Question 1 Consider the heat equation, for $-1 \le x \le 1$, $0 \le t \le 1$,

$$\begin{cases} u_t = u_{xx}, \\ u(x,0) = 2\cosh(x), \\ u_x(-1,t) = 0, \\ u_x(1,t) = 0. \end{cases}$$

- Use uniform grid with $x_{1/2} = -1$ and $x_{n+1/2} = 1$. Find the DCT approximation of $u(x,0) = 2\cosh(x) \approx \sum_{k=0}^{n-1} y_k \cos(k\pi \frac{x+1}{2})$.
- The solution can be approximated as,

$$u(x,t) \approx \sum_{k=0}^{n-1} y_k(t) \cos(k\pi \frac{x+1}{2}).$$

Substitute the approximate solution into the heat equation, we should see,

$$\frac{d}{dt} y_k(t) = -y_k(t) \frac{k^2\pi^2}{4}.$$

So,

$$u(x,t) \approx \sum_{k=0}^{n-1} y_k(0) e^{-k^2\pi^2/4} \cos(k\pi \frac{x+1}{2}).$$

Plot the approximate solution at $t = 1$ for $n = 64, 128, 256, 512, 1024$.

Solving this problem required using the DCT of the vector formed by:

$$\frac{1}{\sqrt{n}} \begin{bmatrix} u(x_0,0) \\ u(x_1,0) \\ \dots \\ u(x_{n-1},0) \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} 2\cosh(x_0) \\ 2\cosh(x_1) \\ \dots \\ 2\cosh(x_{n-1}) \end{bmatrix}$$

The MATLAB DCT command does not divide by the square root of n term so this must be accounted for in the code.

The code to receive the approximate solution is on the next page.

```
% Coding 4 heat equation
% Uses the approximations and substitutions provided.
function [us, xs] = coding_four_heat(n, xl, xr)
xs = linspace(xl, xr, n);
ys = dct(2*cosh(xs))/sqrt(n);
us = zeros(n, 1);

for i=1:n
      x = xs(i);
      u = 0;
      for k=0:n-1
      term = exp(-k^2 * pi^2/4) * cos(k*pi*(x+1)/2);
      u = u + ys(k+1) * term;
      end
      us(i) = u;
end
```

The solutions, stored in the variable us, can be plotted against the xs to yield the following graph for various values of n.

