In [1]:

```
import numpy as np
import math
import random
import pandas as pd
from cvxopt import matrix, solvers
from cvxopt.modeling import variable, op,sum, dot
import matplotlib.pyplot as plt
```

In [2]:

```
N = 20
d = 20
K = 20
S = np.zeros((N, K), dtype = float)
```

In [3]:

```
"""
Define a fucntion turn 0 into -1 and remain those 1's
"""
def function(a):
    if a == 0 :
        return -1
    else:
        return 1
```

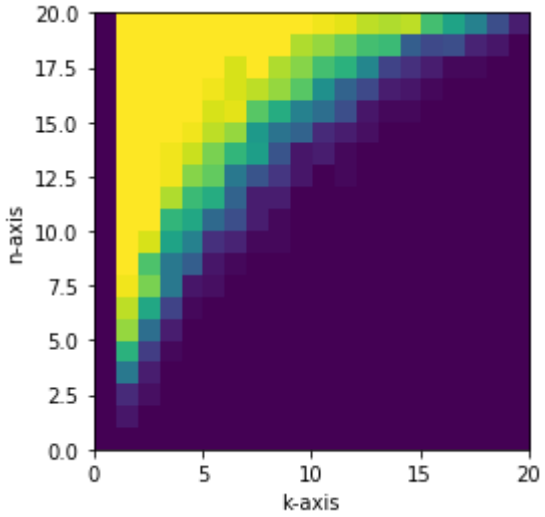
In [4]:

```
for n in range(1,N+1):
    A = np.random.normal(loc=0, scale=1, size=(n, d))
    for k in range(1, n+1):
        for i in range(1, 50+1):
            # Make a sparse x0
            x0 = np.zeros(d)
            t = random.sample(range(d), k)
            rand_bino = np.random.binomial(1, 0.5, k)
            result = map(function, rand_bino)
            result_list = list(result)
            x0[t]=result_list
            # Draw a standard Gaussian Random Matrix
            A = np.random.normal(loc=0, scale=1, size=(n, d))
            b = np.dot(A, x0)
            # = [-1 if x0[i]<0 else 1 for i in range(len(x0))]
            A = A.T
            A = matrix(A)
            b = matrix(b)
            #c = matrix(c)
            # Solve the linear programming problem
            x = variable(d)
            op(sum(abs(x)),[dot(A,x) == b]).solve()
            x = np.asarray(x.value)
            x = np.squeeze(x)
            dist = np.sqrt(np.sum(np.square(x-x0)))
            if dist <= 1e-3:
                S[n, k]+=1
```

```
 6:  3.3107e-01  3.3107e-01  1e-07  4e-08  2e-16  3e-09
Optimal solution found.
     pcost       dcost       gap    pres   dres   k/t
 0:  0.0000e+00 -0.0000e+00  2e+01  8e+00  1e-16  1e+00
 1:  2.9005e-01  2.8905e-01  3e+00  1e+00  2e-16  1e-01
 2:  9.9540e-01  9.9289e-01  1e+00  3e-01  6e-16  4e-02
 3:  9.6365e-01  9.5906e-01  5e-01  2e-01  5e-15  2e-02
 4:  9.9966e-01  9.9959e-01  7e-03  2e-03  4e-16  2e-04
 5:  1.0000e+00  1.0000e+00  7e-05  2e-05  6e-16  2e-06
 6:  1.0000e+00  1.0000e+00  7e-07  2e-07  9e-16  2e-08
 7:  1.0000e+00  1.0000e+00  7e-09  2e-09  4e-16  2e-10
Optimal solution found.
     pcost       dcost       gap    pres   dres   k/t
 0:  0.0000e+00 -0.0000e+00  2e+01  7e+00  1e-16  1e+00
 1:  9.2685e-02  9.2453e-02  2e+00  6e-01  2e-16  8e-02
 2:  5.5463e-01  5.5255e-01  7e-01  2e-01  4e-16  3e-02
 3:  5.1080e-01  5.0562e-01  3e-01  1e-01  2e-15  9e-03
 4:  5.5680e-01  5.5668e-01  7e-03  2e-03  4e-16  2e-04
 5:  5.5720e-01  5.5720e-01  7e-05  2e-05  4e-16  2e-06
 6:  5.5720e-01  5.5720e-01  7e-07  2e-07  4e-16  2e-08
```

In [5]:

```
S = S/50
plt.imshow(S, origin = 'lower', extent = [0, K, 0, N])
plt.xlabel('k-axis')
plt.ylabel('n-axis')
plt.show()
```



Conclusion

The brightness of the point indicates the probability of success, estimated from 50 independent trials. White represents certain success, while dark represents certain failure. The plot evinces that, for a given sparsity level, the minimization technique almost always succeeds when we have an adequate number of measurements, while it almost always fails when we have fewer measurements.

In [ ]:

```
```