


In the previous document, I asked ChatGPT for some test ideas:



NE

Below is an octave function to determine the distance between two points in 3d space. What are some ideas for tests?

```
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
```



Here are a few ideas for tests for the `Distance3d` function:


 

1. Test the function with two points that are known to be a certain distance apart. For example, test the function with points `A = [0, 0, 0]` and `B = [3, 0, 0]`, which are known to be 3 units apart. You could then compare the output of the function to the known distance to ensure that the function is correctly calculating the distance.
2. Test the function with two points that are not aligned on any of the axes. This will help ensure that the function is correctly handling cases where the points are not perfectly aligned on the `x`, `y`, or `z` axes.
3. Test the function with points that have negative coordinates. This will help ensure that the function is correctly handling negative coordinates and correctly calculating the distance between the points.
4. Test the function with points that are very close together, such as `A = [0, 0, 0]` and `B = [0.001, 0, 0]`. This will help ensure that the function is able to handle small distances and is not affected by rounding errors.
5. Test the function with points that are very far apart, such as `A = [0, 0, 0]` and `B = [1e9, 0, 0]`. This will help ensure that the function can handle large distances and is not affected by rounding errors or other numerical issues.


I then asked it to write test #4

NE



create test number 4:




Test the function with points that are very close together, such as `A = [0, 0, 0]` and `B = [0.001, 0, 0]`. This will help ensure that the function is able to handle small distances and is not affected by rounding errors.



Here is an example of how you might implement test 4 for the `Distance3d` function:

 Copy code

```
% Test 4: Test the function with points that are very close together
A = [0, 0, 0];
B = [0.001, 0, 0];
expected_distance = 0.001; % calculated using the correct formula

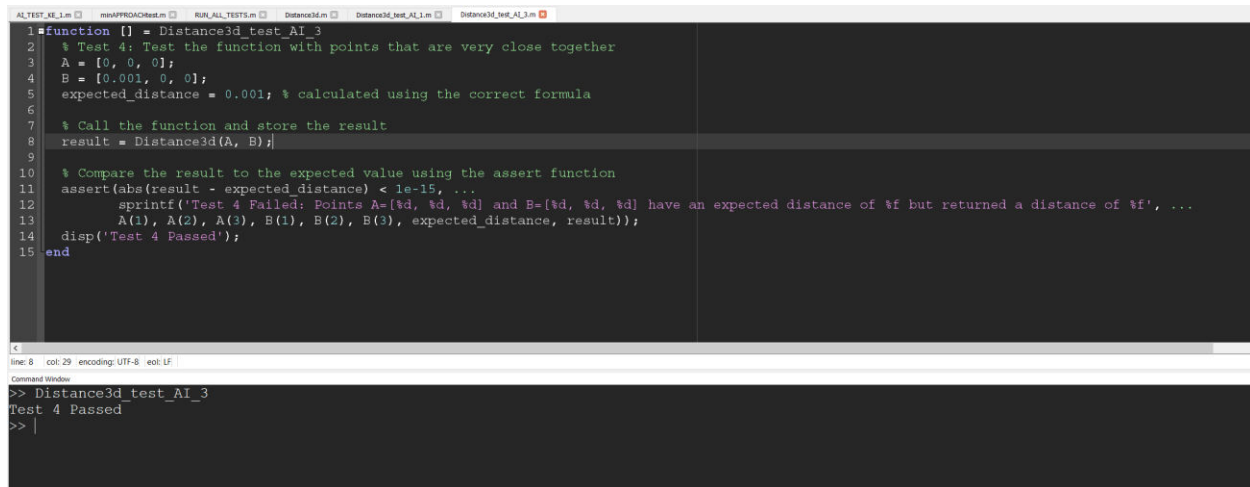
% Call the function and store the result
result = Distance3d(A, B);

% Compare the result to the expected value using the assert function
assert(abs(result - expected_distance) < 1e-15, ...
    sprintf('Test 4 Failed: Points A=[%d, %d, %d] and B=[%d, %d, %d] have an expected distance of %f but returned a distance of %f', ...
        A(1), A(2), A(3), B(1), B(2), B(3), expected_distance, result));
disp('Test 4 Passed');
```

In this test case, the points `A` and `B` are very close together, with a distance of 0.001 units. The test compares the result of the `Distance3d` function to the expected distance, and the test passes if the result is within $1e-15$ of the expected value. This tolerance is included to account for possible rounding errors.