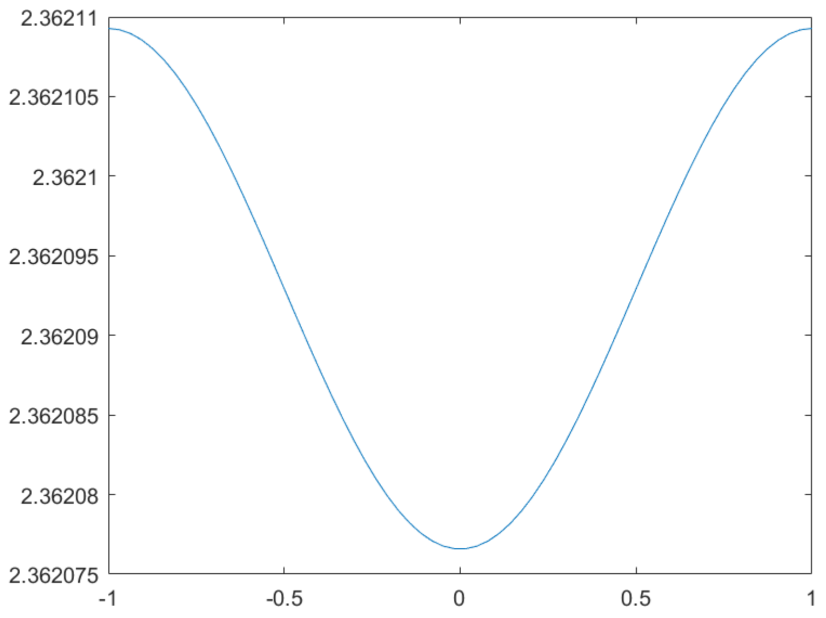


Fig 1. Approximate solution with n = 64

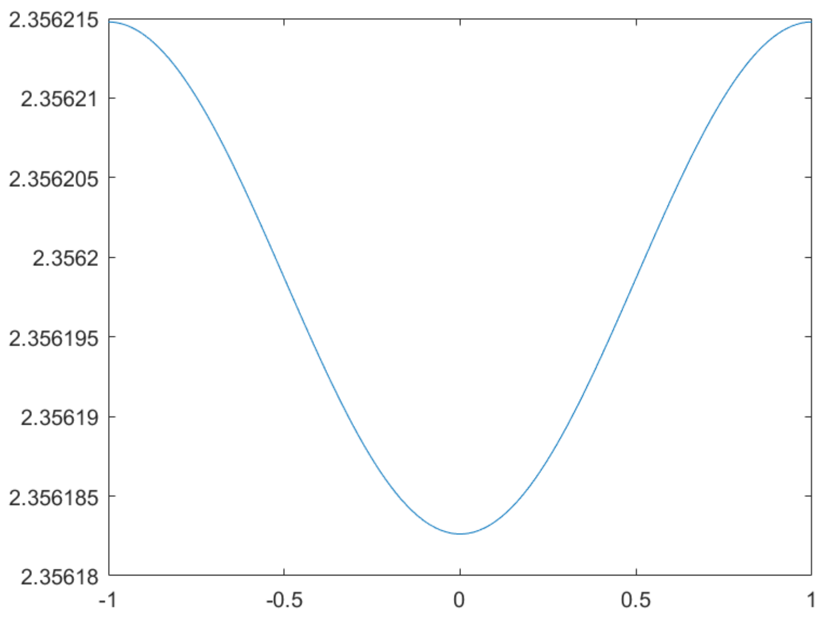


Fig 2. Approximate solution with n = 128

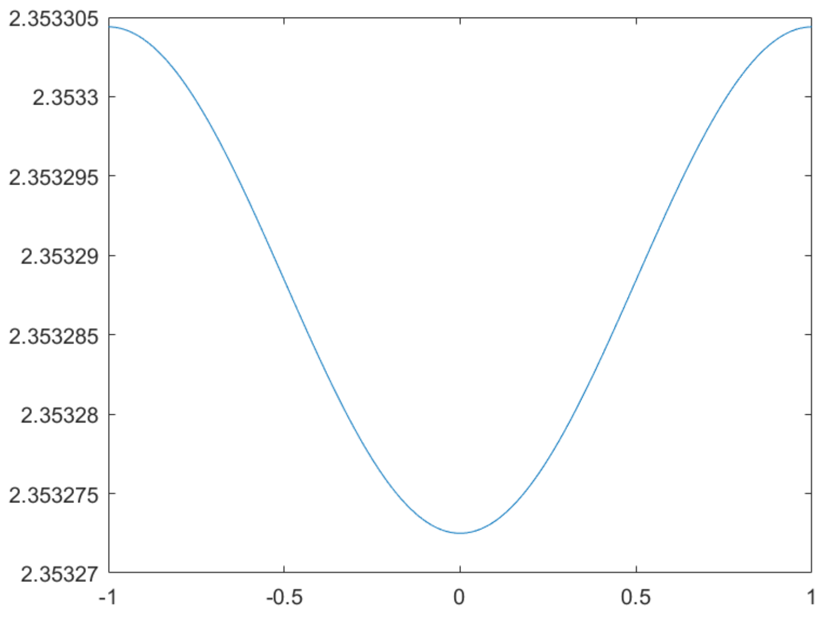


