# ECE 322 Lab 2 Report

By Mattheas Jamieson

## Introduction

The objective of this lab was to apply more formal Black Box testing techniques as compared to lab 1 with the goal of discovering errors in the provided program and comparing the efficacy of different techniques. Black Box testing occurs when internal implementation is unknown to the tester as opposed to White Box testing where the implementation details are given. The testing methods used in are Extreme Point Combination (EPC) & Weak nx1 strategy. Both testing methods were used for each part of the lab. EPC is a testing method that tests around the subdomain limits in all dimensions. The tester selects test points on the min/ max of the subdomain in each dimension as well as slightly below min/ slightly above max. All possible combinations for all dimensions are tested so that a total of $4^n + 1$ test cases are developed. EPC is only guaranteed to find a closure problem or missing boundary. Weak nx1 strategy is used on linear boundaries where n ON points are selected that are located on the boundary to identify the boundary while one OFF point is selected that is off the boundary, with this method any sort of closure problem, missing boundary or boundary shift/ tilt should be identified. Weak nx1 strategy yields $b(n + 1) + 1$ test cases.

## Part One – Drone Program

**Q1** : Provide an explanation of the application under test, they should describe the problem that is being tested :

The program under test GPS enabled drone program. There is a maximum distance $k$ that the drone can travel before it must turn back or else it will not have enough fuel for the return trip. Along the trip the drone can set a maximum of 3 waypoints, labelled $x_1$, $x_2$ $and$ $x_3$. The constraint is that the sum of these waypoint distances must be less than or equal to $k$. The testing performed is to ensure that program will correctly inform us if the flight is viable or not given a set of waypoints. In our testing it is assumed that $k$ is hardcoded to a value of 100.

**Q2**: Explain what testing methods you are using. The testing methods should be explained, what do they test, what are they good for etc ..
Errors in the application should be identified, and some justification given for what may be causing these errors. :

The two Black Box testing methods used are EPC and Weak nx1 strategy. EPC uses extreme points from every dimension of the subdomain in all possible combinations. It is good for

testing subdomains that are rectangular in shape, and can always identify closure problems or missing boundaries. Weak nx1 strategy uses n points located on a boundary to identify it and one point off the boundary for testing. This generally leads to fewer test cases than EPC testing, however Weak nx1 can identify tilted/ shifted/ missing boundaries and closure problems.

Using an EPC testing strategy, a single error was uncovered as seen in test cases **18**, **19**, **20**, **34**, **35**, **36**, **50**, **51** and **52**. Looking at all the test cases it can be clearly seen that constraint on $x_2$ is not properly enforced, i.e., it can take on a negative value. A practical interpretation of this constraint is that you cannot set a waypoint to have a negative distance. Based on the correct evaluation of the other test cases it can be summarized that $x_2$ has either a missing or shifted boundary for its lower limit only. I believe further testing in the form of error guessing would be needed to determine exactly what kind of boundary problem $x_2$ has. $x_2$ can take some unknown range of negative values, meaning in the code the lower constraint was either not performed or was performed with a number that was less than or equal to -1.

Using the Weak nx1 strategy the same error was found in only one test case, i.e. test case **8**.

**Q3**: Discuss the effectiveness of the testing methods used

I believe both testing methods were effective, however because there was a relatively few number of linear boundaries the Weak nx1 strategy was more efficient simply because it found the same error with approximately one quarter of the test cases. I do believe however that EPC gives a better starting point if an error guessing approach was taken afterwards to pinpoint the error because with EPC there are nine failed test cases to assist you. In terms of effort if a automated script is used there is not much difference between the two testing types. If however the test cases must be typed out manually then Weak nx1 would have an advantage.

**Q4** : Test Cases Table Extreme Point Combination(EPC):
**Answer**: Tests 1-65

| ID | Inputs | Description | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | -1,-1,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 2 | -1,-1,0 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 3 | -1,-1,100 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |

| | | | | |
|---|---|---|---|---|
| 4 | -1,-1,101 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 5 | -1,0,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 6 | -1,0,0 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 7 | -1,0,100 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 8 | -1,0,101 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 9 | -1,100,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 10 | -1,100,0 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 11 | -1,100,100 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 12 | -1,100,101 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 13 | -1,101,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 14 | -1,101,0 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 15 | -1, 101,100 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 16 | -1, 101,101 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |

