Verification and Validation Report: TPG

Team 3, Tangle
Calvyn Siong
Cyruss Allen Amante
Edward Gao
Richard Li
Mark Angelo Cruz

March 1, 2025

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

symbol	description
TPG	Tangled Program Graphs
DNNs	Deep Neural Networks
RL	Reinforcement Learning
SRS	Software Requirement Specification
FR	Functional Requirement
NFR	Non-Functional Requirement
SLN	System Level Number
VnV	Verification and Validation

Contents

List of Tables

List of Figures

This document cohesively summarizes the results of each test as specified in the VnV Plan documentation.

3 Functional Requirements Evaluation

3.1 MuJoCo Integration

Table 1: MuJoCo Integration Tests

Test Id	**Notes**	**Result**
FR-SLN1	When executing the appropriate script, all MuJoCo environments can be run. The best-performing agent within the policy can be visualized using OpenGL or an MP4 file.	Pass
FR-SLN2	MuJoCo environments within the TPG framework can be successfully run within the Digital Research Alliance, enabling research to be conducted by executing experiments.	Pass

3.2 Experiment Visualization

Table 2: Experiment Visualization Tests

Test Id	**Notes**	**Result**
FR-SLN3	When an experiment is running or finished training, the best performing policy can be visualized using the TPG CLI tool.	Pass

3.3 Github Actions CI/CD Pipeline

3.4 Software Engineering Practices

Table 3: Software Engineering Practices Tests

Test Id	**Notes**	**Result**
FR-SLN6	Newly added code in the TPG codebase follows Google's C++ Style Guide and software engineering best practices such as design patterns, and object-oriented design. This includes careful review and consideration of code readability, extendability, maintainability and scalability. A linter has also been implemented to check for such styling as discussed in 3.3.	Pass

