Homework 5 - MSSC 6030: Spring 2020

Directions. All work is to be done in *complete sentences*. Assignments must be stapled with a printout of the assignment serving as the first page. Each problem must be on a *separate* sheet of paper. You are welcome to recycle paper, where one side is crossed out to avoid wasting paper, but your work MUST have **no more than one problem per page**. Each problem write-up must begin with the **full statement of the problem**. While you are encouraged to work through confusion with your classmates, your work must be written in your own words. **The assignment is due in dropbox on Wednesday, April 29, 2020 by 3:15pm.**

---

1. Using $N = 4$, write out the matrices for the DFT F_4 and the iDFT $\frac{1}{4}\overline{F_4}$ as described in class. Compute each entry of the matrix.

---

2. For your matrices above, compute each term to show that F_4 times $\frac{1}{4}\overline{F_4}$ gives you the identity matrix.

---

3. Use the data file posted on D2L entitled `DFT_example_data.mat` to determine the original signal from the given frequency data. You are given the sampling points x_j as well as the Discrete Fourier transform F of the data points f. You are tasked with identifying the main frequencies in the signal as well as recovering the signal at the data points i.e. recovering f_j's.

---

4. Write out the factorization of F_8 and multiply out all the matrices to get a hand on what is happening. (Note: Do not just use the matlab built in commands fft, actually go through the process. You can perform the matrix multiplication in matlab, but write out the matrices yourself.)

---

5. Load in a picture of your choice. Perform FFT filtering as described in the lecture to determine an acceptable % of the Fourier basis that can be zeroed out and still retain image quality. What are you using as a measure of image quality? Why?

---

6. Approximate functions using Haar wavelets. Approximate the function $f(t) = 4t^3 - 3t + sin(t)$, for t in $[0, 1]$ using the Haar wavelet basis. How many wavelets do you 'need' to capture the essential behavior? Try just using the first 8 we wrote down on the board together in class (orthonormal version or just amplitude 1 version, either is fine).

---

7. Run the Cascade algorithm (posted on D2L) for the following filters h.
   (a) Daubechies 4, i.e. $h = \left(1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3}\right)/8$
   (b) The Cubic B-spline: $h = (1, 4, 6, 4, 1)/16$
   (c) #12 of Section 4.7: $h = (-1, 2, 6, 2, -1)/8$
   (d) Can you find another one that works?

---

problem 1

N=4

$$\mathrm{DFT}: \widehat{f_k} = \sum_{j=0}^{N-1} f_j e^{-i\alpha}$$

$$\alpha = \frac{2\pi \mathrm{kj}}{N}$$

let $\omega_N = e^{-\frac{2\pi i}{N}}$, $\widehat{f_k} = \sum_{j=0}^{N-1} f_j \omega_N^{\mathrm{jk}}$

When k=1

$$\widehat{f_0} = \sum_{j=0}^{4-1} f_j \omega_4^{\mathrm{j}0} = [1 \quad 1 \quad 1 \quad 1]$$

When k=2

$$\widehat{f_1} = \sum_{j=0}^{4-1} f_j \omega_4^{\mathrm{j}1} = [1 \quad \omega_4 \quad \omega_4^2 \quad \omega_4^3]$$

When k=3

$$\widehat{f_2} = \sum_{j=0}^{4-1} f_j \omega_4^{\mathrm{j}2} = [1 \quad \omega_4^2 \quad \omega_4^4 \quad \omega_4^6]$$

When k=4

$$\widehat{f_3} = \sum_{j=0}^{4-1} f_j \omega_4^{\mathrm{j}3} = [1 \quad \omega_4^3 \quad \omega_4^6 \quad \omega_4^9]$$

$$\mathrm{DFT} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-\frac{2\pi i}{4}} & \left(e^{-\frac{2\pi i}{4}}\right)^2 & \left(e^{-\frac{2\pi i}{4}}\right)^3 \\ 1 & \left(e^{-\frac{2\pi i}{4}}\right)^2 & \left(e^{-\frac{2\pi i}{4}}\right)^4 & \left(e^{-\frac{2\pi i}{4}}\right)^6 \\ 1 & \left(e^{-\frac{2\pi i}{4}}\right)^3 & \left(e^{-\frac{2\pi i}{4}}\right)^6 & \left(e^{-\frac{2\pi i}{4}}\right)^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-\frac{2\pi i}{4}} & e^{-\frac{4\pi i}{4}} & e^{-\frac{6\pi i}{4}} \\ 1 & e^{-\frac{4\pi i}{4}} & e^{-\frac{8\pi i}{4}} & e^{-\frac{12\pi i}{4}} \\ 1 & e^{-\frac{6\pi i}{4}} & e^{-\frac{12\pi i}{4}} & e^{-\frac{18\pi i}{4}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-\frac{\pi i}{2}} & e^{-\pi i} & e^{-\frac{3\pi i}{2}} \\ 1 & e^{-\pi i} & e^{-2\pi i} & e^{-3\pi i} \\ 1 & e^{-\frac{3\pi i}{2}} & e^{-3\pi i} & e^{-\frac{9\pi i}{2}} \end{bmatrix}$$

