Examples for using the TEST_EQUAL and TEST_NOT_EQUAL 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.

Consider two arrays, $\mathbf{X}_1$ and $\mathbf{X}_2$.

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{X}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Thus, $\mathbf{X}_1 = \mathbf{X}_2$. Defining these arrays in MATLAB,

```
X1 = [1  1;
      1  1];
X2 = [1  1;
      1  1];
```

Since the two arrays are exactly the same, TEST_EQUAL should *not* produce an error while TEST_NOT_EQUAL *should*.

```
% does not produce error
TEST_EQUAL(X1,X2);

% produces error
TEST_NOT_EQUAL(X1,X2);
```

```
Error using TEST_NOT_EQUAL
Arrays are equal.
```

Example #2: Two slightly different arrays.

Consider two arrays, $\mathbf{X}_1$ and $\mathbf{X}_2$.

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{X}_2 = \begin{bmatrix} 1.00000001 & 1.00000001 \\ 1.00000001 & 1.00000001 \end{bmatrix}$$

Defining these arrays in MATLAB,

```
X1 = [1  1;
      1  1];
X2 = X1+0.00000001;
```

Since the two arrays differ, TEST_NOT_EQUAL should *not* produce an error while TEST_EQUAL *should*.

```
% does not produce error
TEST_NOT_EQUAL(X1,X2);

% produces error
TEST_EQUAL(X1,X2);
```

```
Error using TEST_EQUAL
Arrays are not equal.
```

Example #3: Equality to a certain number of decimal places.

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

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{X}_2 = \begin{bmatrix} 1.00000001 & 1.00000001 \\ 1.00000001 & 1.00000001 \end{bmatrix}$$

Defining them in MATLAB,

```
X1 = [1  1;
      1  1];
X2 = X1+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 7 decimal places. Under this criteria, we'd have that $\mathbf{X}_1$ is equal to $\mathbf{X}_2$. Therefore, let's set the number of decimal places to 7 for testing this equality.

```
n = 7;
```

Under this precision, TEST_EQUAL should *not* produce an error while TEST_NOT_EQUAL *should*.

```
% does not produce error
TEST_EQUAL(X1,X2,n);

% produces error
TEST_NOT_EQUAL(X1,X2,n);
```

```
Error using TEST_NOT_EQUAL
Arrays are equal.
```

Note that if we set n = 8, then TEST_EQUAL would have thrown an error while TEST_NOT_EQUAL wouldn't have.