

d) Fourier transform

$$r(t) = \begin{cases} k, & |t-c| \leq a/2 \\ 0, & |t-c| > a/2 \end{cases}$$

Property of Fourier transform:

$$f(t) \rightarrow F(\omega), \text{ for } f(t-T) \rightarrow F(\omega) \cdot e^{-i\omega T}$$

$$f(t) = \begin{cases} k, & |t| \leq a/2 \\ 0, & |t| > a/2 \end{cases}$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt = \int_{-a/2}^{a/2} k e^{-i\omega t} dt = \frac{k}{i\omega} \left[e^{\frac{i\omega a}{2}} - e^{-\frac{i\omega a}{2}} \right] =$$

$$= ka \operatorname{sinc}\left(\frac{\omega a}{2}\right)$$

$$f(t) = \begin{cases} k, & |t-c| \leq a/2 \\ 0, & |t-c| > a/2 \end{cases} \Rightarrow F(\omega) = ka \operatorname{sinc}\left(\frac{\omega a}{2}\right) \cdot e^{-i\omega c}$$



~~the~~ the frequency spectrum looks like a sinc function in both x and y directions. The inverse transform of a sinc is a rectangle. Since the lines in the frequency spectrum are thin, the original function should be a rectangular dot (or a very small square).

c) Convolution theorem: the Fourier transform of a convolution of two functions is the pointwise product of their Fourier transform.

The Fourier transform of a delta function $\delta(x) = \begin{cases} 1, & x=0 \\ 0, & x \neq 0 \end{cases}$ is 1, so its convolution with a function or image will not change the frequency of a signal or image.

d) $\begin{pmatrix} 3 & 1 & -9 & -2 & 0 \\ 5 & 2 & 2 & 3 & -1 \\ 9 & 4 & -3 & -8 & 1 \\ 2 & 10 & -20 & -20 & 0 \\ 4 & 8 & 4 & -6 & 0 \end{pmatrix}$ - this matrix is not separable,
its rank is 5 and it will have 5
non-zero singular values (
(to be separable, rank should be 1)

e) $\begin{pmatrix} -21 & 6 & 3 & -12 & 9 \\ 7 & -2 & -1 & -4 & -3 \\ 0 & 0 & 0 & 0 & 0 \\ 35 & -10 & -5 & -20 & -15 \\ -14 & 4 & 2 & 8 & 6 \end{pmatrix}$ - this matrix is separable, its
rank is 1, it has 1 singular value

```
matrix = [
[-21, 6, 3, 12, 9],
[7, -2, -1, -4, -3],
[0, 0, 0, 0, 0],
[35, -10, -5, -20, -15],
[-14, 4, 2, 8, 6]
]
a, b, c = np.linalg.svd(matrix)

value = b[0]
v1 = a[:, 0] * np.sqrt(value)
v2 = c[0] * np.sqrt(value)

print('v1', v1)
print('v2', v2)
print('v1 x v2\n', np.round(np.outer(v1, v2), 2))
```

```
v1 [-3.57900146  1.19300049  0.          5.96500244 -2.38600097]
v2 [ 5.86755837 -1.67644525 -0.83822262 -3.3528905  -2.51466787]
v1 x v2
[[-21.   6.   3.  12.   9.]
 [ 7.  -2.  -1.  -4.  -3.]
 [ 0.  -0.  -0.  -0.  -0.]
 [35. -10.  -5. -20. -15.]
 [-14.   4.   2.   8.   6.]]
```


i) Distance transform 1D:

1) initialize $D[j] \rightarrow 0$ if j in P , otherwise ∞

2) Forward: for j from 1 to $n-1$

$$D[j] = \min(D[j], D[j-1] + 1)$$

3) backward: for j from $n-2$ to 0

$$D[j] = \min(D[j], D[j+1] + 1)$$

2D case: Fwd pass finds closest above and to left
Bwd finds closest below and to right.

∞	0	0	0	0	0	0
∞	∞	∞	0	0	∞	∞
∞	∞	∞	0	0	∞	∞
∞	0	0	0	0	0	∞
∞	0	0	∞	∞	0	∞
∞	0	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

init

∞	0	0	0	0	0	0
∞	1	1	0	0	1	1
∞	2	2	0	0	1	2
∞	3	0	0	0	0	1
∞	0	0	1	1	0	1
∞	0	1	2	2	1	2
∞	1	2	3	3	2	3

Fwd

1	0	0	0	0	0	0
2	1	1	0	0	1	1
3	2	1	0	0	1	2
2	1	0	0	0	0	1
1	0	0	1	1	0	1
1	0	1	2	2	1	2
2	1	2	3	3	2	3

Bwd