iDFT: $F_N^{-1} = \frac{1}{N}\overline{F_N}$

$$F_N^{-1} = \frac{1}{4}\left\| \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-\frac{\pi i}{2}} & e^{-\pi i} & e^{-\frac{3\pi i}{2}} \\ 1 & e^{-\pi i} & e^{-2\pi i} & e^{-3\pi i} \\ 1 & e^{-\frac{3\pi i}{2}} & e^{-3\pi i} & e^{-\frac{9\pi i}{2}} \end{bmatrix} \right\|$$

```
N=4
```

```
N = 4
```

```
F_N=zeros(N,N)
```

```
F_N = 4×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
for j=1:N
    for k=1:N
        F_N(j,k)=(exp(-2*pi*i/N))^((j-1)*(k-1));
    end
end
F_N
```

```
F_N = 4×4 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
```

```
iF_N=1/4*conj(F_N)
```

```
iF_N = 4×4 complex
   0.2500 + 0.0000i   0.2500 + 0.0000i   0.2500 + 0.0000i   0.2500 + 0.0000i
   0.2500 + 0.0000i   0.0000 + 0.2500i  -0.2500 + 0.0000i  -0.0000 - 0.2500i
   0.2500 + 0.0000i  -0.2500 + 0.0000i   0.2500 - 0.0000i  -0.2500 + 0.0000i
   0.2500 + 0.0000i  -0.0000 - 0.2500i  -0.2500 + 0.0000i   0.0000 + 0.2500i
```

```
iF_N*F_N
```

```
ans = 4×4 complex
   1.0000 + 0.0000i  -0.0000 - 0.000
  -0.0000 + 0.0000i   1.0000 + 0.000
   0.0000 + 0.0000i  -0.0000 + 0.000
   0.0000 + 0.0000i   0.0000 + 0.000
```

problem 2

```
N=4
```

```
N = 4
```

```
F_N=zeros(N,N)
```

```
F_N = 4×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
for j=1:N
    for k=1:N
        F_N(j,k)=(exp(-2*pi*i/N))^((j-1)*(k-1));
    end
end
F_N
```

```
F_N = 4×4 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
```

```
iF_N=1/4*conj(F_N)
```

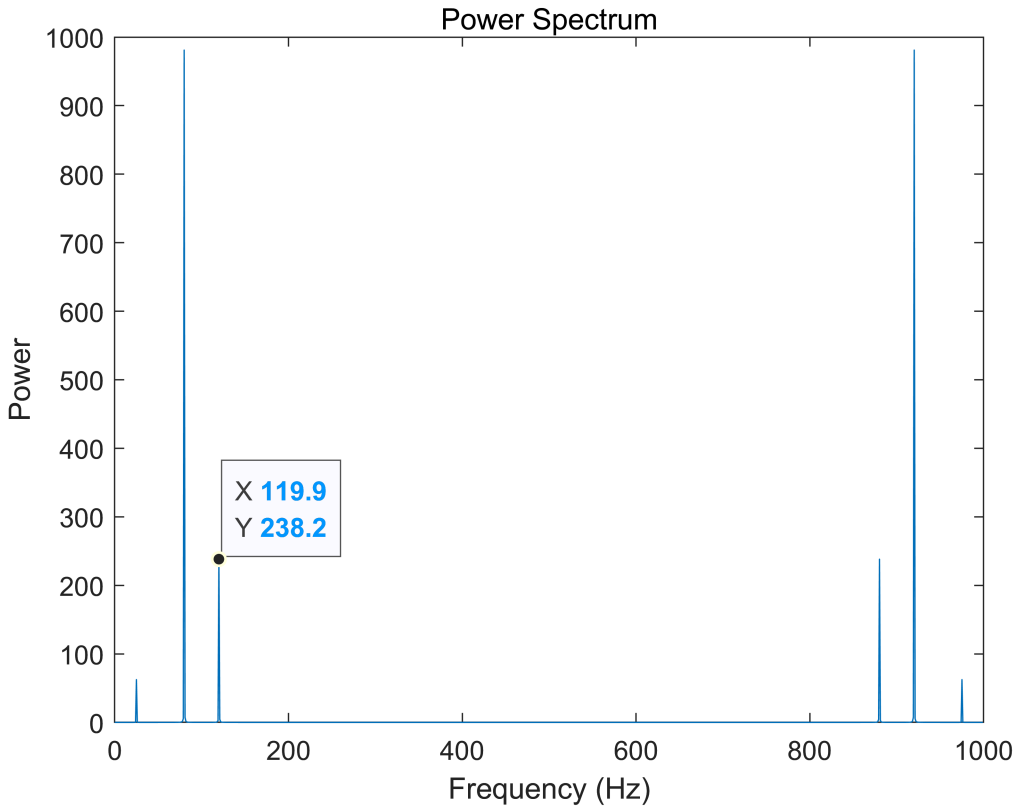
```
iF_N = 4×4 complex
   0.2500 + 0.0000i   0.2500 + 0.0000i   0.2500 + 0.0000i   0.2500 + 0.0000i
   0.2500 + 0.0000i   0.0000 + 0.2500i  -0.2500 + 0.0000i  -0.0000 - 0.2500i
   0.2500 + 0.0000i  -0.2500 + 0.0000i   0.2500 - 0.0000i  -0.2500 + 0.0000i
   0.2500 + 0.0000i  -0.0000 - 0.2500i  -0.2500 + 0.0000i   0.0000 + 0.2500i
```

```
iF_N*F_N
```

```
ans = 4×4 complex
   1.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i
  -0.0000 + 0.0000i   1.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 - 0.0000i
   0.0000 + 0.0000i  -0.0000 + 0.0000i   1.0000 + 0.0000i  -0.0000 - 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   1.0000 + 0.0000i
```

