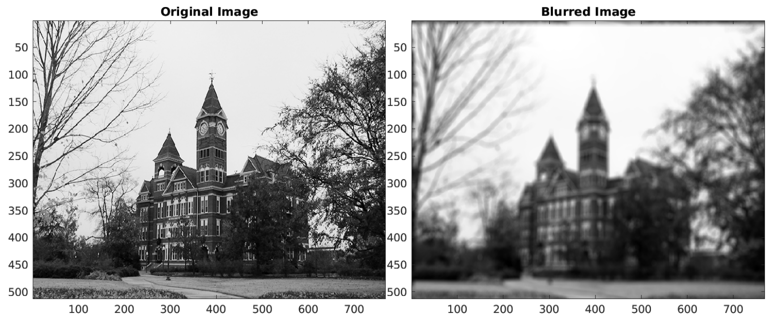
Matt Boler
ELEC7450 – HW6

Commonly, an image will need to be restored if it's corrupted by noise or out-of-focus blurring. We can approximate out-of-focus blurring by applying a disk point-spread-function, shown below. This method uniformly spreads every pixel in the input across a circular area in the output. In a real out-of-focus blur, the output would be blurred by different amounts depending on the distance of the scene and the blur itself would be more similar to a Gaussian blur than a uniform circular blur.



A reasonable first attempt to restore the image would be to mimic plant inversion from control theory and apply the inverse of the point-spread-function. This method, called inverse filtering, handles blur perfectly fine (as our blur is an LSI function, this is unsurprising) but catastrophically fails in the presence of additive noise, so badly in fact that the image is only barely visible with Gaussian noise with a sigma of 0.2 pixels. Our inverse filtering function and our results are shown below.

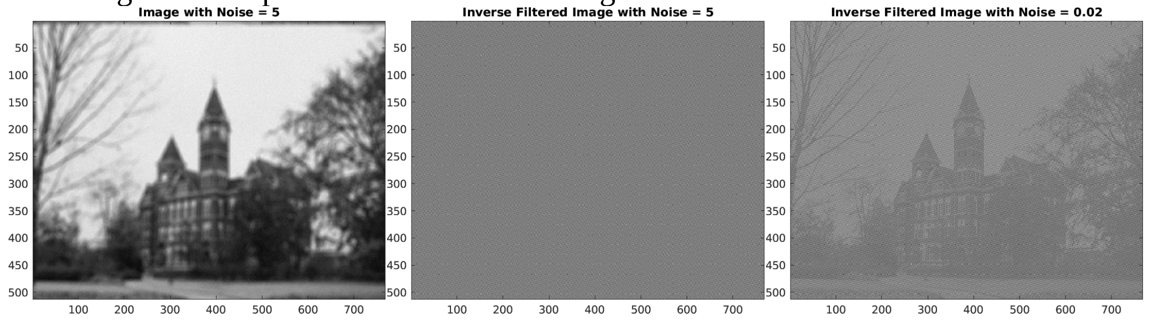
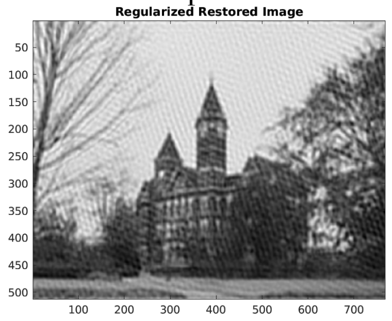


Image with Noise = 5
Inverse Filtered Image with Noise = 5
Inverse Filtered Image with Noise = 0.02

```
inverse_image_one_line = real(ifft2(...
        fft2(blurred_image) ./ fft2(ker, size(blurred_image, 1), size(blurred_image, 2))));
```

In an attempt to avoid the noise amplification this process results in, we can implement regularized inverse filtering. This method minimizes a cost function that includes the input/output similarity and the smoothness of the output image. The weighting factor of the smoothness term is a hyperparameter that must be adjusted to find the best output, which in this case was found at alpha = 0.2. This method handles noise significantly better than a standard inverse filter, but creates noise called 'ringing'. The commands to perform this filtering and the best image are shown below.



```
G = fft2(noise_image);
K_F = fft2(ker, size(image, 1), size(image, 2));
L_F = fft2(L, size(image, 1), size(image, 2));
F_hat = K_F ./ ( K_F.^2 + alpha * L_F.^2 ) .* G;
f_hat = real(ifft2(F_hat));
```