| | | | | |
|---|---|---|---|---|
| 17 | 0,-1,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| **18** | 0,-1,0 | | ERROR: Invald argument – negative value | Success |
| **19** | 0,-1,100 | | ERROR: Invald argument – negative value | Success |
| **20** | 0,-1,101 | | ERROR: Invald argument – negative value | Success |
| 21 | 0,0,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 22 | 0,0,0 | | Success | Success |
| 23 | 0,0,100 | | Success | Success |
| 24 | 0,0,101 | | Failure | Failure |
| 25 | 0,100,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 26 | 0,100,0 | | Success | Success |
| 27 | 0,100,100 | | Failure | Failure |
| 28 | 0,100,101 | | Failure | Failure |
| 29 | 0,101,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 30 | 0,101,0 | | Failure | Failure |
| 31 | 0, 101,100 | | Failure | Failure |
| 32 | 0, 101,101 | | Failure | Failure |
| 33 | 100,-1,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| **34** | 100,-1,0 | | ERROR: Invald argument – negative value | Success |
| **35** | 100,-1,100 | | ERROR: Invald argument – negative value | Failure |
| **36** | 100,-1,101 | | ERROR: Invald argument – negative value | Failure |

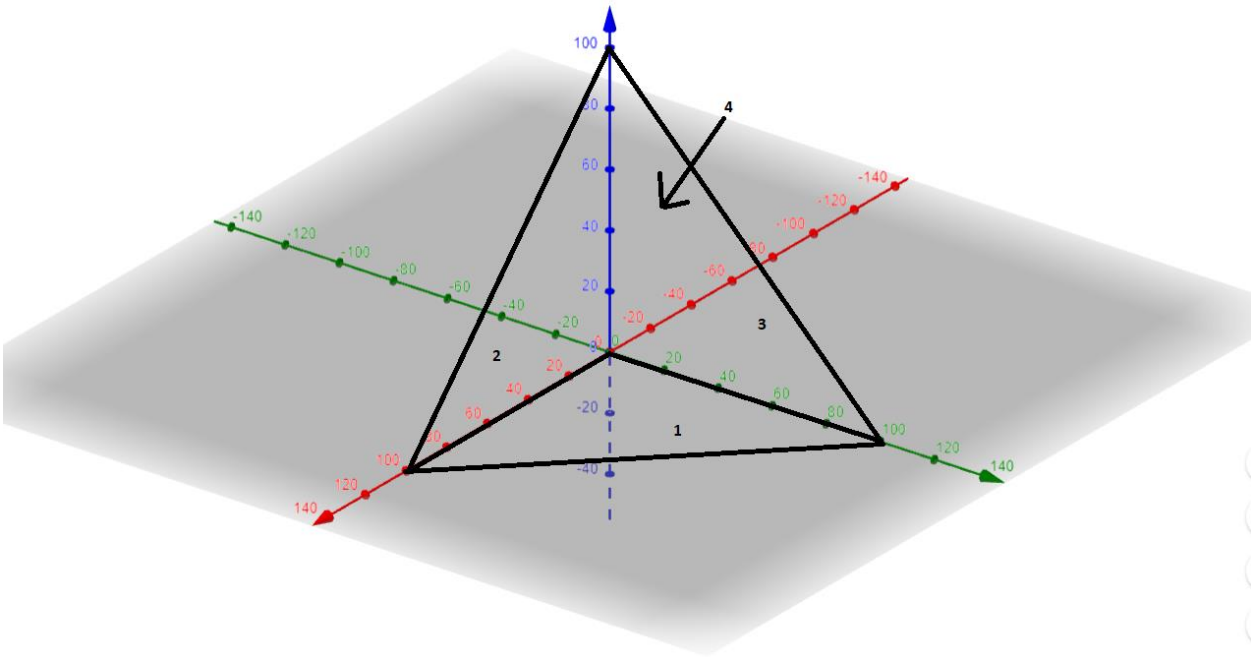
| | | | | |
|---|---|---|---|---|
| 37 | 100,0,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 38 | 100,0,0 | | Success | Success |
| 39 | 100,0,100 | | Failure | Failure |
| 40 | 100,0,101 | | Failure | Failure |
| 41 | 100,100,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 42 | 100,100,0 | | Failure | Failure |
| 43 | 100,100,100 | | Failure | Failure |
| 44 | 100,100,101 | | Failure | Failure |
| 45 | 100,101,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 46 | 100,101,0 | | Failure | Failure |
| 47 | 100, 101,100 | | Failure | Failure |
| 48 | 100, 101,101 | | Failure | Failure |
| 49 | 101,-1,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| **50** | 101,-1,0 | | ERROR: Invald argument – negative value | Success |
| **51** | 101,-1,100 | | ERROR: Invald argument – negative value | Failure |
| **52** | 101,-1,101 | | ERROR: Invald argument – negative value | Failure |
| 53 | 101,0,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 54 | 101,0,0 | | Failure | Failure |
| 55 | 101,0,100 | | Failure | Failure |
| 56 | 101,0,101 | | Failure | Failure |
| 57 | 101,100,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 58 | 101,100,0 | | Failure | Failure |
| 59 | 101,100,100 | | Failure | Failure |
| 60 | 101,100,101 | | Failure | Failure |

