66310837

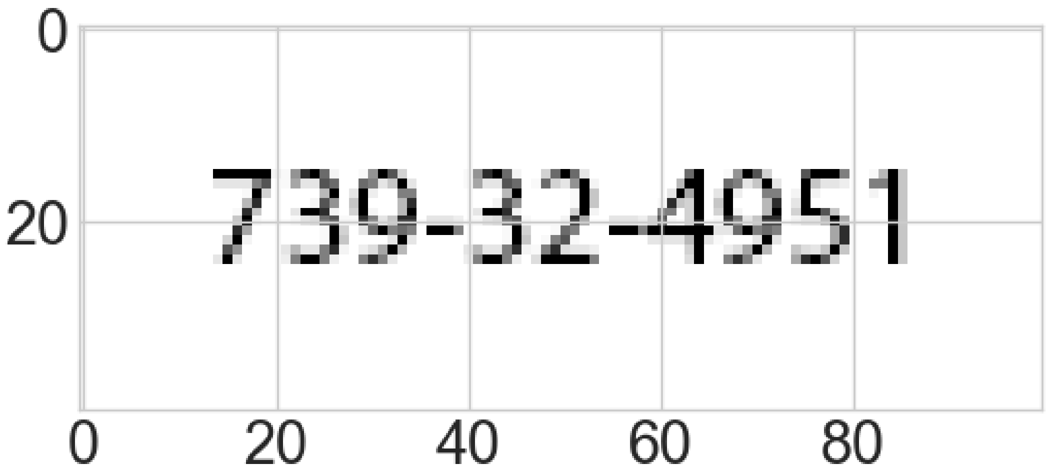
นายจิรัฐ ฟองดา

```
import numpy as np
import numpy.linalg as la
import scipy.linalg as sla
import scipy.sparse as sparse

import matplotlib.pyplot as plt
from PIL import Image
from time import time

import seaborn as sns
sns.set(font_scale=2)
plt.style.use('seaborn-v0_8-whitegrid')

# plt.style.use('bmh')
# plt.style.use('dark_background')
# plt.style.use('fivethirtyeight')

img =Image.open('ssn.png')
xmat = (255-np.array(img).max(axis=2))/255

print(xmat.shape)
print(xmat.min(), xmat.max())
```

```
(40, 100)
0.0 1.0
```

```
plt.imshow(xmat)
```

```
<matplotlib.image.AxesImage at 0x226654f8110>
```



```
x = xmat.flatten()
print(x.shape)
```

```
(4000,)
```

```
def blur_operator(m, n, radius):
    imat, jmat = np.meshgrid(np.arange(m), np.arange(n),
indexing='ij')
    ivec = np.atleast_2d(imat.flatten())
    jvec = np.atleast_2d(jmat.flatten())
    A = np.fmax(0, 1 - np.sqrt((ivec.T - ivec)**2 + (jvec.T -
jvec)**2)/radius)
    A /= A.sum(axis=1)
    return A

A = blur_operator(xmat.shape[0], xmat.shape[1], 5)
A.shape
```

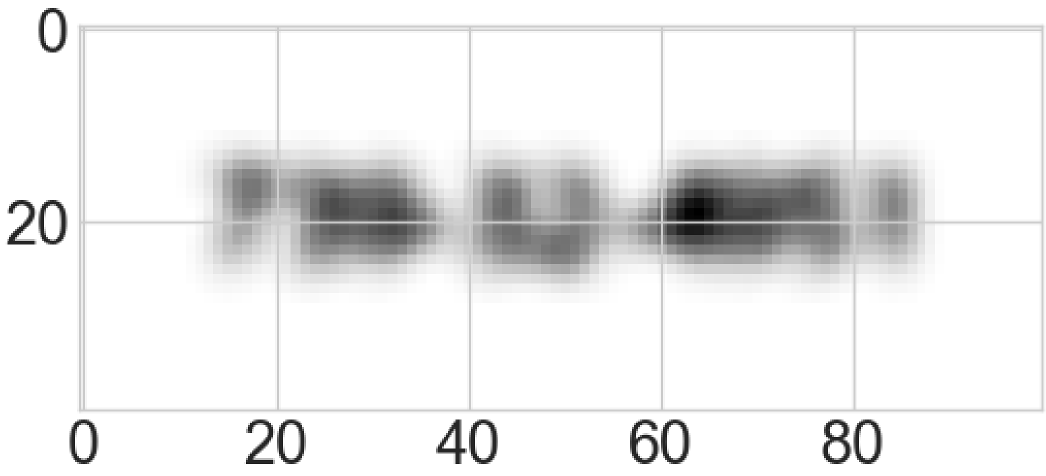
```
(4000, 4000)
```

```
b = A@x
b.shape
```

```
(4000,)
```

```
b2D = b.reshape(xmat.shape)
plt.imshow(b2D)
```

```
<matplotlib.image.AxesImage at 0x226658c7b30>
```

