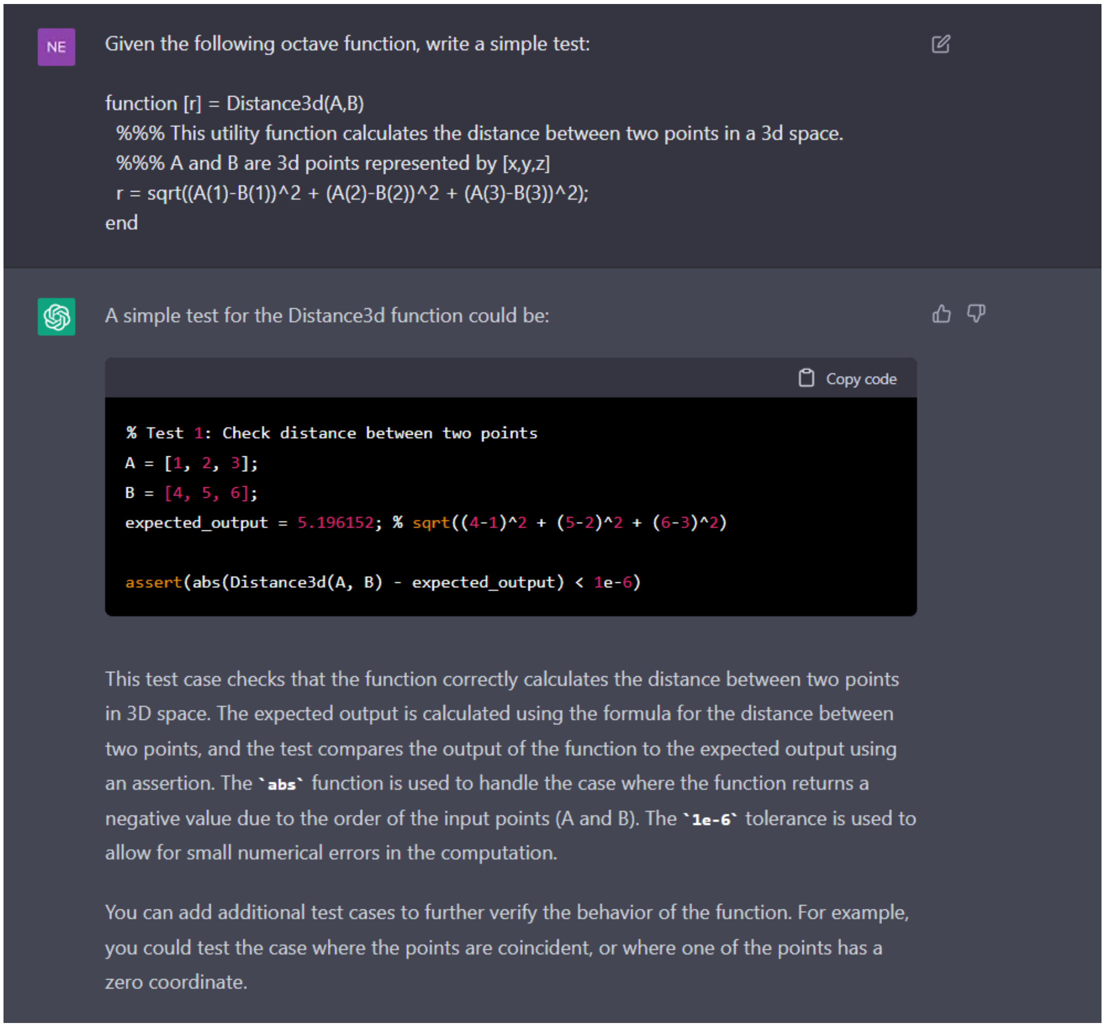


Given the following octave function, write a simple test:

```
function [r] = Distance3d(A,B)
  %%% This utility function calculates the distance between two points in a 3d space.
  %%% A and B are 3d points represented by [x,y,z]
  r = sqrt((A(1)-B(1))^2 + (A(2)-B(2))^2 + (A(3)-B(3))^2);
end
```