problem 3

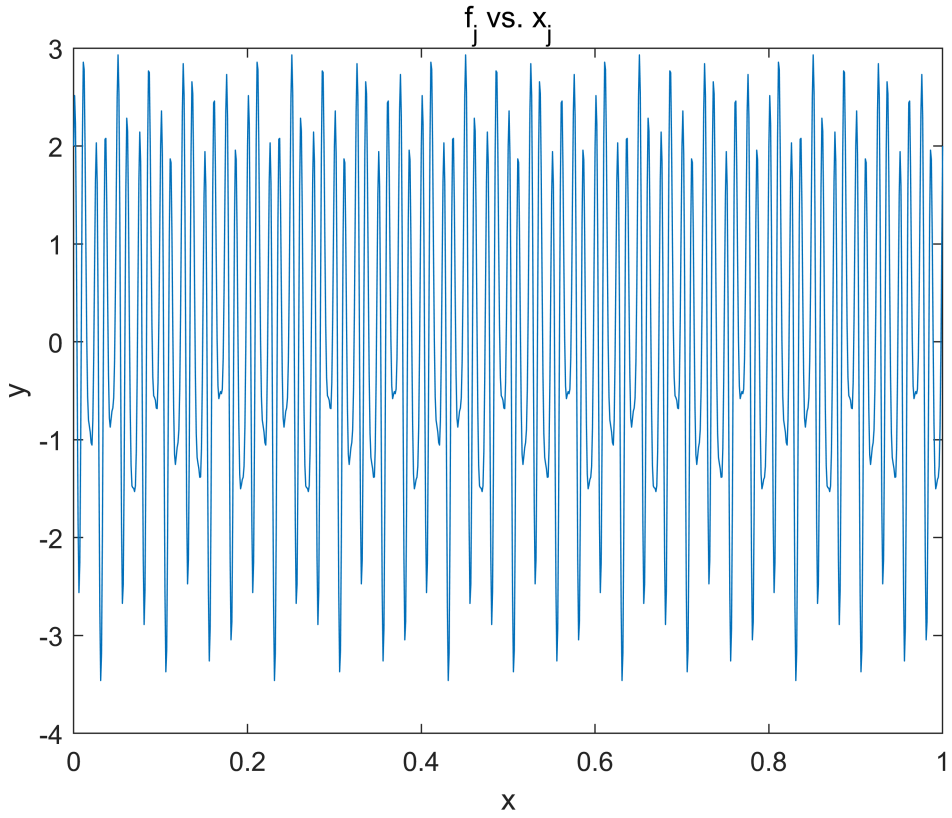
```
clear
load DFT_example_freqData.mat

N=length(x);
psd=F.*conj(F)/N;%compute the spectrum
dx=x(2)-x(1);
freq_axis = 1/(dx*N) * [0:N-1];

plot(freq_axis,psd)
title('Power Spectrum')
xlabel('Frequency (Hz)')
ylabel('Power')
```



By focus on half of the plot, the spectrums are 24.98, 79.92 and 119.9.

```
%compute the fft matrix
F_N=zeros(N,N);
for j=1:N
    for k=1:N
        F_N(j,k)=(exp(-2*pi*1i/N))^((j-1)*(k-1));
    end
end

%recovering f_j
fj=F*F_N^(-1);
```

```
plot(x,real(fj))
title('f_j vs. x_j')
xlabel('x')
ylabel('f')
```



```
%problem 1
```

f

```
clear
N8=8
```

```
N8 = 8
```

```
fn8=fftt(N8)
```

```
fn8 = 8×8 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i ···
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 + 1.0000i  -0.7071 + 0.7071i
```

$N = 8$

$\omega_8 = e^{\frac{-2\pi i}{8}} = e^{\frac{-\pi i}{4}}$

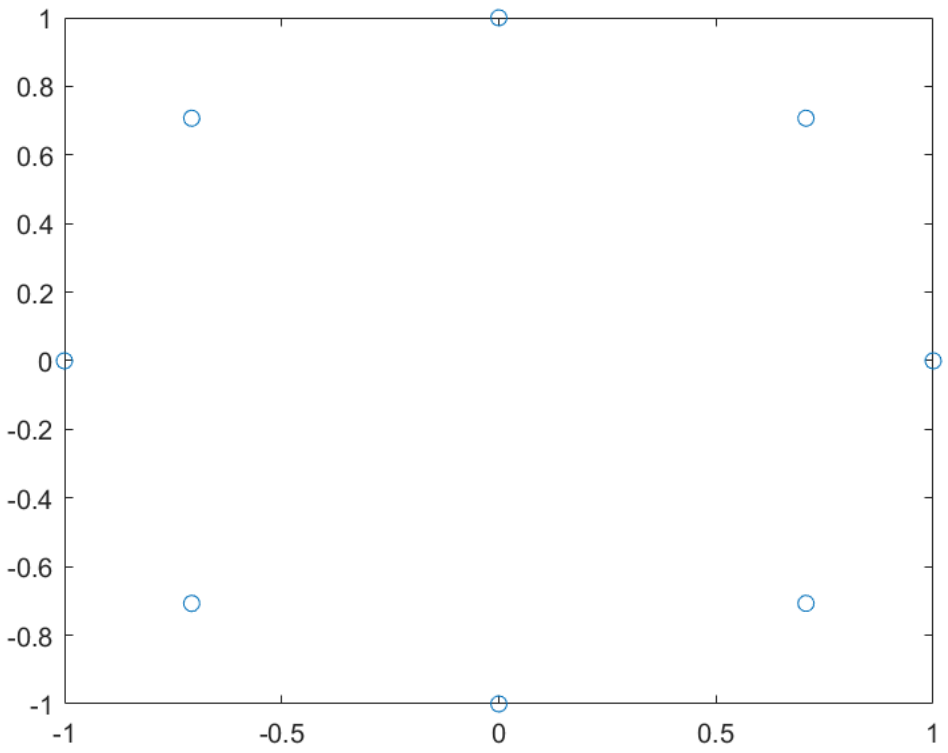
```
x8=0:1:(N8-1)^2
```

```
x8 = 1×50
     0     1     2     3     4     5     6     7     8     9    10    11    12 ···
```

```
y8=exp(-pi*1i/4).^x8
```

```
y8 = 1×50 complex
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i ···
```

```
plot(y8,'o')
```




```
N4=4
```

```
N4 = 4
```

