$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{+j2\pi(ux/M+vy/N)}$$

$u=0,\ v=0$

$F(0,0)e^{0}$

1	1
1	1

$u=1,\ v=0$

$F(1,0)e^{j2\pi\frac{x}{2}}$

1	-1
1	-1

$u=0,\ v=1$

$F(0,1)e^{j2\pi\frac{y}{2}}$

1	1
-1	-1

$u=1,\ v=1$

$F(1,1)e^{j2\pi(\frac{x}{2}+\frac{y}{2})}$

1	-1
-1	1

۲	۳
1	۴

$= f_{0,0}$ [1 1; 1 1] $+ f_{1,0}$ [1 -1; 1 -1] $+ f_{0,1}$ [1 1; -1 -1] $+ f_{1,1}$ [1 -1; -1 1]

$$۲ = f_{0,0}+f_{1,0}+f_{0,1}+f_{1,1}$$
$$۳ = f_{0,0}-f_{1,0}+f_{0,1}-f_{1,1}$$
$$1 = f_{0,0}+f_{1,0}-f_{0,1}-f_{1,1}$$
$$۴ = f_{0,0}-f_{1,0}-f_{0,1}+f_{1,1}$$

$$\Rightarrow \begin{matrix} f_{0,0}+f_{1,1}=3 \\ f_{0,0}-f_{1,1}=2 \end{matrix}$$

$$\Rightarrow f_{0,0}=\frac{5}{2},\ f_{1,1}=\frac{1}{2}$$

$$f_{1,0}+f_{0,1}=-1$$
$$-f_{1,0}+f_{0,1}=1$$
$$\Rightarrow f_{0,1}=0,\ f_{1,0}=-1$$

2	3
1	4

$= \frac{5}{2}$ [1 1; 1 1] -1 [1 -1; 1 -1] $+0+\frac{1}{2}$ [1 -1; -1 1]

$$\Rightarrow F = \begin{matrix} \frac{5}{2} & 0 \\ -1 & \frac{1}{2} \end{matrix}$$

۲.الف. از آن‌جایی که هیچ اطلاعاتی نسبت به تصویر نداریم بخش حقیقی دامنه فرکانسی می‌تواند هر مقداری داشته باشد و نمی‌دانیم کدام مقادیر صفر یا ثابت هستند. پس در حالت کلی ابعاد قسمت حقیقی دامنه فرکانسی در فضا برای یک تصویر n در n، n^2 است.

۲.ب. برابر میانگین شدت روشنایی (رنگ) پیکسل‌های تصویر است.

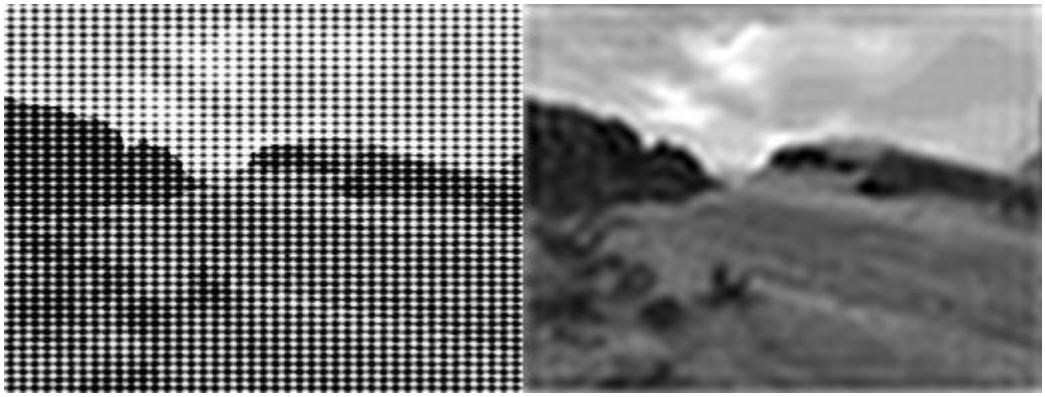
$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

$$\Rightarrow F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

۳.پ. بدلیل وجود نویز نمک و فلفل شدیدتر در تصویر داده شده، وقتی فیلتر میانگین‌گیر اعمال می‌شود اثر تارشدگی بدی ایجاد می‌شود و انگار نقاط نویزی بجای اینکه پاک شوند، پخش می‌شوند چون مقدار نویز (یا همان داده‌پرت) با بقیه مقادیر سالم تصویر میانگین گرفته می‌شود و روی آن مقادیر سالم اثر می‌گذارد. وقتی نویز نمک و فلفل شدید نباشد این اثر قابل چشم‌پوشی است و زیاد حس نمی‌شود اما در تصویر داده شده به‌خوبی قابل مشاهده است و اصلا مطلوب نیست. در حالی که وقتی از فیلتر میانه‌گیر استفاده می‌شود چون داده‌های پرت بعد از مرتب‌کردن چند پیکسل مجاور در ابتدا یا انتهای پیکسل‌ها قرار می‌گیرد وقتی میانه می‌گیریم با احتمال بسیار زیادی پیکسل داده پرت حذف می‌شود (مگر اینکه تعداد پیکسل‌های نویزی در یک مجاورت انتخاب‌شده بسیار زیاد باشد که میانه هم از پیکسل‌های نویزی انتخاب شود) ایراد فیلتر میانه‌گیر در قسمت‌های نازک تصویر است که ممکن است به علت اقلیت بودن پیکسل‌های آن شی نازک تصویر، شی را به‌طور کامل از دست بدهیم و مقداری از جزئیات تصویر پاک شوند.