A simple test for the Distance3d function could be:

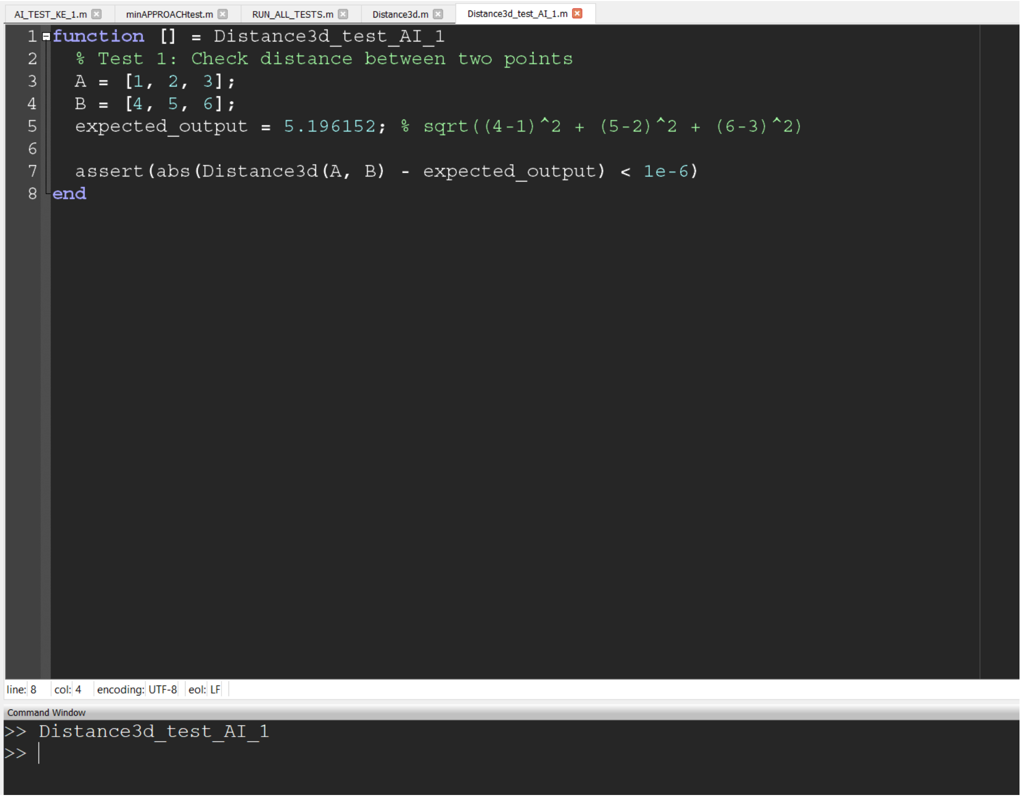
```
% Test 1: Check distance between two points
A = [1, 2, 3];
B = [4, 5, 6];
expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

assert(abs(Distance3d(A, B) - expected_output) < 1e-6)
```

This test case checks that the function correctly calculates the distance between two points in 3D space. The expected output is calculated using the formula for the distance between two points, and the test compares the output of the function to the expected output using an assertion. The `abs` function is used to handle the case where the function returns a negative value due to the order of the input points (A and B). The `1e-6` tolerance is used to allow for small numerical errors in the computation.

You can add additional test cases to further verify the behavior of the function. For example, you could test the case where the points are coincident, or where one of the points has a zero coordinate.