| | | | | |
|---|---|---|---|---|
| 61 | 101,101,-1 | | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 62 | 101,101,0 | | Failure | Failure |
| 63 | 101, 101,100 | | Failure | Failure |
| 64 | 101, 101,101 | | Failure | Failure |
| 65 | (10,10,10) | | Success | Success |

**Q5** : Test Cases Table Weak nx1 Strategy:
**Answer**: Tests 1-17

| ID | Inputs | Description | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | 1,1,0 | Plane 1 | Success | Success |
| 2 | 5,5,0 | Plane 1 | Success | Success |
| 3 | 10,10,0 | Plane 1 | Success | Success |
| 4 | 4,4,-1 | Not on plane 1 | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 5 | 1,0,1 | Plane 2 | Success | Success |
| 6 | 10,0,10 | Plane 2 | Success | Success |
| 7 | 25,0,25 | Plane 2 | Success | Success |
| **8** | 3,-1,3 | Not on plane 2 | ERROR: Invald argument – negative value | Success |
| 9 | 0,1,1 | Plane 3 | Success | Success |
| 10 | 0,20,20 | Plane 3 | Success | Success |
| 11 | 0,30,40, | Plane 3 | Success | Success |
| 12 | -1,20,20 | Not on plane 3 | ERROR: Invald argument – negative value | ERROR: Invald argument – negative value |
| 13 | 25,25,50 | Plane 4 (x+y+z=100) | Success | Success |
| 14 | 0,50,50 | Plane 4 | Success | Success |
| 15 | 10,10,80 | Plane 4 | Success | Success |
| 16 | 70,20,11 | Not on plane 4 | Failure | Failure |
| 17 | 20,20,20 | Inside subdomain | Success | Success |

**Q6** : Subdomain Diagram (w/ planes labelled 1 through 4)




## Part 2 – Remote Car Program

**Q1**: explanation of the application under test, and a description of the problem

The remote car program specifies a circle with a radius $r$ as the range that a remote car can travel inside while connected to the remote control, if the car travels outside this radius then the signal becomes to weak to control the car. This subdomain is approximated by 4 linear segments where the assumption is that this approximation is the real domain for testing purposes.

The formulas for the approximated linear boundaries are:

y=x+1
y=-x+1
y=-x-1
y=x-1

where x must be in the range [0,1]

**Q2**: Test Cases Table Extreme Point Combination(EPC):

**Answer**: Tests 1-17

| ID | Inputs | Description | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | -1.1,-1.1 | | Out of range | Out of range |
| 2 | -1.1,-1 | | Out of range | Out of range |
| 3 | -1.1,1 | | Out of range | Out of range |
| 4 | -1.1,1.1 | | Out of range | Out of range |
| 5 | -1,-1.1 | | Out of range | Out of range |
| 6 | -1,-1 | | Out of range | Out of range |
| 7 | -1,1 | | Out of range | Out of range |
| 8 | -1,1.1 | | Out of range | Out of range |
| 9 | 1,-1.1 | | Out of range | Out of range |
| 10 | 1,-1 | | Out of range | Out of range |
| 11 | 1,1 | | Out of range | Out of range |
| 12 | 1,1.1 | | Out of range | Out of range |
| 13 | 1.1,-1.1 | | Out of range | Out of range |
| 14 | 1.1,-1 | | Out of range | Out of range |
| 15 | 1.1,1 | | Out of range | Out of range |
| 16 | 1.1,1.1 | | Out of range | Out of range |
| 17 | 0.5,0.5 | | Ok | Ok |