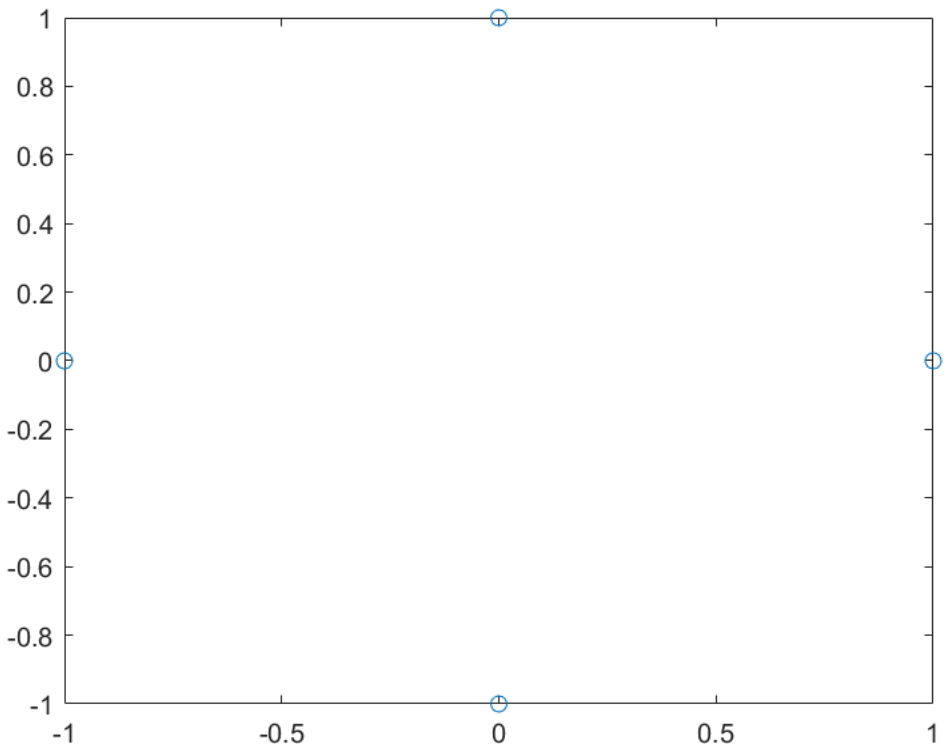
```
x4=0:1:(N4-1)^2
```

```
x4 = 1×10
     0     1     2     3     4     5     6     7     8     9
```

```
y4=exp(-pi*1i/2).^x4
```

```
y4 = 1×10 complex
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i ···
```

```
plot(y4,'o')
```




```
%F8-F4

N4=4
```

```
N4 = 4
```

```
fn4=fftt(N4)
```

```
fn4 = 4×4 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
```

```
B8=BB(N8)
```

```
B8 = 8×8
     1     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     1     0
     0     1     0     0     0     0     0     0
     0     0     0     1     0     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     1
```

```
k8=kk(N8,fn4)
```

```
k8 = 8×8 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i ···
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
```

```
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
D8=[eye(N4,N4) diag(y8(1:N4));
   eye(N4,N4) -diag(y8(1:N4))]
```

```
D8 = 8×8 complex
   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

```
tm=D8*k8*B8
```

```
tm = 8×8 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i ···
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -1.0000 + 0.0000i   1.0000 + 0.0000i  -1.0000 + 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 + 1.0000i  -0.7071 + 0.7071i
```

```
tm-fn8
```

```
ans = 8×8 complex
10^-14 ×
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0161 + 0.0000i  -0.0111 + 0.0111i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0322i   0.0322 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0482 + 0.0000i  -0.0444 - 0.0444i
   0.0000 + 0.0000i   0.0000 + 0.0444i   0.0000 - 0.0888i   0.0000 + 0.1332i
   0.0000 + 0.0000i   0.0333 + 0.0333i  -0.1049 + 0.0000i   0.1110 - 0.1110i
   0.0000 + 0.0000i   0.0444 + 0.0000i   0.0000 + 0.1210i  -0.1654 + 0.0000i
   0.0000 + 0.0000i   0.0333 - 0.0333i   0.1371 + 0.0000i   0.1332 + 0.1332i
```

```
N2=2
```

```
N2 = 2
```

```
fn2=fftt(N2)
```

```
fn2 = 2×2 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i
```

```
k=kk(N8,fn2)
```

```
k = 8×8 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
   1.0000 + 0.0000i  -1.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
B4=BB(N4)
```

```
B4 = 4×4
     1     0     0     0
     0     0     1     0
     0     1     0     0
     0     0     0     1
```

```
D4=[eye(N2,N2) diag(y4(1:N2));
   eye(N2,N2) -diag(y4(1:N2))]
```

```
D4 = 4×4 complex
   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 1.0000i
```

```
D=[D4 zeros(N4)
  zeros(N4) D4]
```

```
D = 8×8 complex
   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i ···
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 1.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
B=[B4 zeros(N4)
    zeros(N4) B4]
```

```
B = 8×8
     1     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     0     1     0     0     0     0
     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     1
```

```
tm2=D*k*B8
```

```
tm2 = 8×8 complex
   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i ···
   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 - 0.0000i   0.0000 + 0.0000i
```

```
   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 - 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i  -1.0000 - 0.0000i
```

fn8-tm2

