School of Computing 2025/26

Module Code and Title	M33174 Artificial Intelligence
Module Coordinator Other lecturers	Dr. ioannis kagalidis <ioannis.kagalidis@port.ac.uk>
Assessment Item number	Item 1
Assessment Title	Coursework
Date Issued	2025-10-16

Deliverables

Deliverable	Weight	Format	Deadline / Date	Late deadline ECF deadline
Code and associated files	50%	Code and text files.	2025-11-24 1pm (for Moodle or Wiseflow coursework submissions add the following) with submissions permitted until 2025-11-26 1pm [GMT/BST] (48 hours after deadline)	2025-12-5 1pm (10 working days after deadline)

Notes and Advice

- The Extenuating Circumstances procedure is there to support you if you have had any circumstances (problems) that have been serious or significant enough to prevent you from attending, completing or submitting an assessment on time. If you complete an Extenuating Circumstances Form (ECF) for this assessment, it is important that you use the correct module code, item number and deadline (not the late deadline) given above.
- ASDAC are available to any students who disclose a disability or require additional support for their academic studies with a good set of resources on the ASDAC moodle site
- The University takes any form of misconduct (such as plagiarism or cheating) seriously, so please make sure your work is your own. Please ensure you adhere to our *Student Conduct Policy* and watch the video on Plagiarism.
- Any material included in your coursework should be fully cited and referenced in **APA 7** format. Detailed advice on referencing is available from the library, also see TECFAC 08 Plagiarism.

- Any material submitted that does not meet format or submission guidelines, or falls outside of the submission deadline could be subject to a cap on your overall result or disqualification entirely.
- If you need additional assistance, you can ask your personal tutor, student success advisor ana.baker@port.ac.uk, academic tutors simon.jones@port.ac.uk & eleni.noussi@port.ac.uk or your lecturers.
- If you are concerned about your mental well-being, please contact our Well-being service.

PLEASE READ:

"*We require that most work submitted for assessment is your own original content, demonstrating your knowledge, skills, and critical thinking abilities.*
***The University's position on the use of AI is that it is permitted as a tool to assist and inform research and the generation of ideas, planning, and output. The use of AI in submitted work must be underpinned by the principles of academic integrity, proper citation, and referencing, with clear indication given as to where AI has been utilised in all submissions. Failure to do so will be considered an act of academic misconduct.*"**

Module:M33174 Artificial Intelligence.

Coursework Title: *Using GAs and Neural Networks for Solving the Spider Kinematics Problem.*
Group coursework. Students should include an agreed upon contribution sheet with their agreed contribution to the work done.

Provided

A MATLAB function that simulates a 3D model of a spider with 8 articulated legs, each leg composed of 3 rotational joints. The function accepts a **1x24 vector** as input, where each set of 3 consecutive values represents the joint angles for one leg, ordered from the innermost joint to the outermost (e.g., coxa, femur, tibia). The legs are labelled in a specific order for symmetry and animation consistency: **[L1, L2, L3, L4, R4, R3, R2, R1]**, starting from the spider's front-left leg and circling around to the front-right. Therefore, the input vector is structured as follows:
[L1a, L1b, L1c, L2a, L2b, L2c, L3a, L3b, L3c, L4a, L4b, L4c, R4a, R4b, R4c, R3a, R3b, R3c, R2a, R2b, R2c, R1a, R1b, R1c]. **The joint angles are in radians.**
This format ensures a consistent mapping between the input angles and the spider's kinematics. The code can be found on the module Moodle page.