**Q3** : Test Cases Table Weak nx1 Strategy:

**Answer**: Tests 1-13

| ID | Inputs | Description | Expected Result | Actual Result |
|---|---|---|---|---|
| 1 | -0.75,0.25 | On line y=x+1 | success | Ok |
| 2 | -0.25,0.75 | On line y=x+1 | success | Ok |
| 3 | -0.5,0.6 | Not On line y=x+1 | Out of range | Ok |
| 4 | 0.75,0.25 | On line y=-x+1 | success | Ok |
| 5 | 0.25,0.75 | On line y=-x+1 | success | Ok |
| 6 | 0.5,0.6 | Not On line y=-x+1 | Out of range | Ok |
| 7 | 0.75,-0.25 | On line y=x-1 | success | Ok |
| 8 | 0.25,-0.75 | On line y=x-1 | success | Ok |
| 9 | 0.5,-0.6 | Not On line y=x-1 | Out of range | Ok |
| 10 | -0.75,-0.25 | On line y=-x-1 | success | Ok |
| 11 | -0.25,-0.75 | On line y=-x-1 | success | Ok |
| 12 | -0.5,-0.6 | Not On line y=-x-1 | Out of range | Ok |
| 13 | 0.25,0.25 | Inside Boundary | success | Ok |

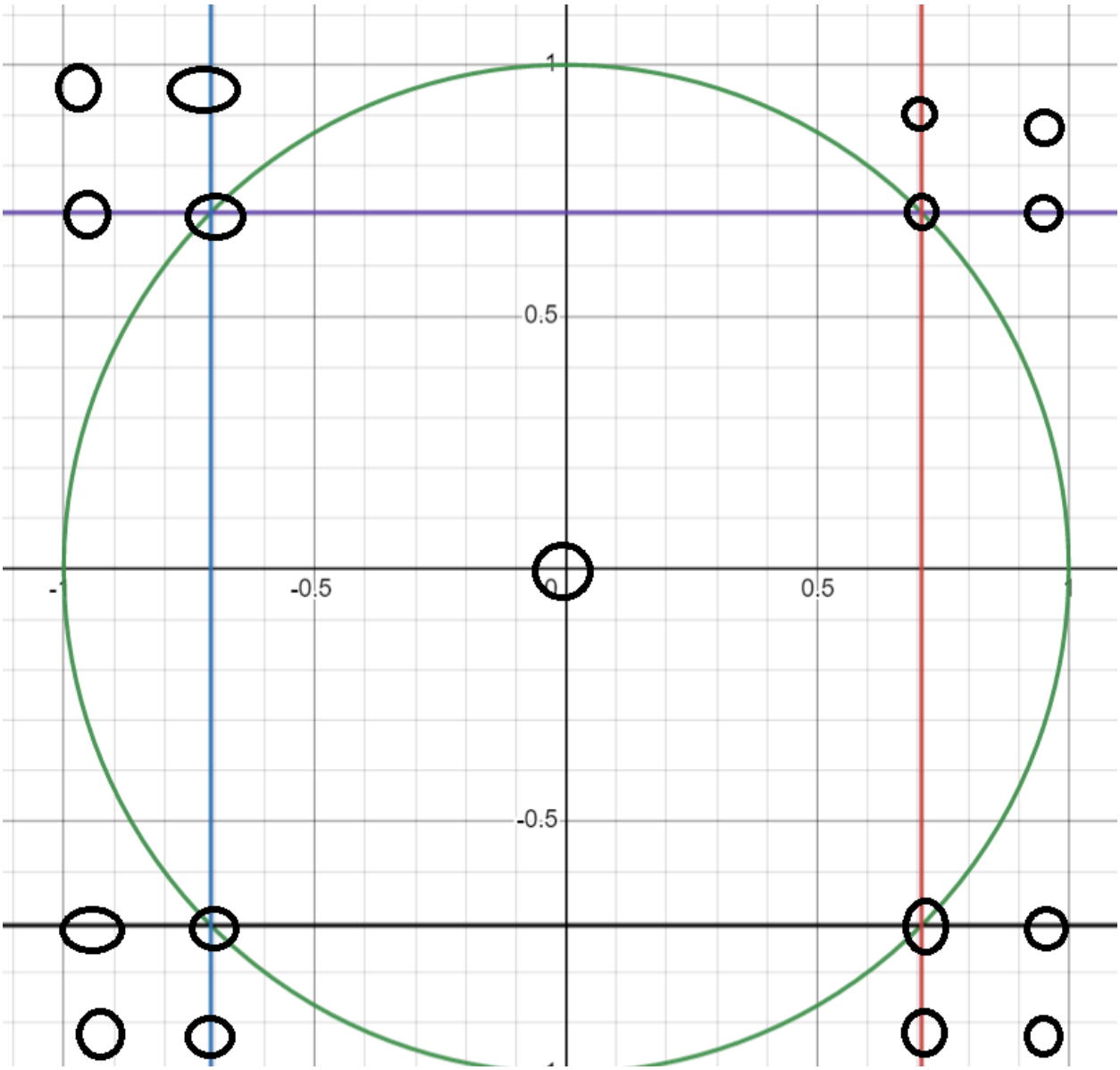
**Q4**: Identification of error(s) in the program. Must be identified and justified

