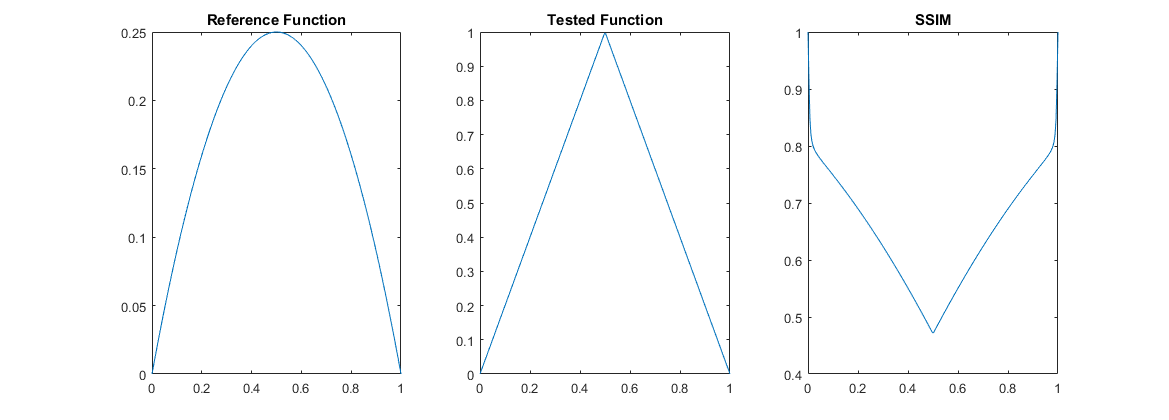
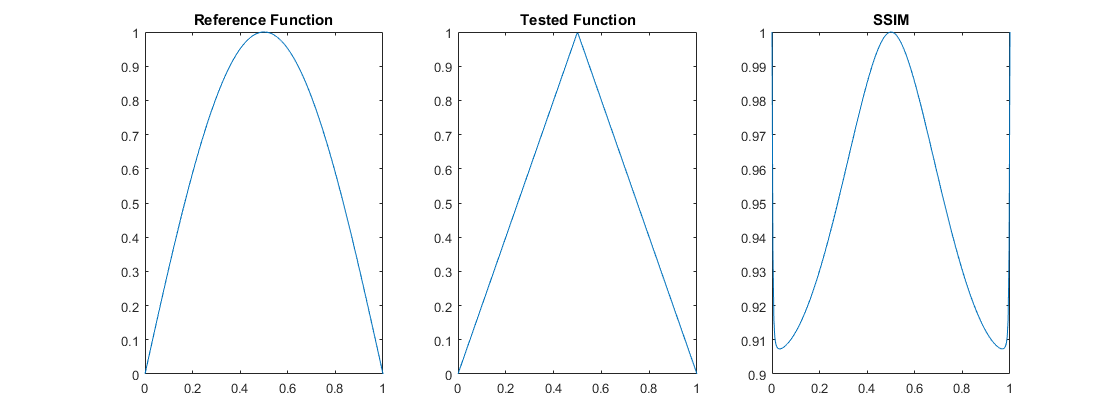
I’ll admit, I consider myself a pretty good programmer and I really like the material for this course, so it has been a little depressing to see just how little effort I’ve put into the previous assignments. I figured I’d dump some time into this one both for the sake of my ego and to make poor Ge happy! I’m going to do most of this in Matlab rather than Python for two main reasons. First, I’m more comfortable doing lots of math in Matlab than I am in python, and second, I need a brief Matlab demo to show off to some other undergrads and this might as well be it.

First, I wanted to use Matlab to just implement the formula as written in the paper with the functions as given. This, admittedly, presents a problem on the interval given ([0,1]). The dynamic range of sin(pi\*x) is 0-1, for x(1-x) it is 0-0.25, for 1-2|x-0.5| it is 0-1. So two of the functions given have the same dynamic range, while the third doesn’t. I decided to just run it as given at first, but to scale these to have the same dynamic range, I could just use 4x(1-x). First, we’ll do this numerically. I ran each of the functions from 0-1 using 256 values in-between.

The paper uses an 11x11 Gaussian window, but with the way they define their σxy term (as an inner product) means that the sampling window needs to be a square matrix so that internal dimensions match. For these functions, in 1D, I need to use a window of 1. You can see the attached Matlab code for the instrumentation, but it returns an MSSIM of 1 when it is provided 2 identical functions. When it is provided two different ones however, it computes a nice SSIM function!

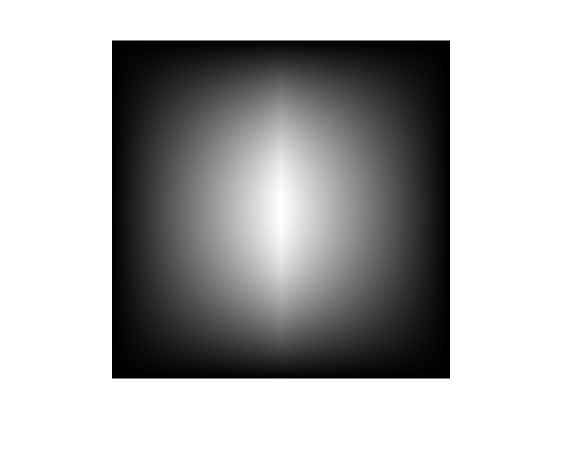
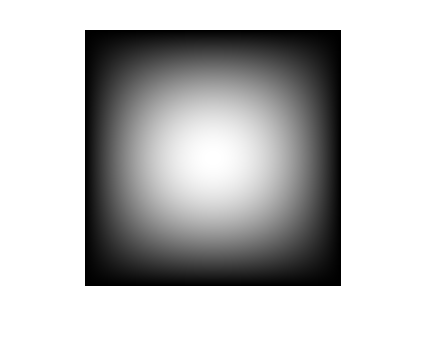


As you can see, the SSIM is only near 1 (minimum error) at the boundaries. Because the dynamic range of these two functions is wildly different, the SSIM never comes back to 1 in the middle, which you can see it clearly doing when I use different functions.



I thought I might take a crack at getting the solution solved out analytically by hand, but decided after looking at all those integrals that it was going to be a pain. Instead, I thought I’d have some fun and make SSIM work in 2D as originally intended. Only… Matlab has beaten me to the punch and has a SSIM function already! So, I thought, let’s just play around with the three functions we have, and make some cool pictures. I decided to combine the functions to make 2D images, and then run those through Matlab’s SSIM code.

To accomplish this, I represented the functions as a horizontal and a vertical array, and then through dot product, created one large 256x256 image. I also normalized all functions to have a dynamic range of 1 for the sake of my sanity. I used mat2gray to make the matrices into images. The images created from this are…. not that entertaining if I’m being honest with you. See below. From left to right, this is 1-2|x-.5| merged with sin(pi\*x), then x(1-x) merged with 1-2|x-.5|, and then x(1-x) merged with sin(pi\*x).



Let’s call the three above images A, B and C for our sake. There are, of course, many more combinations, but for the sake of time I’m only rolling out a few. As you can see these are riveting images! But they ARE in grayscale, so that’s fantastic! I was having some trouble implementing the ssim function matlab has with these images, so instead I am using a function called ssim\_index (also attached, credit is included in the function). The SSIM map for B/C and for A/B (respectively) are as follows.



As you can see, the patterns we start seeing in 2D get WAY cooler. If you stare at them long enough I swear they start to move. They also contain information about the functions they came from, which is hardbaked into that image! Pretty cool, right!

Anyway, that’s enough from me for now, I hope you all have fun grading!