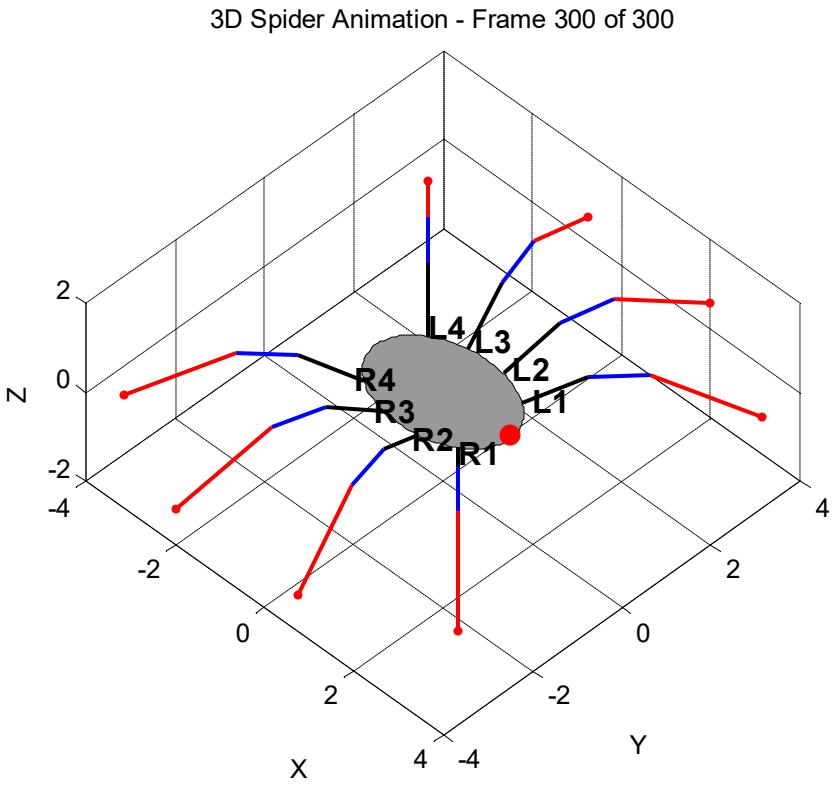

Figure 1. Spider 3D model Plot.

*Part 1 of 2: Chromosome Optimization using Genetic Algorithms

Total Marks (Part 1): 100 marks

Submission Format: Code files (with inline comments) **OR** Code + explanation in accompanying .txt files
Deadline: [24/11/2025 by 13:00]
Weighting: Part 1 contributes **50%** of the total coursework grade (200 total marks)

Coursework Overview

In this coursework, students will apply **Genetic Algorithms (GAs)** to a **simplified 3D model of a spider**. The spider has **8 legs**, each with **3 joints**, resulting in **24 degrees of freedom** (i.e., 24 joint angles). The aim is to use a GA to find a valid chromosome (a potential solution) that represents a suitable configuration of these 24 angles. Although the testing Matlab function supports only one 1x24 vector, students are required to produce a full gait (=pattern of walking). This would take about 300 1x24 vectors to create a full gait. Please see the video on Moodle.

While a basic 3D model of the spider is provided in **MATLAB**, students may use **any programming language** of their choice to implement the GA and visualize the results.

This first part of the coursework focuses on implementing a **complete and functioning GA** that can evolve solutions to this high-dimensional problem.

Objectives

By completing this coursework, students should demonstrate the ability to:

- Encode a multi-dimensional joint configuration as a chromosome
- Design and implement a working GA including initialization, selection, crossover, and mutation
- Create a fitness function suitable for guiding spider pose optimization
- Develop visualization tools to assess GA performance
- Evaluate and explain the effectiveness of different GA components

Deliverables

1. **Code Submission:** Source code with either:
    - Inline documentation / detailed comments, or
    - Separate .txt explanation files per component (e.g., chromosome encoding, fitness function)
2. **User Interface Tools:** Any plots, animations, or visual indicators that help evaluate whether the GA is working (e.g., fitness over generations, spider pose previews)

3. (Optional but encouraged) A README file explaining how to run the code and interpret outputs.

Marking Rubric (Part 1 – Genetic Algorithm) – 100 Marks

Category	Description	Marks
1. Chromosome Encoding	Encoding of the 24 joint angles in a way that is compatible with crossover and mutation operations. Efficient and logically structured.	**15**
2. Fitness Function	Fitness function designed to meaningfully reflect the goals of the problem (e.g., stability, realistic movement, symmetry, target pose, etc.). Should be clearly explained.	**15**
3. Genetic Operators	**Selection method**: Correct and justified choice (e.g., roulette wheel, tournament)**Crossover implementation**: Correctly applied and explained (e.g., 1-point, n-point, uniform)**Mutation method**: Clearly implemented and parameterized	**30**
4. Algorithm Design	GA logic is complete (initiation, loop, termination). Parameters (population size, mutation rate, etc.) are tunable.	**10**
5. Performance Visualization	Graphs showing fitness progression Plots or 3D renderings of spider poses Other relevant tools (e.g., console logs, animations)	**10**
6. Explanation and Justification	Clear and reasoned explanations of all design decisions, either via comments or accompanying .txt files. Should demonstrate understanding of trade-offs.	**10**
7. Code Quality and Execution	Code should run without errors, be well-structured, and follow good programming practices.	**10**

Total: 100 Marks

Suggested Learning Resources

- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*
- MATLAB documentation on GA and optimization
- Python libraries (e.g., DEAP, PyGAD) if Python is used
- Matlab 3D spider model available on Module Moodle page.

Notes

- You **do not** need to model the full physics of spider movement, but your fitness function should guide the joint angles toward plausible, stable, or goal-directed poses.
- You are free to experiment with hybrid techniques or enhancements (e.g., elitism, adaptive mutation).
- You may use plotting libraries to visualize the evolution of fitness, the configuration of the spider, or other aspects of your GA.

