Lab 24. Spectral Analysis

In [1]:
```
import networkx as nx
import matplotlib.pylab as plt
import numpy as np
```

In [2]:
```
# to draw a histogram of eigenvalues
def draw_spectrum(E, bins=50, title=None):
    R = [x.real for x in E]
    ys, xs = np.histogram(R, bins=bins)
    plt.plot(xs[:-1], ys, '--')
    if title is not None:
        plt.title(title)
    plt.show()

def save_spectrum(E, bins=50, title=None):
    plt.figure()
    R = [x.real for x in E]
    ys, xs = np.histogram(R, bins=bins)
    plt.plot(xs[:-1], ys, '--')
    if title is not None:
        plt.title(title)
    plt.savefig(title, dpi=300)
```

Adjacency Spectra

Path Graph

- $\lambda_k = 2\cos(\pi k/(n+1))$ for $k = 1, 2, \cdots, n$

In [3]:
```
# path graph with 8 nodes
Gpath = nx.cycle_graph(8)
Gpath.remove_edge(0,7)
E = nx.adjacency_spectrum(Gpath)
print(E)
```

```
[-1.87938524+0.j -1.53208889+0.j -1.        +0.j -0.34729636+0.j
  0.34729636+0.j  1.87938524+0.j  1.53208889+0.j  1.        +0.j]
```

Cycle Graph

- $\lambda_k = 2\cos(2\pi k/n)$ for $k = 0, 1, \cdots n-1$

In [4]:
```
# cycle graph with 8 nodes
Gcycle = nx.cycle_graph(8)
E = nx.adjacency_spectrum(Gcycle)
print(E)
```

```
[-2.00000000e+00+0.j -1.41421356e+00+0.j  2.00000000e+00+0.j
 -2.73024277e-16+0.j  1.41421356e+00+0.j  5.06493435e-18+0.j
 -1.41421356e+00+0.j  1.41421356e+00+0.j]
```

Cospectral (Isospectral)

In [5]:
```
# G1 on page 311
G1 = nx.cycle_graph(4)
G1.add_node(4)
nx.adjacency_spectrum(G1)
```

Out[5]:
```
array([-2.00000000e+00+0.j,  6.59737022e-17+0.j,  2.00000000e+00+0.j,
        0.00000000e+00+0.j,  0.00000000e+00+0.j])
```

In [6]:
```
# G2 on page 311
G2 = nx.star_graph(4)
nx.adjacency_spectrum(G2)
```

Out[6]:
```
array([ 2.+0.j, -2.+0.j,  0.+0.j,  0.+0.j,  0.+0.j])
```

Complete Graph

In [7]:
```
# complete graph with 6 nodes
C6 = nx.complete_graph(6)
nx.adjacency_spectrum(C6)
```