```
ans = 8×8 complex
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i ···
   0.0000 + 0.0000i   0.7071 - 0.7071i   1.0000 - 1.0000i  -0.7071 - 0.7071i
   0.0000 + 0.0000i   0.0000 - 1.0000i  -2.0000 - 0.0000i  -0.0000 + 1.0000i
   0.0000 + 0.0000i  -0.7071 - 0.7071i   1.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -2.0000 - 0.0000i   1.0000 + 0.0000i  -2.0000 - 0.0000i
   1.0000 + 0.0000i  -1.7071 + 0.7071i   0.0000 - 1.0000i   1.7071 + 0.7071i
   1.0000 + 0.0000i  -1.0000 + 1.0000i  -1.0000 - 0.0000i  -1.0000 - 1.0000i
   1.0000 + 0.0000i  -0.2929 + 0.7071i  -0.0000 + 1.0000i   0.2929 + 0.7071i
```

```
function [F_N]=fftt(N)
    F_N=zeros(N,N);
    for j=1:N
        for k=1:N
            F_N(j,k)=(exp(-2*pi*1i/N))^((j-1)*(k-1));
        end
    end
end

function [B]=BB(N)
    B=zeros(N,N);
    for j=1:N/2
        B(j,2*j-1)=1;
    end
    for j=1:N/2
        B(j+N/2,2*j)=1;
    end
end

function [k]=kk(N,fn)
    [m,n]=size(fn);
    k=zeros(N,N);
    for j=1:N/m
        k((j-1)*m+1:(j-1)*m+1+m-1,(j-1)*n+1:(j-1)*n+1+n-1)=fn;
    end

end
```

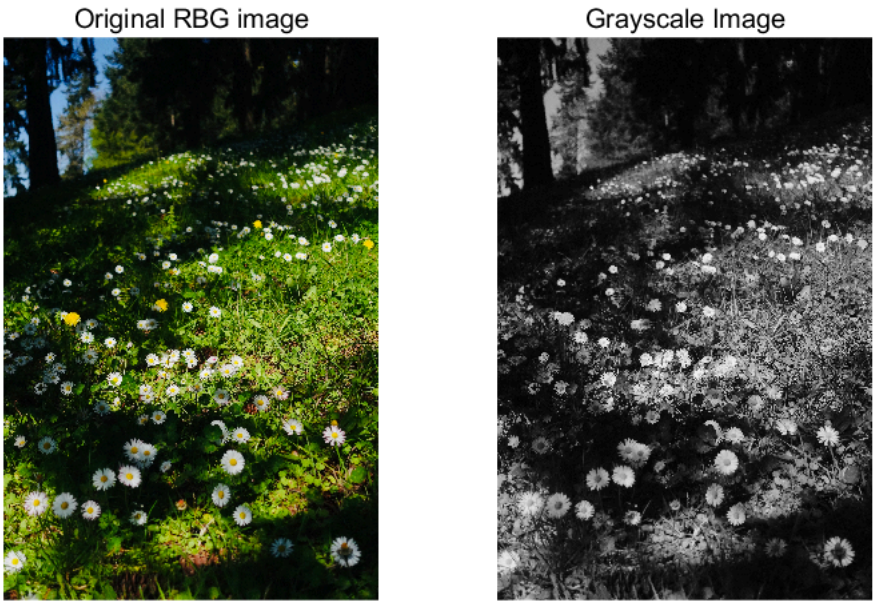
```
%problem 2
clear all; close all; clc;

A = imread('C:\Users\shuaizhouWang\Desktop\6030\img\en.jpg');
size(A)
```

```
ans = 1×3
        3888        2592           3
```

```


figure;
subplot(1,2,1)
imshow(A);
title('Original RBG image', 'FontSize', 18)

Abw = rgb2gray(A); % turn the RGB image into a grayscale image
size(Abw)
```

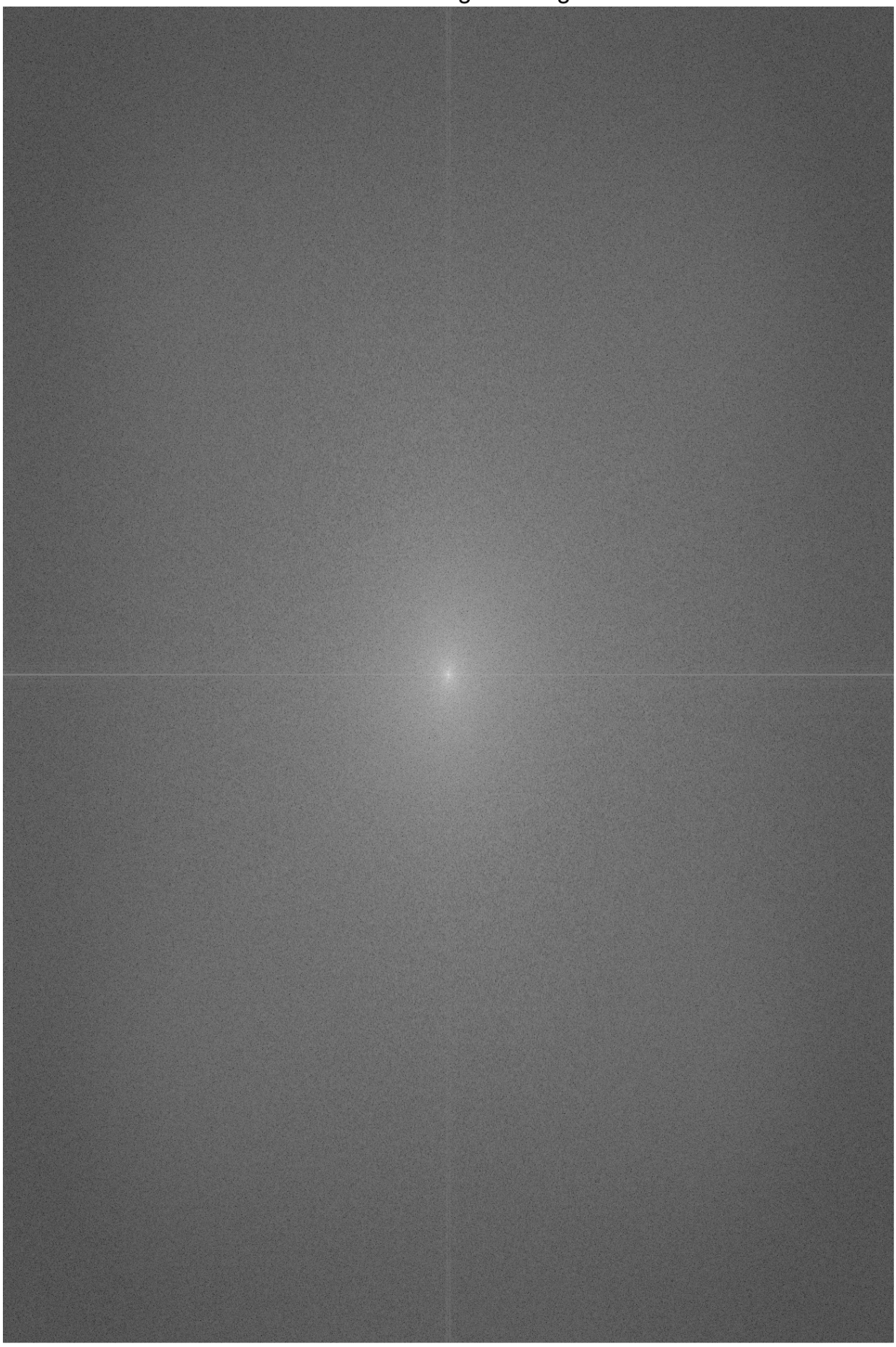
```
ans = 1×2
        3888        2592
```