4 Nonfunctional Requirements Evaluation

4.1 Usability

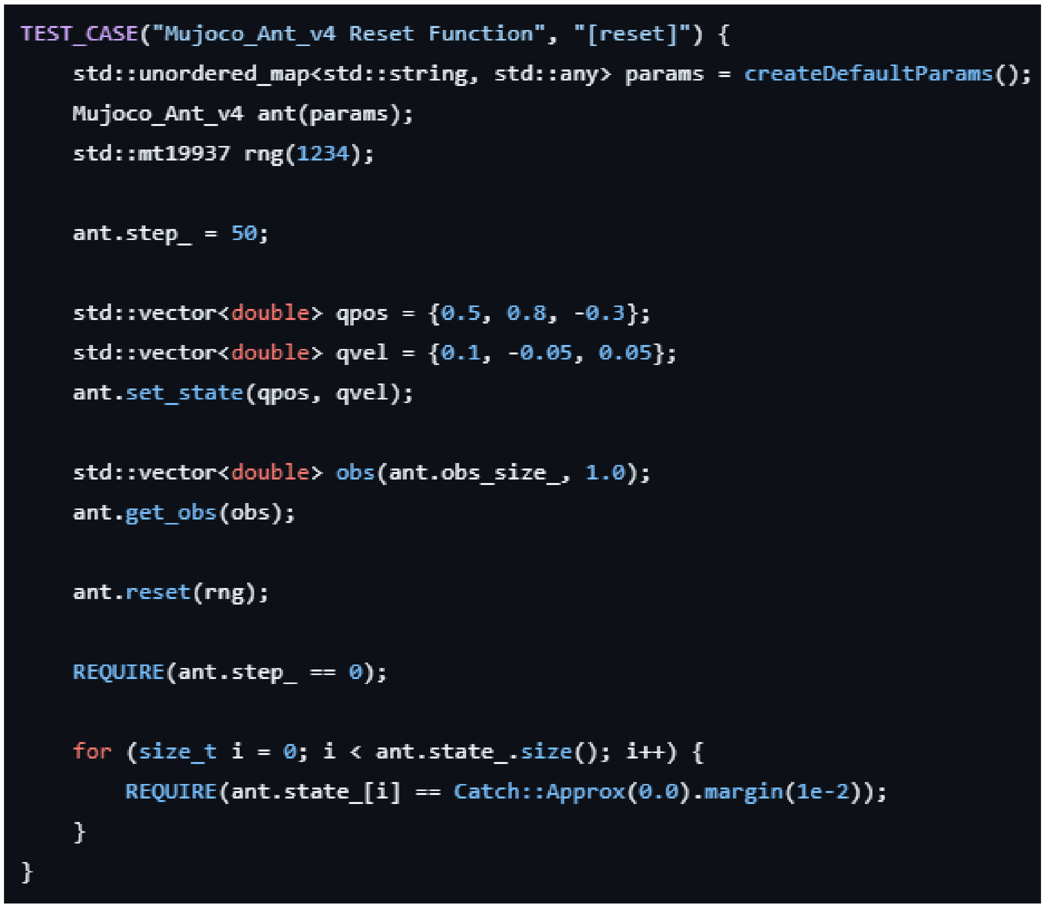
4.2 Performance

Table 4: Performance Tests

Test Id	**Result**
NFR-SLR4	Pass

For NFR-SLN4, test cases within TPG for the experimental environments have been implemented to check for the accuracy of the numerical computations associated during training. Declaration of variables with proper types (e.g. signed long or int, unsigned long or int) has also been taken into consideration to reduce issues in the future for extremely large or small numbers that may overflow. TPG has been comprehensively tested to guarantee that all computations with high numerical precision (e.g. during the runtime of an experiment) are accurate and contain an acceptable tolerance limit of

0.00001. The results were inspected manually by comparing the actual output to the anticipated output, and performing a calculation to check for quantitative error, and if such error meets the requirements for numerical precision.



```cpp
TEST_CASE("Mujoco_Ant_v4 Reset Function", "[reset]") {
    std::unordered_map<std::string, std::any> params = createDefaultParams();
    Mujoco_Ant_v4 ant(params);
    std::mt19937 rng(1234);

    ant.step_ = 50;

    std::vector<double> qpos = {0.5, 0.8, -0.3};
    std::vector<double> qvel = {0.1, -0.05, 0.05};
    ant.set_state(qpos, qvel);

    std::vector<double> obs(ant.obs_size_, 1.0);
    ant.get_obs(obs);

    ant.reset(rng);

    REQUIRE(ant.step_ == 0);

    for (size_t i = 0; i < ant.state_.size(); i++) {
        REQUIRE(ant.state_[i] == Catch::Approx(0.0).margin(1e-2));
    }
}
```

Figure 1: Example of a Numerical Computation Test

4.3 Operational and Environmental

Table 5: Operational and Environmental Tests

Test Id	**Result**
NFR-SLR6	Pass

For NFR-SLN6, TPG now supports contributions from macOS, Windows, and Linux developers. Previously, only Linux was supported because TPG used SCons for C++ builds and Linux-specific dependencies from requirements.txt. With VSCode Dev Containers, a Linux development environment is automatically launched for all developers, ensuring a standardized setup. Simply follow the Wiki instructions to download all necessary Linux dependencies and build the C++ code reliably. Onboarding on a Macbook has been reduced from 2 weeks to just 5 minutes.