Out[7]:
```
array([-1.+0.j,  5.+0.j, -1.+0.j, -1.+0.j, -1.+0.j, -1.+0.j])
```

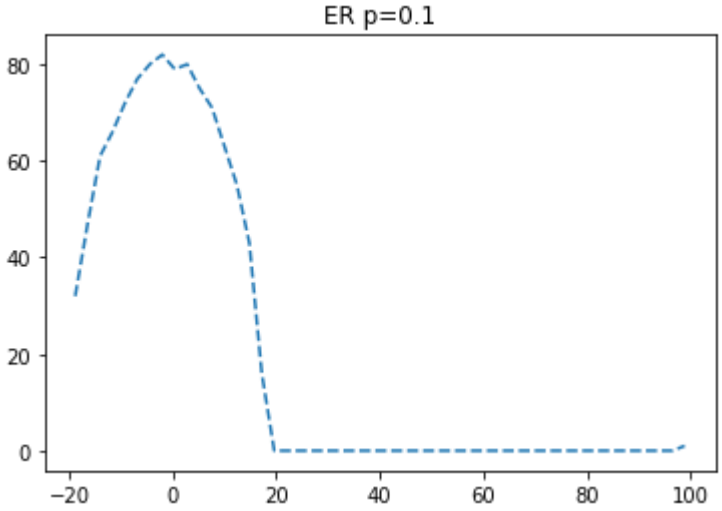
Bipartite Graph

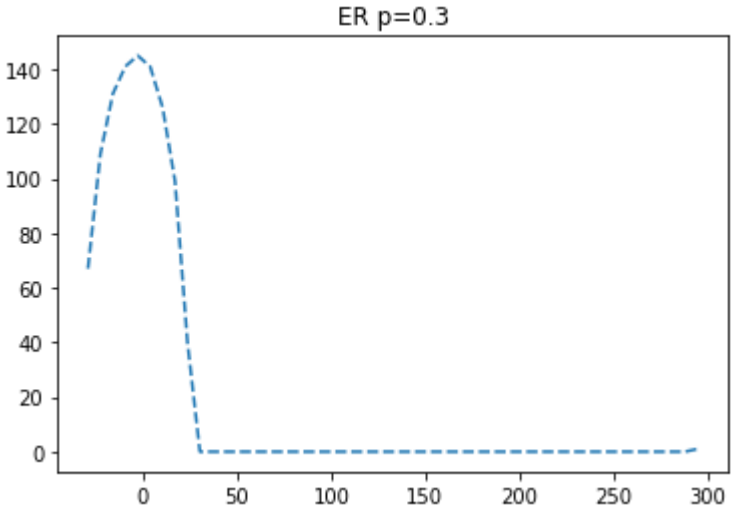
In [8]:
```
Gbi = nx.bipartite.random_graph(5, 7, 0.2)
nx.adjacency_spectrum(Gbi)
```

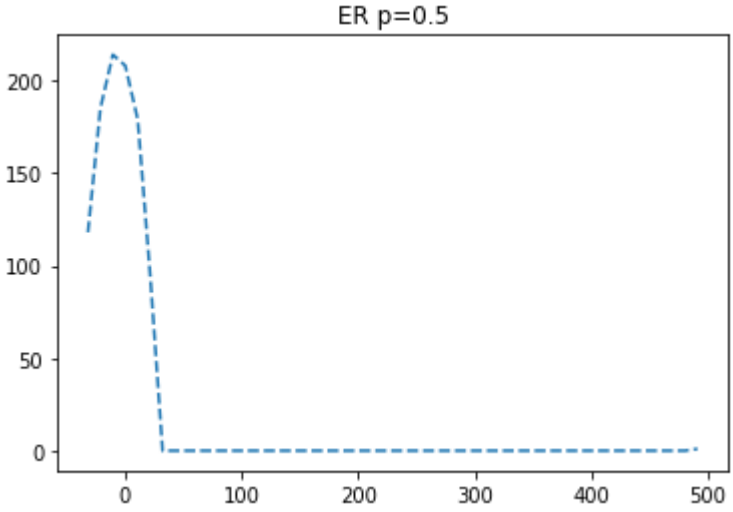
Out[8]:
```
array([ 2.58145356e+00+0.j, -2.58145356e+00+0.j,  1.67337956e+00+0.j,
        1.33625869e+00+0.j,  8.66205000e-01+0.j, -1.67337956e+00+0.j,
       -1.33625869e+00+0.j, -8.66205000e-01+0.j,  1.40793535e-16+0.j,
       -6.05643949e-17+0.j,  3.79768273e-17+0.j,  0.00000000e+00+0.j])
```

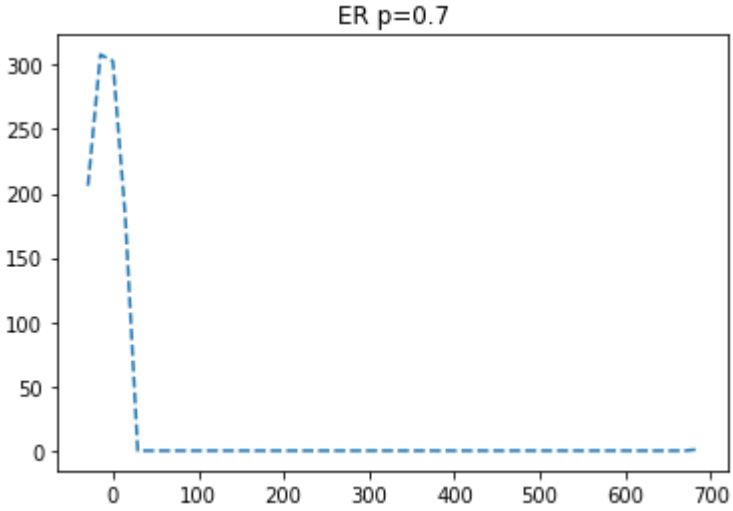
Random Graph (ER random model)