```
subplot(1,2,2)
imshow(Abw)
title('Grayscale Image', 'FontSize', 18)
figure(3);
subplot(2,2,1)
imshow(Abw)
title('Grayscale Image', 'FontSize', 18)


plot_ind    = 1; % used for subplot index tracking
[nx,ny]     = size(Abw); % identify number of rows and columns in A
N_entries   = nx*ny;

A_fft = fft2(Abw); % take the 2D fft of the image
F     = log(abs(fftshift(A_fft))+1); % helpful for visualizing the data in frequency domain
F     = mat2gray(F); % scaling image for visualizing in grayscale

figure;
imshow(F, []);
```

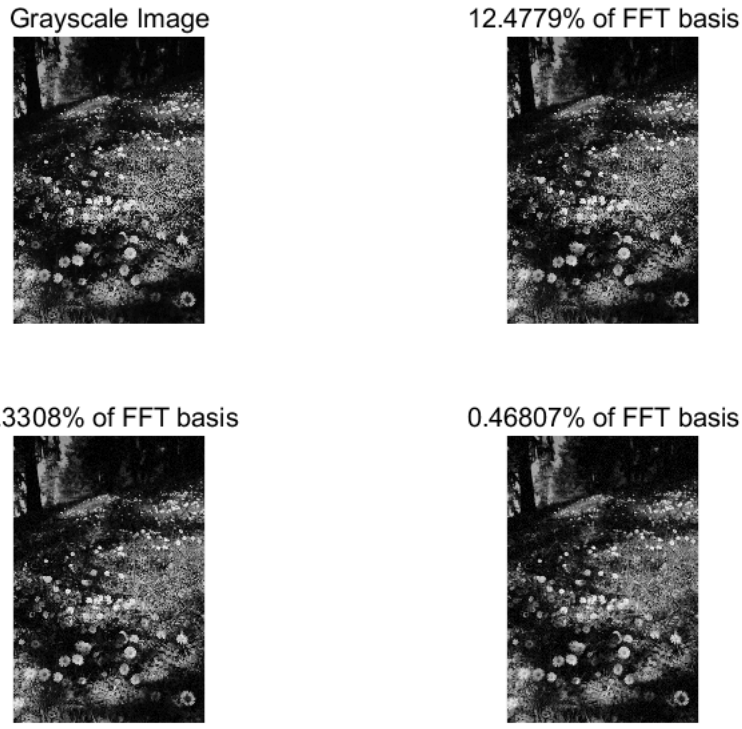



```
title('2D FFT of original image', 'FontSize', 18)
```

2D FFT of original image

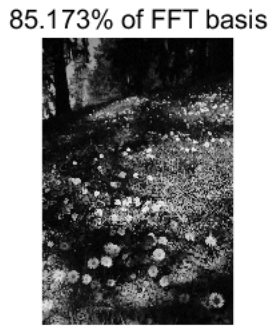


```
for thresh = 0.01*[0.01 0.05 0.1]*max(abs(A_fft(:))) % loop over 3 levels of compression (filte
    plot_ind        = plot_ind + 1; % this is just used to identify where in our subplot to plo
    inds            = abs(A_fft)>thresh; % identify which indices correspond to magnitudes of t
    A_fft_filter    = A_fft.*inds; % set all such corresponding entries to zero
    filt_count      = N_entries - nnz(A_fft_filter); % count the number of entries filtered out
    filt_percent    = 100 - filt_count/N_entries*100; % compute the percentage of filtering per

    % Transform the filtered data back to physical (image) space using the iFFT
    A_filter        = uint8(ifft2(A_fft_filter)); % the uint8 just puts the data in a format th


    % Plot the filtered image in the subplot
    figure(3);
    subplot(2,2,plot_ind)
    imshow(A_filter)
    drawnow; % show the image as the code is running
    title([num2str(filt_percent) '% of FFT basis'], 'FontSize', 18)
end
```