Fig 3. Approximate solution with n = 256

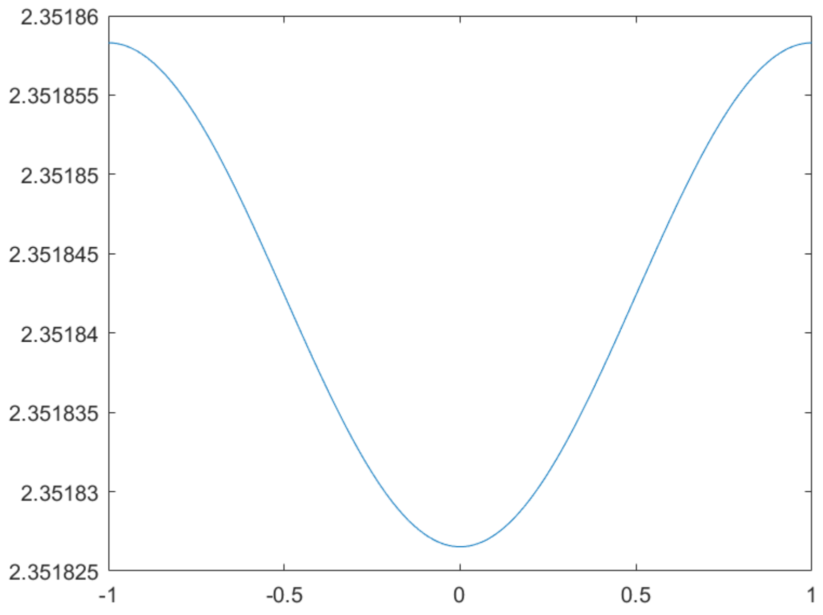


Figure 4. Approximate solution with n = 512

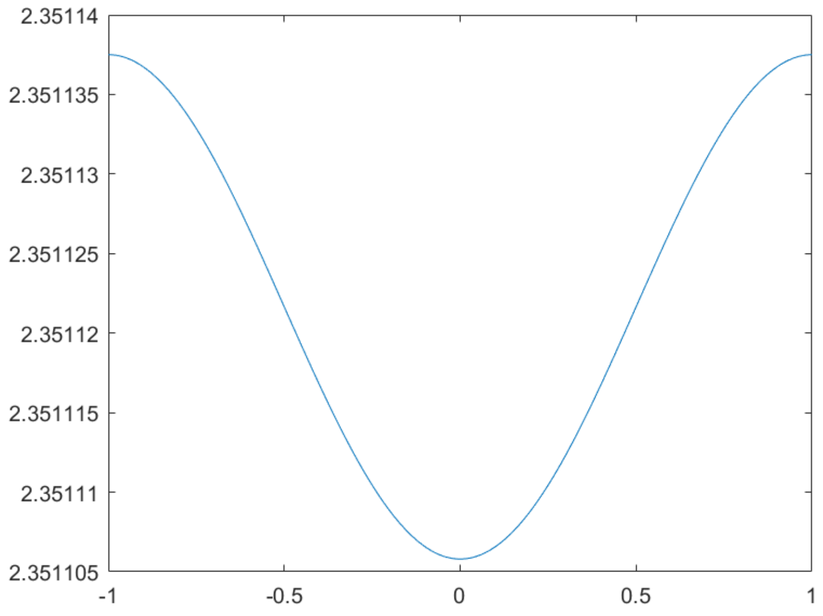
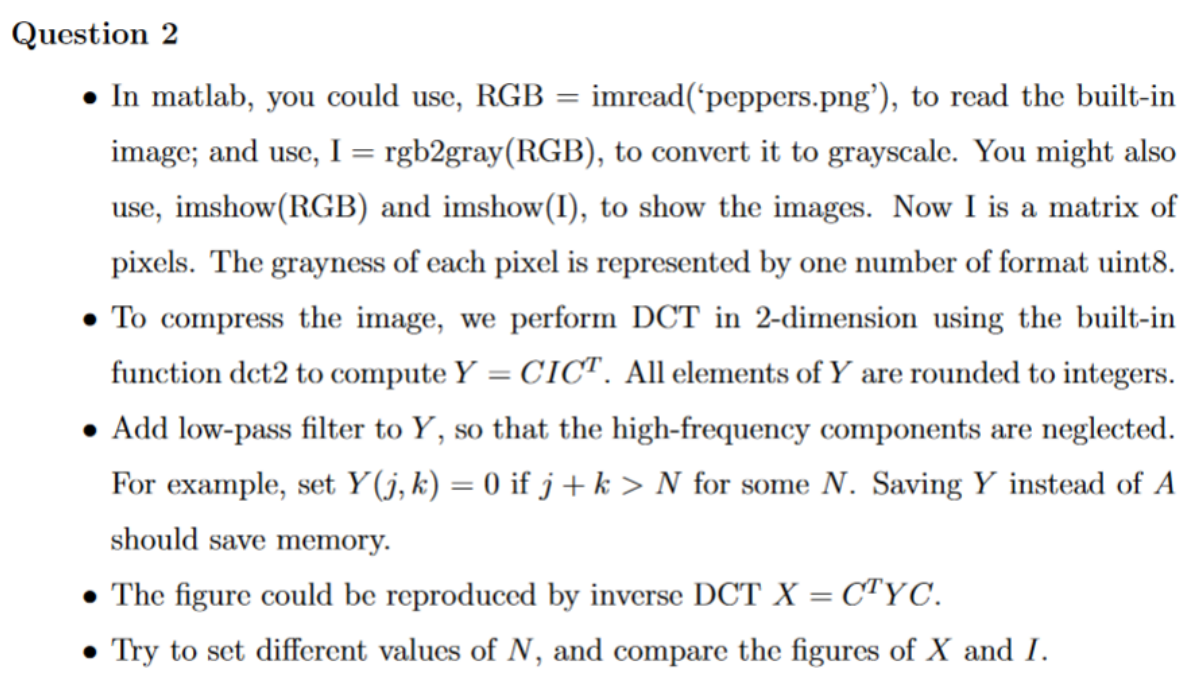