This test case will help ensure that the function is able to handle small distances and is not affected by rounding errors. It will also help verify that the function is correctly handling the case where the points are not perfectly aligned on the `x`, `y`, or `z` axes.

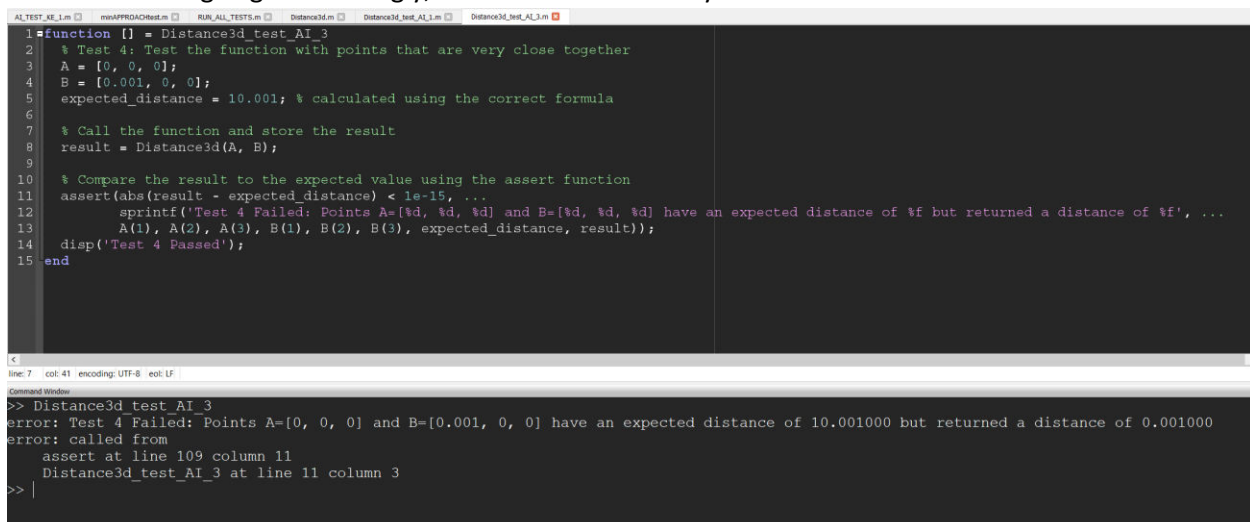
It spit out some junk about how this test will also “verify that the function is correctly handling the case where the points are not perfectly aligned on the x, y, or z axes.”, but whatever, the test passes.



```
1 function [] = Distance3d_test_AI_3
2 % Test 4: Test the function with points that are very close together
3 A = [0, 0, 0];
4 B = [0.001, 0, 0];
5 expected_distance = 0.001; % calculated using the correct formula
6
7 % Call the function and store the result
8 result = Distance3d(A, B);
9
10 % Compare the result to the expected value using the assert function
11 assert(abs(result - expected_distance) < 1e-15, ...
12     sprintf('Test 4 Failed: Points A=[%d, %d, %d] and B=[%d, %d, %d] have an expected distance of %f but returned a distance of %f', ...
13     A(1), A(2), A(3), B(1), B(2), B(3), expected_distance, result));
14 disp('Test 4 Passed');
15 end
```

```
>> Distance3d_test_AI_3
Test 4 Passed
>> |
```