۳.ت. برای رهایی از نویز می‌توانیم قبل از اینکه فیلتر مشتق‌گیر را اعمال کنیم از فیلترهای رفع نویز (مثل میانگین‌گیر یا میانه‌گیر) استفاده کنیم. برای اینکه فیلتر نویزگیر در کار فیلتر مشتق‌گیر اخلال ایجاد نکند و تغییرات ناگهانی رنگ تصویر را از بین نَبَرَد، می‌توانیم در راستای عمود بر راستای اصلی، نویزگیر را اجرا کنیم. مثلا اگر می‌خواهیم در راستای افقی از تصویر مشتق بگیریم می‌توانیم فقط در راستای عمودی نویزگیر را اجرا کنیم. (مثلا کرنل میانگین‌گیر را طوری تنظیم کنیم که فقط در راستای عمودی مقدار داشته باشد)

۴.الف.



تصویر حاصل در کنار تصویر نویزی در فایل *denoised_by_fft.png* ذخیره شده

```
denoised = image.copy()

height = denoised.shape[0]
width = denoised.shape[1]

# Convert to freq domain
denoised = np.fft.fft2(denoised)

# Remove 80% of middle freqs and keep 10% of beginning and 10% of ending freqs (because of circularity of freqs)
keep_ratio = 0.1
denoised[int(height * keep_ratio):int(height * (1 - keep_ratio))] = 0
denoised[:, int(width * keep_ratio):int(width * (1 - keep_ratio))] = 0

# Convert back to space domain
denoised = np.fft.ifft2(denoised).real
return denoised
```

بعد از کپی‌گرفتن از تصویر اصلی، به کمک *numpy.fft* تصویر را به فضای فرکانسی بردم. (*fft.fft2* تبدیل فوریه ۲بعدی است) سپس ۸۰٪ فرکانس‌های بالا را حذف کردم. از آن‌جایی که ماهیت فرکانس‌ها به خاطر یکی بودن *0* و *2π* حالت چرخشی (*circular*) دارد پس ۱۰٪ از فرکانس‌های نزدیک صفر و از ۱۰٪ فرکانس‌های انتهایی (که در واقع بسیار نزدیک به صفر هستند) را نگه داشتم و بقیه را حذف کردم. این کار را هم در راستای *x* و هم در راستای *y* انجام دادم تا قسمت‌های با فرکانس بالا که نمایانگر تغییرات شدید ناگهانی هستند (و احتمالا نویزها را شامل می‌شوند) حذف شوند و در نهایت نویزها از بین بروند. در نهایت پس از برگرداندن به دامنه مکان از بخش حقیقی خروجی استفاده کردم.

۴.ب. نسبت پیک سیگنال به نویز در کاربردهای پردازش تصویر برای سنجش میزان نویزی بودن یا کیفیت رفع نویز بکار می‌رود. هرچه عدد بزرگ‌تری بدست آید یعنی تصویر شامل نویز کمتری است و اگر به ازای خود تصویر بدون هیچ تغییری بخوایم PSNR محاسبه کنیم از آن‌جایی که MSE صفر می‌شود به PSNR=100 خواهیم رسید که بهترین حالت است و هرچه کوچکتر از ۱۰۰ باشد یعنی توان نویز در تصویر داده شده بیشتر است.

پس از رفع نویز تصویر وقتی PSNR تصویر نویزی و تصویر رفع نویزشده را محاسبه می‌کنیم مشاهده می‌کنیم PSNR بعد از رفع نویز تقریبا سه برابر شده که نشان‌دهنده عملکرد مناسب تابع رفع نویز است.

```
PSNR between noisy image and original image =  8.122255096865844
PSNR between denoised image and original image =  24.804180799257377
```

۴.پ. نویز ضرب‌شونده یعنی نویزی که در آن مقدار سیگنال خالص در فرآیند نویزی شدن ضرب‌در مقدار سیگنالی نامطلوب می‌شود در حالی که در نویز جمع‌شونده مقدار سیگنال اصلی با سیگنالی دیگر جمع می‌شود.

```
for i in range(X):
  for j in range(Y):
    noise[i,j] = f(i,j)*100

noisy_image = original_image + noise
```

در اینجا نویز اضافه شده از نوع جمع شونده است چون بعد از ساخته شدن سیگنال اضافی نویز با تصویر اصلی جمع شده نه اینکه در مقادیر آن ضرب شود.

منابع:

سوال ۴.الف: numpy.fft.fft2 — NumPy v1.23 Manual

numpy.fft.ifft2 —NumPy v1.23 Manual

Image denoising by FFT — Scipy lecture notes (scipy-lectures.org)

سوال ۴.ب: Python | Peak Signal-to-Noise Ratio (PSNR) - GeeksforGeeks

نسبت سیگنال به نویز چیست؟ — از صفر تا صد – فرادرس - مجله(faradars.org)

سوال ۴.پ: What are additive and multiplicative noise? - Quick Image Processing Research Guide (gofastresearch.com)