Figure 5. Approximate solution with n = 1024

As the value of n increases, there are a couple noticeable effects. The values of the peaks slowly converge to about 2.35511375, while the value of the trough slowly converges to about 2.351105, both decreasing towards that final value. Additionally, the range between the peak and the trough decreases. This, along with the very overall very small amplitude of the observed wave, suggests that the value at this time step is approximately constant.



Question 2

- In matlab, you could use, RGB = imread(‘peppers.png’), to read the built-in image; and use, I = rgb2gray(RGB), to convert it to grayscale. You might also use, imshow(RGB) and imshow(I), to show the images. Now I is a matrix of pixels. The grayness of each pixel is represented by one number of format uint8.
- To compress the image, we perform DCT in 2-dimension using the built-in function dct2 to compute $Y = CIC^T$. All elements of Y are rounded to integers.
- Add low-pass filter to Y, so that the high-frequency components are neglected. For example, set $Y(j,k) = 0$ if $j + k > N$ for some N. Saving Y instead of A should save memory.
- The figure could be reproduced by inverse DCT $X = C^T YC$.
- Try to set different values of N, and compare the figures of X and I.

The code implemented to perform the 2D DCT, low-pass filter, and then the 2D IDCT is as follows:

```matlab
% Coding 4 Image
function X = coding_four_image(I, N)
Y = round(dct2(I));
[m, n] = size(Y);
for i=1:m
   for j=1:n
       if i+j > N
           Y(i, j) = 0;
       end
   end
end
X = uint8(idct2(Y));
```

All the images passed through the argument I are already set to gray-scale. I executed this code for N = 2, 4, 8, … up until N was greater than both dimensions of the image, for two images found by default in MATLAB, “peppers.png” and “autumn.tif”. Additionally, I created new images to visually show the differences between the two images by taking the absolute value of the difference between the original grayscale image and the compressed version. The results start on the next page.