Part 2 of 2: Predicting Joint Angles Using Neural Networks

Total Marks (Part 2): 100 marks
Submission Format: Code files (with inline comments) **OR** Code + explanation in accompanying .txt files
Deadline: [24/11/2025 by 13:00]
Weighting: Part 2 contributes **50%** of the total coursework grade (200 total marks)

Coursework Overview

In this second part of the coursework, students will design and implement a **Neural Network (NN)** solution to produce joint angle configurations for the simplified 3D spider model (8 legs × 3 joints = **24 joint angles**).

Unlike Part 1, where a **Genetic Algorithm** was used to evolve joint configurations, students will now create and train a Neural Network to output valid and effective sets of joint angles, given an appropriate set of inputs (to be designed by the student or produced by your GA in part 1).

Students must make informed decisions about the structure of their network(s), including:

- Number of **layers** and **nodes**
- Choice of **activation function(s)**
- Use of **loss function(s)** and **backpropagation**
- Format of **input and output vectors**
- Training data generation or usage (can be real, simulated, or generated via GA solutions from Part 1)

Objectives

By completing this coursework, students should demonstrate the ability to:

- Design appropriate neural network architectures for high-dimensional output problems
- Choose and justify key hyperparameters (e.g., learning rate, architecture, activation functions)
- Train a neural network using appropriate data and learning strategies
- Evaluate and visualize the effectiveness of the NN in generating valid joint angles

- Reflect critically on NN performance and suggest improvements

Deliverables

1. **Code Submission:** Source code for the Neural Network implementation, with:
    - Detailed inline documentation/comments, or
    - Accompanying .txt files explaining the architecture, reasoning, and methodology
2. **Neural Network Details:** (Must include:)
    - Number of layers and neurons per layer
    - Activation functions
    - Learning rate
    - Loss function used
    - Training method (e.g., stochastic gradient descent, Adam, etc.)
    - Training dataset and how it was generated/processed
    - A saved model of your final successful network for testing.
3. **User Interface Tools:** Visualizations or plots that demonstrate:
    - Training progress (e.g., loss over epochs)
    - Examples of output poses (joint angles or rendered spider poses)
    - Comparison of predicted outputs with expected or target configurations
4. **(Optional but recommended):** A README explaining how to run your code and interpret results.

Marking Rubric (Part 2 – Neural Network Solution) – 100 Marks

Category	Description	Marks
1. Neural Network Architecture	Clear and justified choice of NN structure (layers, neurons, depth). Includes rationale for input/output shape. Architecture is appropriate for the problem's complexity.	**20**
2. Activation Functions and Loss Function	Appropriate use of activation functions (e.g., ReLU, tanh, sigmoid, etc.) and a suitable loss function (e.g., MSE). Clear justification of choices.	**15**
3. Training Method & Learning Rate	Correct implementation of training strategy (e.g., batch size, optimizer, learning rate). Learning rate is appropriately tuned and explained.	**15**
4. Backpropagation and Convergence	Evidence of correct backpropagation and learning. NN should converge toward a reasonable solution. May use libraries (e.g., PyTorch, TensorFlow), but must understand and explain the process.	**15**
5. Data Handling & Input/Output Format	Thoughtful approach to designing input data and target outputs. If training data is self-generated (e.g., from GA results), its construction should be explained.	**10**
6. Performance Visualization	Training loss plotsSample outputs of joint angles3D renderings or plots showing pose quality (optional but rewarded)	**10**
7. Explanation and Justification	Clear, well-structured explanations of design choices and methodology, in comments or separate .txt files. Shows understanding of trade-offs.	**10**
8. Comparison with available libraries	You may use libraries (e.g., PyTorch, TensorFlow) for comparison or checking, but must understand and explain the process and compare them with the code and models you developed from scratch. Libraries do not contribute to marks in the marking scheme from 1 to 7.	**5**

Total: 100 Marks

Suggested Learning Resources

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (MIT Press)
- Python libraries:
    - **PyTorch** (https://pytorch.org)
    - **TensorFlow/Keras** (https://www.tensorflow.org)
- Online tools like Google Colab for cloud-based training
- Visualizing loss and learning:
    - matplotlib, seaborn, or NN-specific tools like **TensorBoard**

Notes

- You may use neural network libraries and frameworks, but must understand the underlying processes and the do not count for marks in categories from 1 to 7.
- Creativity in input design is encouraged. For example, inputs could represent a goal position, time step, terrain, or previous poses.
- You are welcome to combine insights or outputs from Part 1 as training data or benchmarks.