The test passes right off the bat:



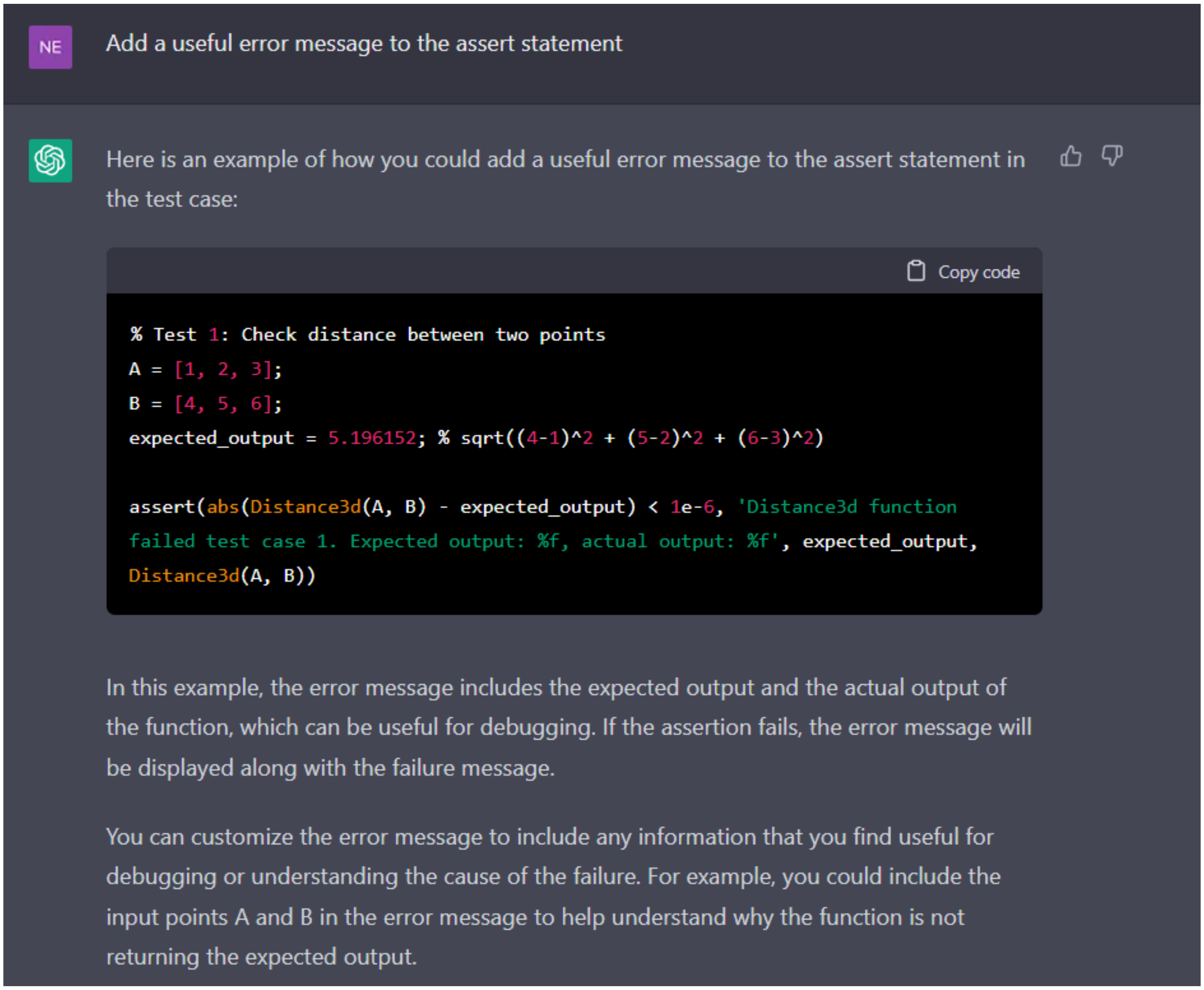
```
function [] = Distance3d_test_AI_1
  % Test 1: Check distance between two points
  A = [1, 2, 3];
  B = [4, 5, 6];
  expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

  assert(abs(Distance3d(A, B) - expected_output) < 1e-6)
end
```

line: 8 col: 4 encoding: UTF-8 eol: LF

```
>> Distance3d_test_AI_1
>> |
```

Trying to add an error message:



Add a useful error message to the assert statement

Here is an example of how you could add a useful error message to the assert statement in the test case:

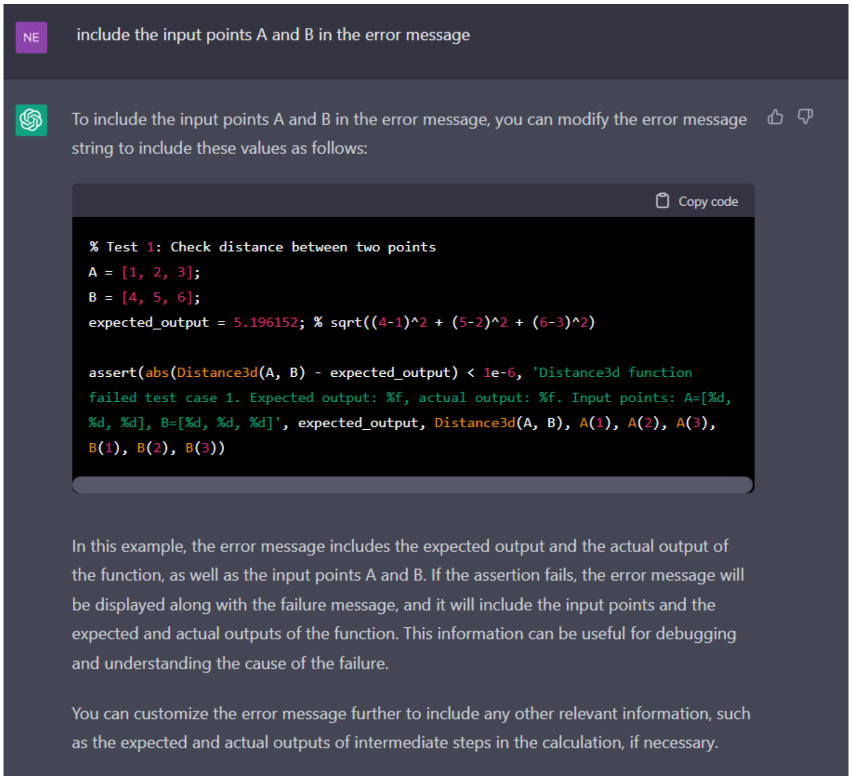
```
% Test 1: Check distance between two points
A = [1, 2, 3];
B = [4, 5, 6];
expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

assert(abs(Distance3d(A, B) - expected_output) < 1e-6, 'Distance3d function
failed test case 1. Expected output: %f, actual output: %f', expected_output,
Distance3d(A, B))
```

In this example, the error message includes the expected output and the actual output of the function, which can be useful for debugging. If the assertion fails, the error message will be displayed along with the failure message.

You can customize the error message to include any information that you find useful for debugging or understanding the cause of the failure. For example, you could include the input points A and B in the error message to help understand why the function is not returning the expected output.

Test passes again, so I ask it to add the points A and B to the error message as it suggested.



include the input points A and B in the error message

To include the input points A and B in the error message, you can modify the error message string to include these values as follows:

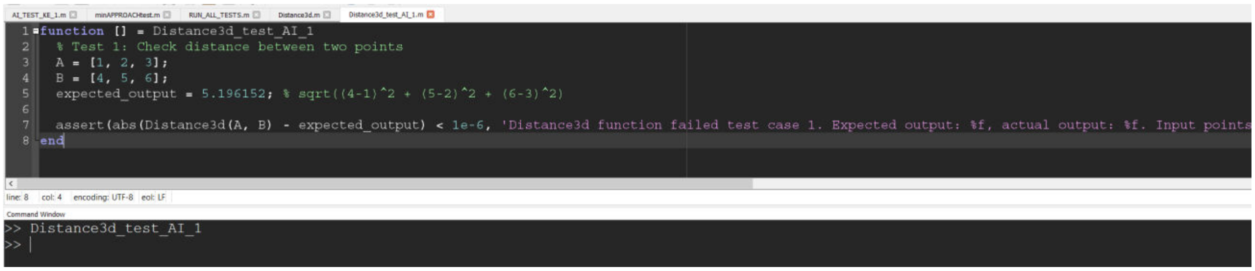
```
% Test 1: Check distance between two points
A = [1, 2, 3];
B = [4, 5, 6];
expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

assert(abs(Distance3d(A, B) - expected_output) < 1e-6, 'Distance3d function
failed test case 1. Expected output: %f, actual output: %f. Input points: A=[%d,
%d, %d], B=[%d, %d, %d]', expected_output, Distance3d(A, B), A(1), A(2), A(3),
B(1), B(2), B(3))
```