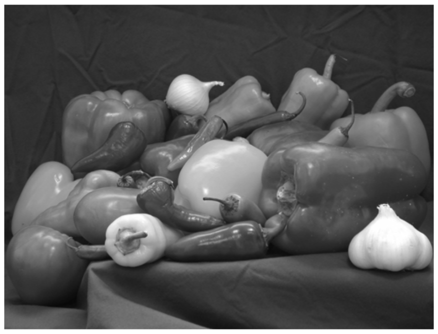


Figure 6. Greyscale peppers.png

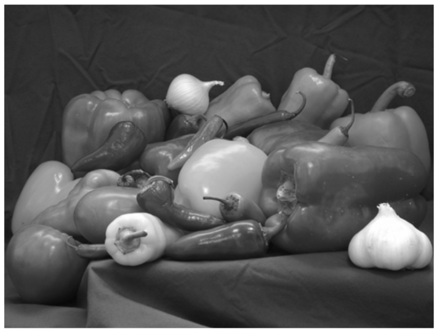


Figure 7. Peppers with N = 512

Figure 8. Difference between original and compressed with N = 512

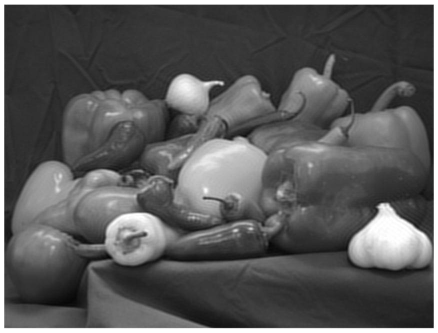


Figure 9. Peppers with N = 256

Figure 10. Difference between original and compressed with N = 256

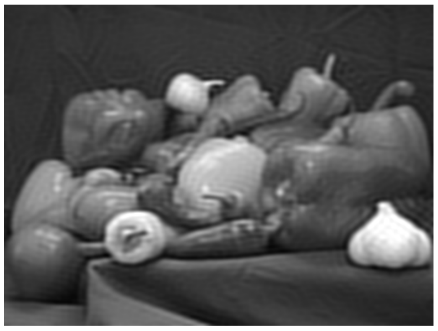


Figure 11. Peppers with N = 128

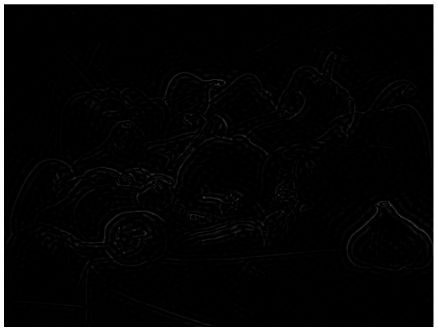


Figure 12. Difference between original and compressed with N = 128

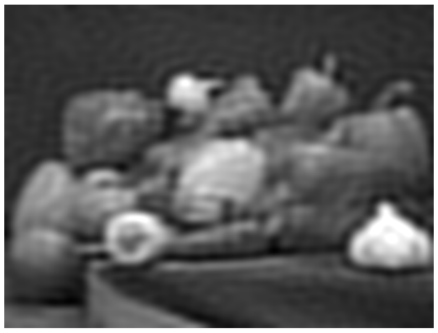


Figure 13. Peppers with N = 64

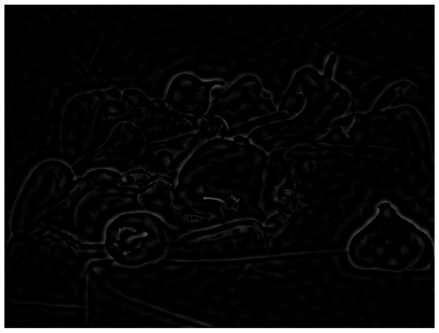


Figure 14. Difference between original and compressed with N = 64

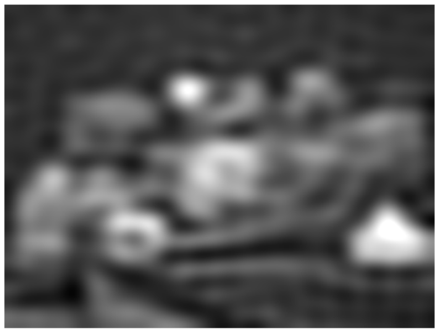


Figure 15. Peppers with N = 32

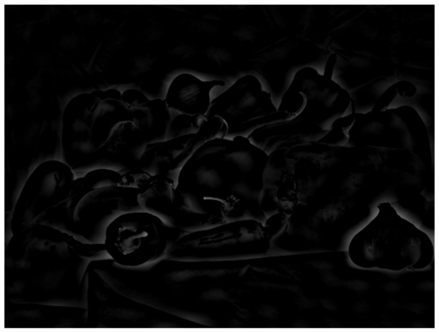


Figure 16. Difference between original and compressed with N = 32

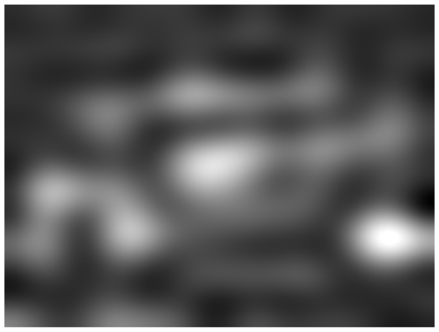


Figure 17. Peppers with N = 16

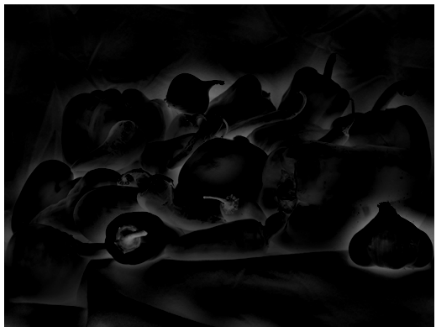


Figure 18. Difference between original and compressed with N = 16

Figure 19. Peppers with N = 8

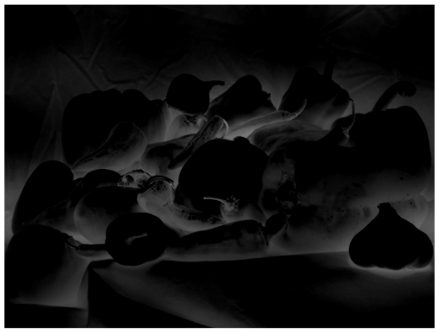


