Examples for using the TEST_EQUAL and TEST_UNEQUAL functions.

Table of Contents

Note: In these examples, we only deal with 2D arrays. However, the function can handle higher dimensional arrays as well.

Example #1: Two identical arrays.

Let's say a function produces a result

$$\mathbf{A}_{\text{actual}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

The true (i.e. expected) result is also

$$\mathbf{A}_{\text{expected}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Thus, $\mathbf{A}_{\text{actual}} = \mathbf{A}_{\text{expected}}$. Defining these arrays in MATLAB,

```
actual = [1  1;
          1  1];
expected = [1  1;
            1  1];
```

Since the actual and expected results are exactly the same, TEST_UNEQUAL *should* produce an error while TEST_EQUAL should *not*.

```
% does not produce error
TEST_EQUAL(actual,expected);

% produces error
TEST_UNEQUAL(actual,expected);
```

```
Error using assert
Assertion failed.

Error in TEST_UNEQUAL (line 68)
        assert(err(i) > min_err);
```

Example #2: Two slightly different arrays.

Let's say a function produces a result

$$\mathbf{A}_{\text{actual}} = \begin{bmatrix} 1.00000001 & 1.00000001 \\ 1.00000001 & 1.00000001 \end{bmatrix}$$

The true (i.e. expected) result should be

$$\mathbf{A}_{\text{expected}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Defining these arrays in MATLAB,

```
expected = [1  1;
            1  1];
actual = expected+0.00000001;
```

Since the actual and expected results differ, `TEST_EQUAL` *should* produce an error while `TEST_UNEQUAL` should *not*.

```
% does not produce error
TEST_UNEQUAL(actual,expected);

% produces error
TEST_EQUAL(actual,expected);
```

```
Error using assert
Assertion failed.

Error in TEST_EQUAL (line 67)
       assert(err(i) <= max_err);
```

Example #3: Equality to within some specified precision.

Let's consider the same two arrays from Example #2:

$$\mathbf{A}_{\text{actual}} = \begin{bmatrix} 1.00000001 & 1.00000001 \\ 1.00000001 & 1.00000001 \end{bmatrix}$$

$$\mathbf{A}_{\text{expected}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Defining them in MATLAB,

```
expected = [1  1;
            1  1];
actual = expected+0.00000001;
```

While these arrays are not *exactly* equal, in many cases, we can consider them to be effectively equal. Specifically, consider the case where we say two arrays are equal if their elements are equal to within 10^{-6}. Under this criteria, we'd have that $\mathbf{A}_{\text{actual}}$ is equal to $\mathbf{A}_{\text{expected}}$. First, let's set the error criteria to 10^{-6}.

```
err = 1e-6;
```

Under this error criteria, `TEST_UNEQUAL` *should* produce an error while `TEST_EQUAL` should *not*.

```
% does not produce error
TEST_EQUAL(actual,expected,err);

% produces error
TEST_UNEQUAL(actual,expected,err);
```

```
Error using assert
Assertion failed.

Error in TEST_UNEQUAL (line 68)
       assert(err(i) > min_err);
```