only one design task.
- **Redo Opportunity**: If you fail a design task, you will have a chance to redo it during Weeks 11 and 12.
  - Lab sessions for redoing tasks will run from 9:00 AM to 3:00 PM on:
    - 6-hour lab session on Wednesday of Week 11
    - 6-hour lab sessions on Wednesday and Friday of Week 12
  - If your resubmitted task meets the requirements for a satisfactory grade, you will receive a mark of 50% for that task.
  - The grading and assessment will be conducted on the Friday lab session of Week 12 after the student submits their task during the same session.
  - Please be aware that you are allowed to **redo ONLY one design task**.
- **Supplementary Lab Sessions**: The above-mentioned three 6-hour lab sessions in Weeks 11 and 12 can also be used as make-up sessions for students who missed earlier labs due to approved special consideration. In such cases, students will be assessed normally and will receive their actual earned mark, rather than the fixed 50% awarded for task redos. Please be aware that students are strictly permitted to complete only one design task during the supplementary lab sessions in Weeks 11 and 12. No exceptions will be made beyond this single opportunity.
- If needed, students may be given access to open lab sessions to work outside regular lab hours. However, lecturer and demonstrator support will only be available during the two scheduled weekly lab sessions.
- **Formal feedback** will be provided well within two weeks of the relevant submission date through Moodle, followed by a brief discussion during the lab session.
- **Note**: Use of AI tools (such as ChatGPT or Copilot) for your learning in this course is allowed, but you are responsible for everything you produce - designs, reports, graphs, etc. (keeping in mind that the output from AI tools can be incorrect). Any use of AI tools must be declared.

[5] https://www.student.unsw.edu.au/grade