Figure 20. Difference between original and compressed with N = 8

Figure 21. Peppers with N = 4

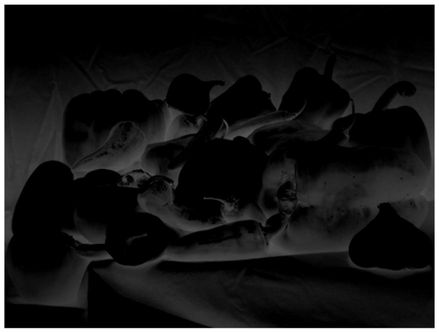


Figure 22. Difference between original and compressed with N = 4

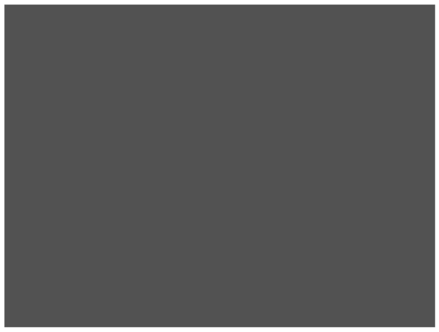


Figure 23. Peppers with N = 2

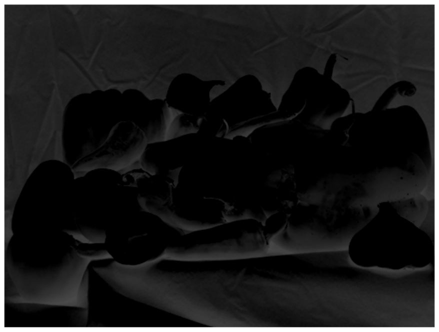


Figure 24. Difference between original and compressed with N = 2

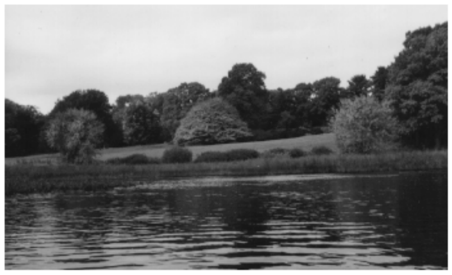


Figure 25. Greyscale autumn.tif

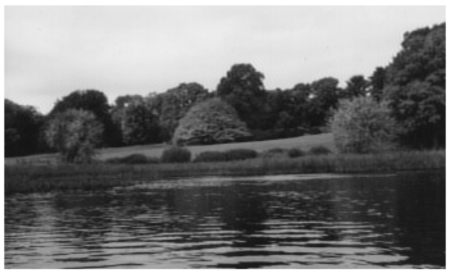


Figure 26. Autumn with N = 256

Figure 27. Difference between original and compressed with N = 256

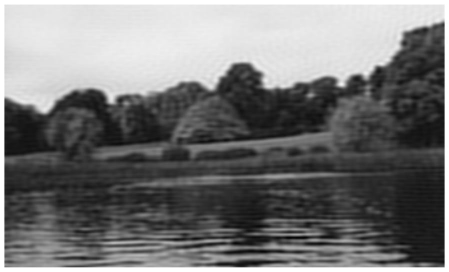


Figure 28. Autumn with N = 128

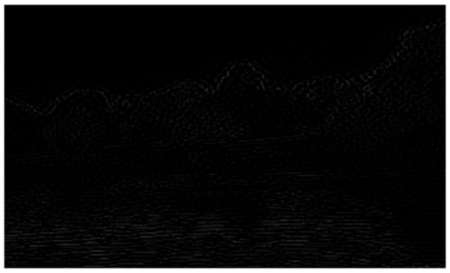


Figure 29. Difference between original and compressed with N = 128

Figure 30. Autumn with N = 64

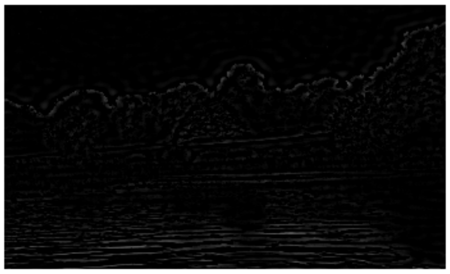


Figure 31. Difference between original and compressed with N = 64

Figure 32. Autumn with N = 32

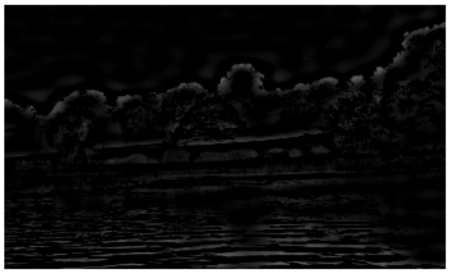


Figure 33. Difference between original and compressed with N = 32

Figure 34. Autumn with N = 16

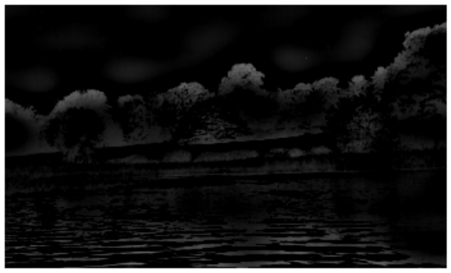


Figure 35. Difference between original and compressed with N = 16