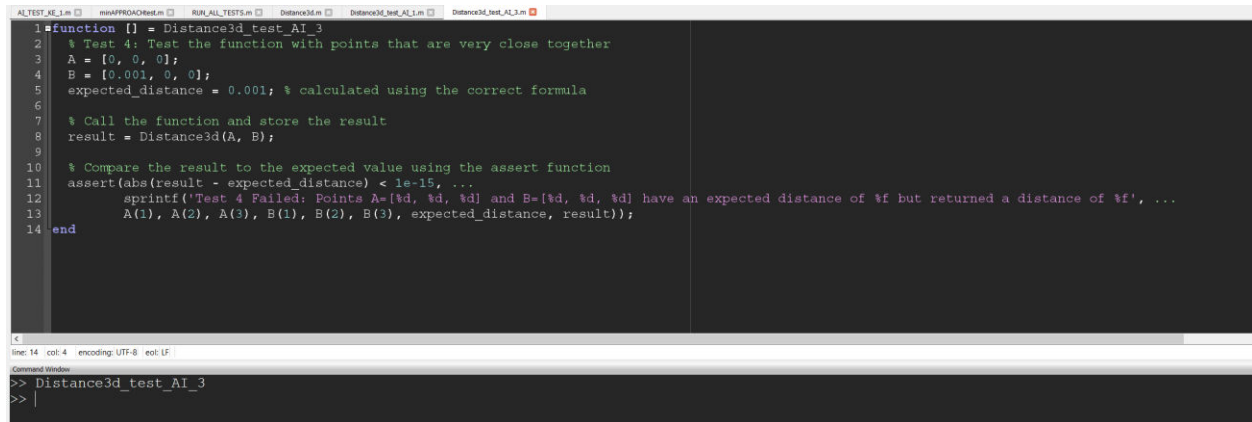
The error message again looks ugly, so I fixed that manually:



```
1 function [] = Distance3d_test_AI_3
2 % Test 4: Test the function with points that are very close together
3 A = [0, 0, 0];
4 B = [0.001, 0, 0];
5 expected_distance = 10.001; % calculated using the correct formula
6
7 % Call the function and store the result
8 result = Distance3d(A, B);
9
10 % Compare the result to the expected value using the assert function
11 assert(abs(result - expected_distance) < 1e-15, ...
12     sprintf('Test 4 Failed: Points A=[%d, %d, %d] and B=[%d, %d, %d] have an expected distance of %f but returned a distance of %f', ...
13     A(1), A(2), A(3), B(1), B(2), B(3), expected_distance, result));
14 disp('Test 4 Passed');
15 end
```

```
>> Distance3d_test_AI_3
error: Test 4 Failed: Points A=[0, 0, 0] and B=[0.001, 0, 0] have an expected distance of 10.001000 but returned a distance of 0.001000
error: called from
    assert at line 109 column 11
    Distance3d_test_AI_3 at line 11 column 3
>> |
```

I reverted the expected distance to the correct value, and removed the last line, as the test should pass silently. The test now passes:



The image shows a MATLAB editor window with several tabs. The active tab is 'Distance3d_test_AI_3.m'. The code in the editor is as follows:

```
1 function [] = Distance3d_test_AI_3
2 % Test 4: Test the function with points that are very close together
3 A = [0, 0, 0];
4 B = [0.001, 0, 0];
5 expected_distance = 0.001; % calculated using the correct formula
6
7 % Call the function and store the result
8 result = Distance3d(A, B);
9
10 % Compare the result to the expected value using the assert function
11 assert(abs(result - expected_distance) < 1e-15, ...
12        sprintf('Test 4 Failed: Points A=[%d, %d, %d] and B=[%d, %d, %d] have an expected distance of %f but returned a distance of %f', ...
13               A(1), A(2), A(3), B(1), B(2), B(3), expected_distance, result));
14 end
```

Below the editor is a Command Window. It shows the command 'Distance3d_test_AI_3' being executed, and the prompt '>>' is visible, indicating the command was successful.