In this example, the error message includes the expected output and the actual output of the function, as well as the input points A and B. If the assertion fails, the error message will be displayed along with the failure message, and it will include the input points and the expected and actual outputs of the function. This information can be useful for debugging and understanding the cause of the failure.

You can customize the error message further to include any other relevant information, such as the expected and actual outputs of intermediate steps in the calculation, if necessary.

Test passes:

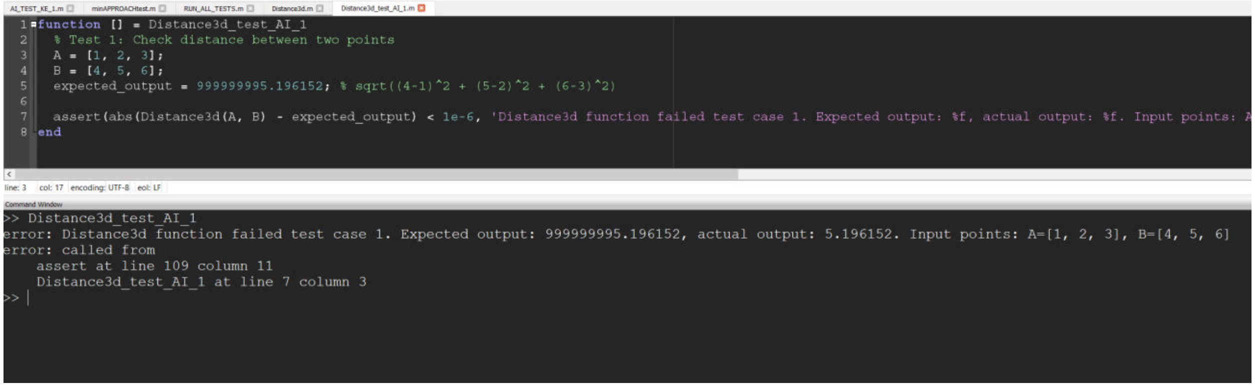


```
function [] = Distance3d_test_AI_1
  % Test 1: Check distance between two points
  A = [1, 2, 3];
  B = [4, 5, 6];
  expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

  assert(abs(Distance3d(A, B) - expected_output) < 1e-6, 'Distance3d function failed test case 1. Expected output: %f, actual output: %f. Input points
end
```

```
>> Distance3d_test_AI_1
>> |
```

Checking for error message if it fails by manually changing the expected value to be incorrect:

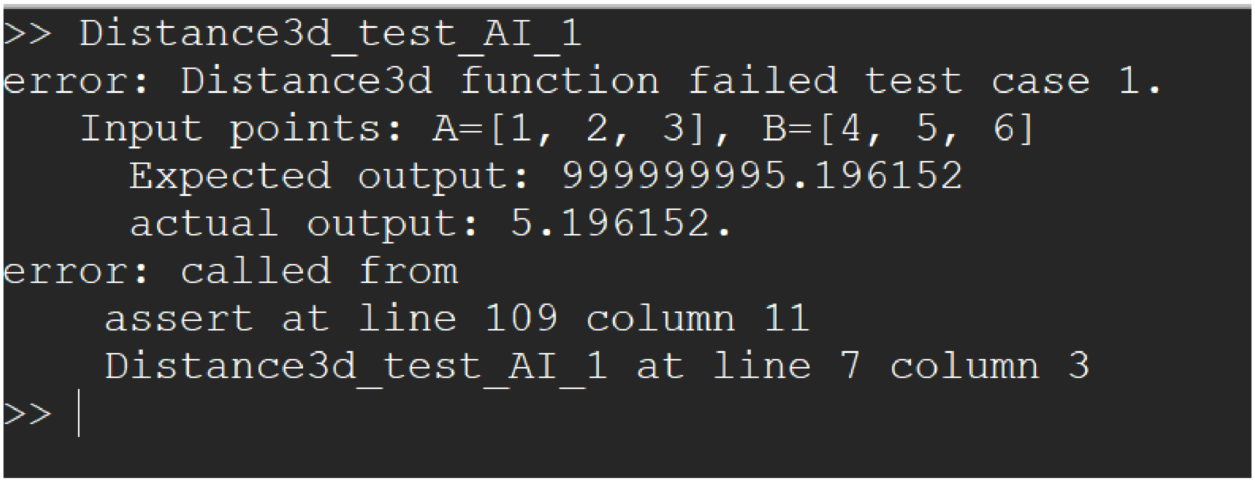


```
function [] = Distance3d_test_AI_1
  % Test 1: Check distance between two points
  A = [1, 2, 3];
  B = [4, 5, 6];
  expected_output = 999999995.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

  assert(abs(Distance3d(A, B) - expected_output) < 1e-6, 'Distance3d function failed test case 1. Expected output: %f, actual output: %f. Input points: A
end
```

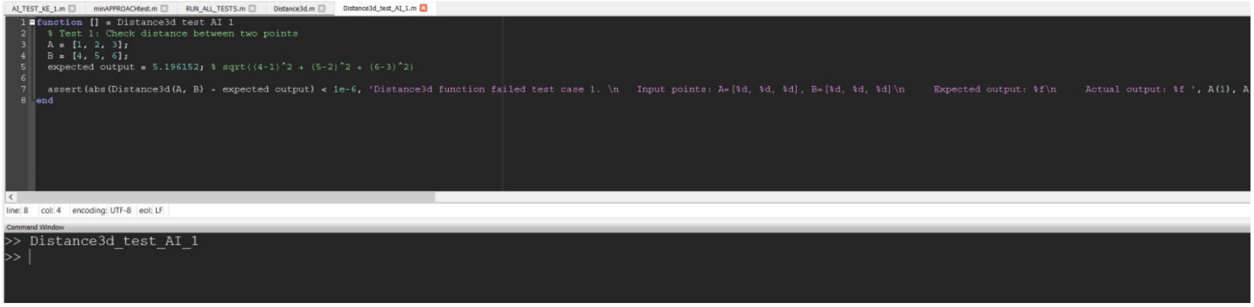
```
>> Distance3d_test_AI_1
error: Distance3d function failed test case 1. Expected output: 999999995.196152, actual output: 5.196152. Input points: A=[1, 2, 3], B=[4, 5, 6]
error: called from
    assert at line 109 column 11
    Distance3d_test_AI_1 at line 7 column 3
>> |
```

ChatGPT does a great job at understanding all the prompts here, but it doesn't do a few things that a human likely would. For example, a human would use \n in the error message to make it easier to read. A human would also likely provide the input first, followed by the expected and actual outputs. Here is an example error message.



```
>> Distance3d_test_AI_1
error: Distance3d function failed test case 1.
   Input points: A=[1, 2, 3], B=[4, 5, 6]
     Expected output: 999999995.196152
     actual output: 5.196152.
error: called from
    assert at line 109 column 11
    Distance3d_test_AI_1 at line 7 column 3
>> |
```

Here is the final passing test, incorporating the manual changes to the error message:



```
function [] = Distance3d_test_AI_1
  % Test 1: Check distance between two points
  A = [1, 2, 3];
  B = [4, 5, 6];
  expected_output = 5.196152; % sqrt((4-1)^2 + (5-2)^2 + (6-3)^2)

  assert(abs(Distance3d(A, B) - expected_output) < 1e-6, 'Distance3d function failed test case 1. \n   Input points: A=[%d, %d, %d], B=[%d, %d, %d]\n     Expected output: %f\n     Actual output: %f ', A(1), A
end
```

```
>> Distance3d_test_AI_1
>> |
```