Figure 36. Autumn with N = 8

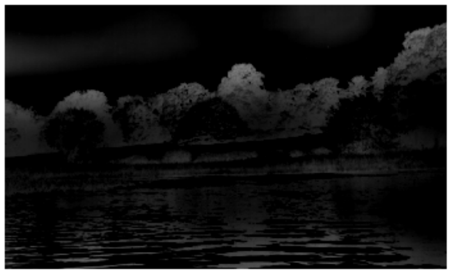


Figure 37. Difference between original and compressed with N = 8

Figure 38. Autumn with N = 4

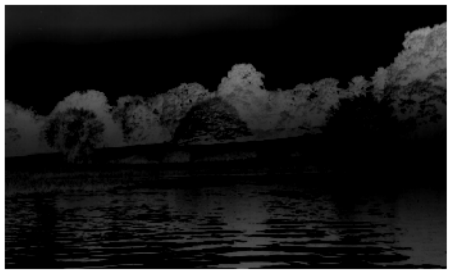


Figure 39. Difference between original and compressed with N = 4

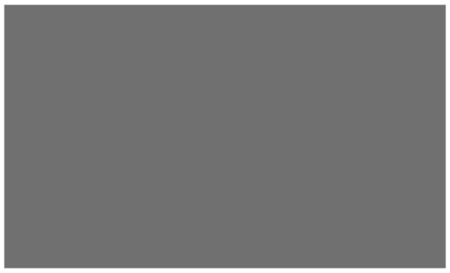


Figure 40. Autumn with N = 2

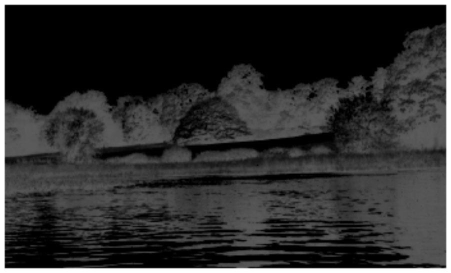


Figure 41. Difference between original and compressed with N = 2

As N falls further and further away from the dimensions of the image, the interpolating formula is more and more hard pressed to deal with sudden changes in value, resulting in a lot of noticeable compression where the value of the pixel is very different from some of the pixels around it. For the peppers, this results in the edges of the peppers being far less accurate and thus highlighted in the image that shows the differences. Meanwhile, for the autumn image, the ripples on the water and the area around the trees tend to fall victim more frequently.

This pattern suggests a possible application of the lossy DCT in edge detection, but that may be something far more better suited for convolutions.