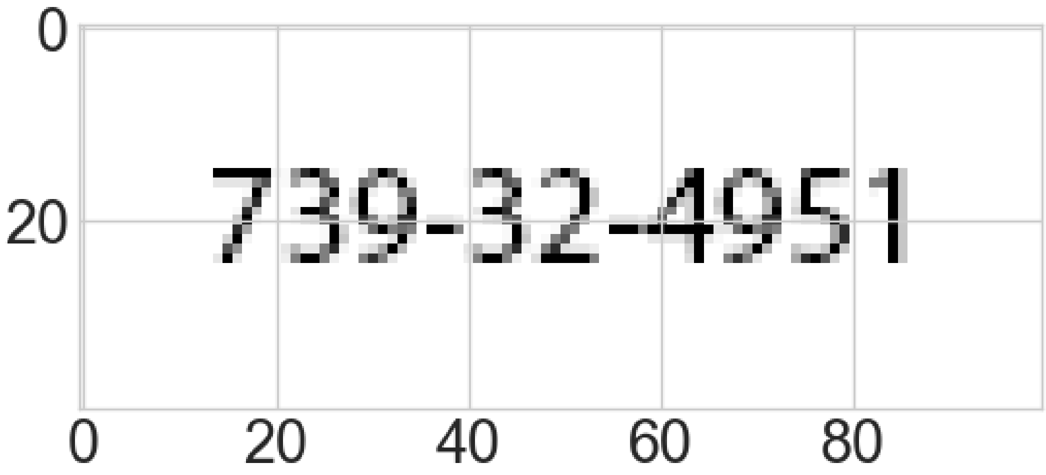


```
x_solve1 = la.solve(A, b)
x_solve1.shape
```

```
(4000,)
```

```
plt.imshow(x_solve1.reshape(xmat.shape))
```

```
<matplotlib.image.AxesImage at 0x22665a046e0>
```



```
xx = la.inv(A)@b
np.isclose(xx, x_solve1, rtol=1e-05, atol=1e-08,
equal_nan=False).sum() # ค่าในแต่ละเมทริกใกล้เคียงกันไหม
```

```
np.int64(4000)
```

```
b_noisy = b + 1e-05*np.random.rand(b.size)
x_noisy = la.solve(A, b)
```

```
M1 = np.array([[2,0,0,0], [3,2,0,0], [1,2,6,0], [1,3,4,2]])
b1 = np.array([2, 2, 6, 4])
```

```
M1
```

```
array([[2, 0, 0, 0],
       [3, 2, 0, 0],
       [1, 2, 6, 0],
       [1, 3, 4, 2]])
```

```
def my_forward_substitution(M, b):
    n = len(b)
    y = np.zeros(n)
    for i in range(n):
        tmp = b[i]
        for j in range(i):
            tmp -= y[j]*M[i, j]
        y[i] = tmp/M[i, i]
    return y
```

```
my_forward_substitution(M1, b1)
```

```
array([ 1.  , -0.5 ,  1.  ,  0.25])
```

```
sla.solve_triangular(M1, b1, lower=True)
```

```
array([ 1.  , -0.5 ,  1.  ,  0.25])
```

```
M2 = np.array([[2,8,4,2],[0,4,4,3],[0,0,6,2],[0,0,0,2]])
b2 = np.array([2,4,4,1])
```

```
M2
```

```
array([[2, 8, 4, 2],
       [0, 4, 4, 3],
       [0, 0, 6, 2],
       [0, 0, 0, 2]])
```

```
def my_backward_substitution(M, b):
    n = len(b)
    y = np.zeros(n)
    for i in range(n-1, -1, -1):
        tmp = b[i]
        for j in range(i+1, n):
            tmp -= y[j]*M[i,j]
        y[i] = tmp/M[i,i]
    return y
```

```
my_backward_substitution(M2, b2)
```

```
array([-1.   ,  0.125,  0.5  ,  0.5  ])
```

```
sla.solve_triangular(M2, b2)
```

```
array([-1.   ,  0.125,  0.5  ,  0.5  ])
```