According to the Weak nx1 testing method there were four errors, however these are approximation errors. The test cases 3, 6, 9 and 12 that failed all had points that were outside the approximated linear boundaries but were still inside the actual subdomain boundary defined by a circle with a radius of 1, i.e. $x^2 + y^2 \leq 1$. While the EPC yielded no failed test cases the actual test cases themselves were problematic because due to the nature of EPC testing strategy and the "diamond" shape of the approximated subdomain, every test case fell outside both the approximated and the actual subdomain except for test case 17, meaning the results of the testing effort are almost useless.

**Q5**: Discuss the effectiveness of the testing methods used

The EPC strategy had very little effectiveness due to the shape of the approximated subdomain. The 17 test cases only essentially tested one interior point and one exterior point between them. The Weak nx1 strategy was more effective with the same approximated subdomain where the majority of the test cases provided some insight. I believe that with the approximated subdomain that the EPC strategy was not efficient while the Weak nx1 was somewhat efficient.

To increase the effectiveness of the EPC strategy I believe simply rotating the existing approximated subdomain so it looked like the diagram below would increase the effectiveness and make its efficiency on par with the Weak nx1 strategy using the original "diamond" subdomain:




As seen above with a new approximated subdomain using the EPC strategy, its effectiveness and efficiency both increase and match the Weak nx1 strategy with the original subdomain.

The original approximation or the rotated one seen above both use four linear segments to approximate. This produces an okay approximation, where approximately 63% of the subdomain is encapsulated by the approximation(based on the area of the square vs the area of the circle). Increasing the number of linear segments will increase the area encapsulated (i.e. the approximation), however this only increases the effectiveness when using the Weak nx1 strategy. When using the EPC strategy the most efficient and effective is the rotated square shown above, using any more linear segments reduces the effectiveness to the case with the original linear approximation where almost all the points fall outside the original and approximated subdomains providing almost useless test cases. However with Weak nx1

strategy the effectiveness will increase with diminishing returns as the linear segments increase. The efficiency will decrease however as more linear segments are added as due to the formula $b(n+1)+1$, where $b$ is the number of linear segments. The complexity of the testing effort will also increase greatly with added linear segments because an equation must be found for each linear segment. For the EPC strategy no matter the amount of linear segments the number of test cases will always remain the same because it is based of the dimensionality of the problem, however for the Weak nx1 strategy using 8 linear segments for example would increase the total test cases to 25, approximately doubling the number of test cases form when only 4 linear segments were used, the same applies if $m = 16$, where then the tests cases would double again to be ~50.

## Conclusion (3 marks)

The purpose of the lab was to experiment with different methods of Black Box testing on two programs to find errors. The use of EPC and Weak nx1 testing strategies resulted in a more formal testing approach where one error was found in Part One by both testing methods and the differences and limitations of the testing methods was explored in Part Two. The ideas explored were the effectiveness and efficiency of both testing methods, and how factors such as the shape of the subdomain affected that. Part Two involved using a linear approximation to complete the testing, where shortcomings of the EPC strategy was observed due to input sub domain shape, and shortcomings of Weak nx1 strategy were observed such as the reduction in efficiency as the linear approximation was made more accurate.