```
plot_ind    = 1; % used for subplot index tracking
[nx,ny]     = size(Abw); % identify number of rows and columns in A
N_entries   = nx*ny;


for thresh = 0.01*[0.0001 0.0005 0.001]*max(abs(A_fft(:))) % loop over 3 levels of compression
    plot_ind        = plot_ind + 1; % this is just used to identify where in our subplot to plo
```

```
    inds            = abs(A_fft)>thresh; % identify which indices correspond to magnitudes of t
    A_fft_filter    = A_fft.*inds; % set all such corresponding entries to zero
    filt_count      = N_entries - nnz(A_fft_filter); % count the number of entries filtered out
    filt_percent    = 100 - filt_count/N_entries*100; % compute the percentage of filtering per

    % Transform the filtered data back to physical (image) space using the iFFT
    A_filter        = uint8(ifft2(A_fft_filter)); % the uint8 just puts the data in a format th


    % Plot the filtered image in the subplot
    figure(4);
    subplot(2,2,plot_ind)
    imshow(A_filter)
    drawnow; % show the image as the code is running
    title([num2str(filt_percent) '% of FFT basis'], 'FontSize', 18)
end
```



problem 3

$4t^3 - 3t + \sin(t)$

$f(t) = 4t^3 - 3t + \sin(t)$

$f(t) = \sum_{n=0}^{8} c_n h_n(t)$

$c_n = \langle h_n, f(t) \rangle$

$\int_0^1 h_0(t) f(t)\,dt \quad \int_0^1 4t^3 - 3t + \sin(t)\,dt = -0.04030$

$\int_0^1 h_1(t) f(t)\,dt = \int_0^{\frac{1}{2}} 4t^3 - 3t + \sin(t)\,dt + \int_{\frac{1}{2}}^1 -4t^3 + 3t - \sin(t)\,dt$

$= -0.19008 - 0.14978 = -0.33986$

$\int_0^1 h_2(t) f(t)\,dt = \int_0^{1/4} f(t)\,dt + \int_{1/4}^{1/2} -f(t)\,dt = -0.05876 + 0.13133$

$\int_0^1 h_3(t) f(t)\,dt = \int_{1/2}^{3/4} f(t)\,dt + \int_{3/4}^1 -f(t)\,dt = -0.06994 + (-0.21873)$

$\int_0^1 h_4(t) f(t)\,dt = \int_0^{1/8} f(t) + \int_{1/8}^{2/8} -f(t) = -0.015391 + 0.04327$

$h_5 \quad \int_{2/8}^{3/8} + \int_{3/8}^{4/8} = -0.06291 + 0.06841$

$h_6 \quad \int_{4/8}^{5/8} + \int_{5/8}^{6/8} = -0.05423 + 0.01472$

$h_7 \quad \int_{6/8}^{7/8} + \int_{7/8}^{1} = 0.0558 + (-0.16291)$

$c_n = -0.04030 \quad -0.33986 \quad 0.07257 \quad -0.28768$

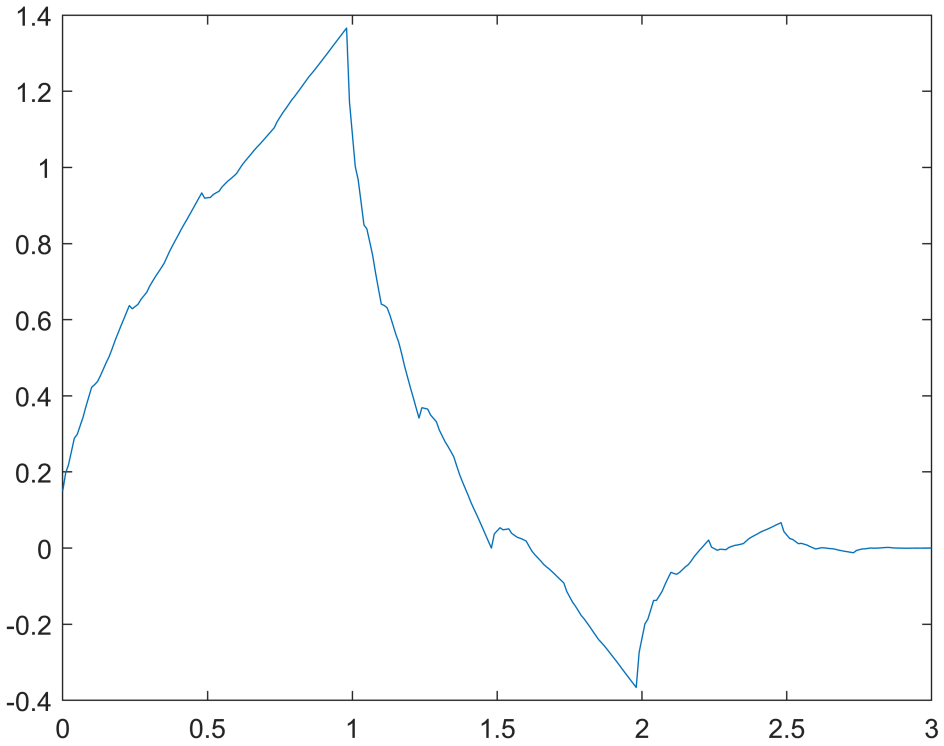
$0.027979 \quad 0.0055 \quad -0.03951 \quad -0.10715$