4.4 Maintainability

4.5 Security

Table 6: Security Tests

Test Id	**Result**
NFR-SLR8	Pass

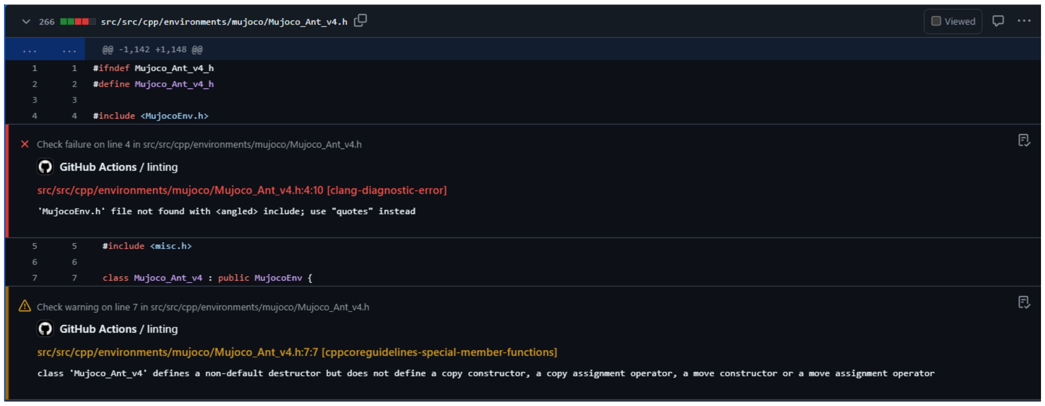
For NFR-SLN8, the .csv, .txt, .png and .mp4 files that are generated within Classic Control and MuJoCo experiments are ignored by Git when making commits to the public repositories in GitHub and GitLab to reduce chance of oversharing sensitive data. Currently, none of these files generate sensitive data, but to follow best practice and to keep the repository at a clean state, these are not recognized when synchronizing code to each respective repository. Additionally, the team has also manually checked all stored .csv, .txt, .png and .mp4 files along with others that may contain textual information to see if data within them are sensitive and must be kept private.

4.6 Compliance

Table 7: Compliance Tests

Test Id	Result
NFR-SLR9	Pass

The modified codebase is successfully analyzed using Clang-Tidy and Clang-Format within the CI/CD pipeline. Code change discussions take place through pull request conversations made to the main branch. All errors and warnings are generated based on the C++ Style Guidelines. Any critical errors found during the linting process create blocking pull request conversations that must be resolved before merging into the main branch.



```
266 src/src/cpp/environments/mujoco/Mujoco_Ant_v4.h
@@ -1,142 +1,148 @@
#ifndef Mujoco_Ant_v4_h
#define Mujoco_Ant_v4_h

#include <MujocoEnv.h>

Check failure on line 4 in src/src/cpp/environments/mujoco/Mujoco_Ant_v4.h
GitHub Actions / linting
src/src/cpp/environments/mujoco/Mujoco_Ant_v4.h:4:10 [clang-diagnostic-error]
'MujocoEnv.h' file not found with <angled> include; use "quotes" instead

 #include <misc.h>

 class Mujoco_Ant_v4 : public MujocoEnv {

Check warning on line 7 in src/src/cpp/environments/mujoco/Mujoco_Ant_v4.h
GitHub Actions / linting
src/src/cpp/environments/mujoco/Mujoco_Ant_v4.h:7:7 [cppcoreguidelines-special-member-functions]
class 'Mujoco_Ant_v4' defines a non-default destructor but does not define a copy constructor, a copy assignment operator, a move constructor or a move assignment operator
```

Figure 2: Example of a Linter Error

5 Unit Testing

6 Changes Due to Testing

6.1 Feedback from Rev 0

The feedback given by the instructor and teaching assistant during Revision 0 was essential in guiding the next steps as the team looks toward the final demonstration. Emphasis was placed on ensuring that usability testing was executed systematically rather than in the more ad-hoc manner initially planned by the team. Some additional changes to be made include ensuring that unit testing and benchmarking of the implemented environments are cohesively executed and investigating whether the integration of deployment within the DRA is possible.

7 Automated Testing

As a result of the team's conversion from building the project using SCons to CMake, automated testing became significantly easier to execute and debug. To run any automated tests within a developer's local environment, a developer can simply execute a command to build the project. This not only compiles everything but also runs all automated tests. If a developer wishes to run only the tests, they must navigate to the directory where the tests were already compiled (typically `/build/tests`). From there, the command `ctest` can be entered into the command prompt. Similar to the compilation process, all automated tests are executed once this command is run.

From the repository's point of view, tests are executed using GitHub Actions or GitLab CI (depending on which repository is being viewed). Both linting and compilation are performed using the same commands that would be executed within a developer's local environment. These tests run when a new pull request is made to the main branch, ensuring that all tests pass before merging. The compiler workflow is also executed after merging into the main branch to ensure no errors or unintended changes in code behaviour have occurred. If any test or workflow fails, the logs of the workflow can be reviewed, providing a detailed summary of the reason for failure. This not only allows for easier debugging but also resolves the "works on my machine" issue.

8 Trace to Requirements

9 Trace to Modules

10 Code Coverage Metrics

References

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?
4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)