```
clear
```

```
%problem 4

h1 = [1+sqrt(3),3+sqrt(3),3-sqrt(3),1-sqrt(3)]/8
```

```
h1 = 1×4
    0.3415    0.5915    0.1585   -0.0915
```

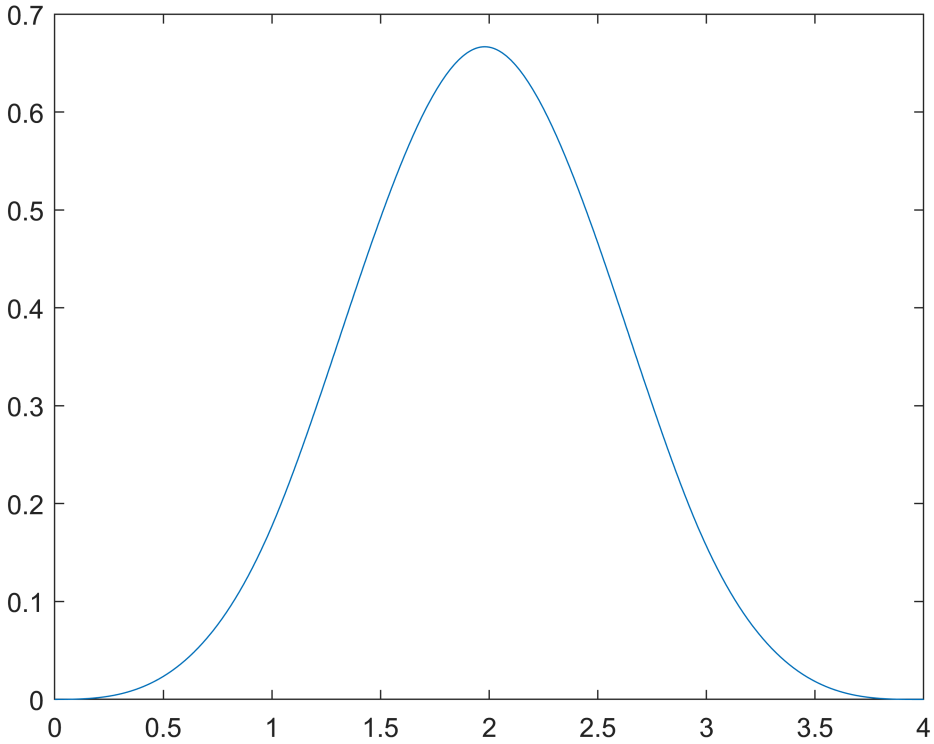
```
cascade(h1)
```




```
h2 = [1,4,6,4,1]/16
```

```
h2 = 1×5
    0.0625    0.2500    0.3750    0.2500    0.0625
```

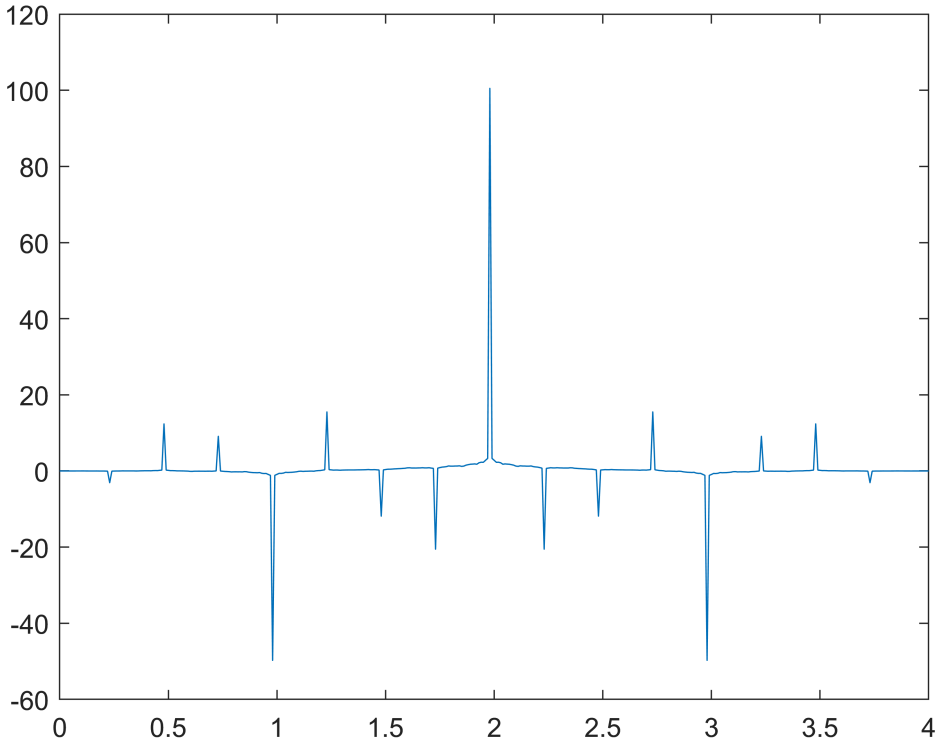
```
cascade(h2)
```




```
h3 = [-1,2,6,2,-1]/8
```

```
h3 = 1×5
   -0.1250    0.2500    0.7500    0.2500   -0.1250
```

```
cascade(h3)
```



错误使用 Untitled2>cascade (line 34)
divergence

```
function []=cascade(h)
    n      = length(h)-1;
    tsplit = 100;
    tt     = 0:1/tsplit:n;
    ntt    = length(tt);
    phi    = double(tt<1);

    while 1 % Iterate until convergence or divergence
        phinew=0*phi;

        for j=1:ntt
            for k=0:n
                index=2*j-k*tsplit+1;

                if index>=1 && index<=n*tsplit+1
                    phinew(j)=phinew(j)+2*h(k+1)*phi(index);
                end
            end
        end

        plot(tt,phinew),pause(1e-1)

        if max(abs(phinew))>100
            error('divergence');
        end
```

```
        if max(abs(phinew-phi))<1e-3
            break;
        end

        phi=phinew;
    end

end
```