In [9]:
```
for prob in [.1, .3, .5, .7]:
    G = nx.erdos_renyi_graph(1000, prob)
    draw_spectrum(nx.adjacency_spectrum(G), title="ER p="+str(prob))
```


ER p=0.1
80
60
40
20
0
−20
0
20
40
60
80
100

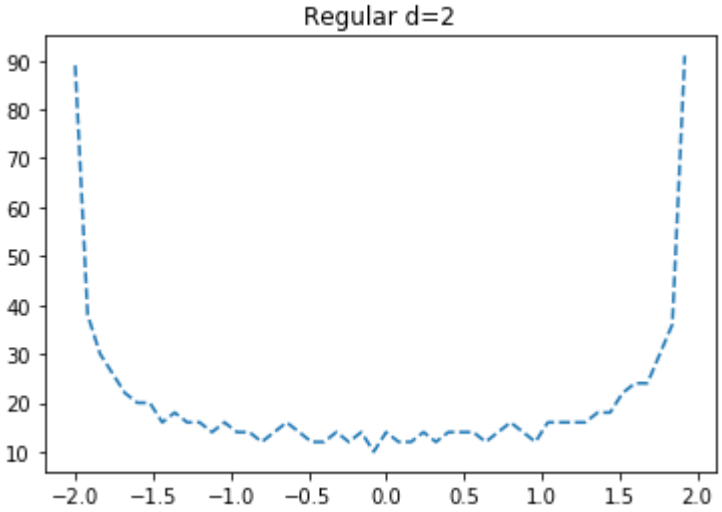

ER p=0.3
140
120
100
80
60
40
20
0
0
50
100
150
200
250
300

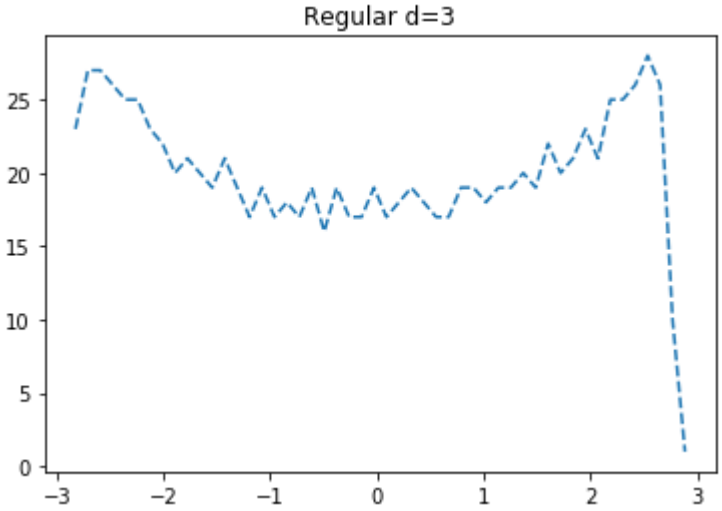

ER p=0.5
200
150
100
50
0
0
100
200
300
400
500

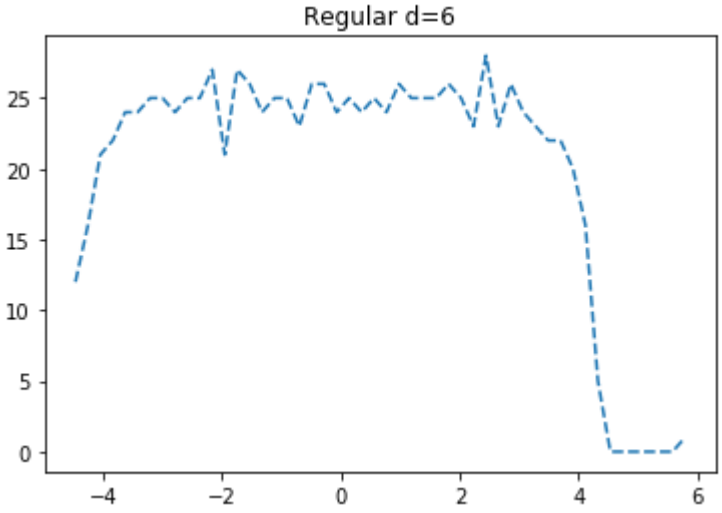


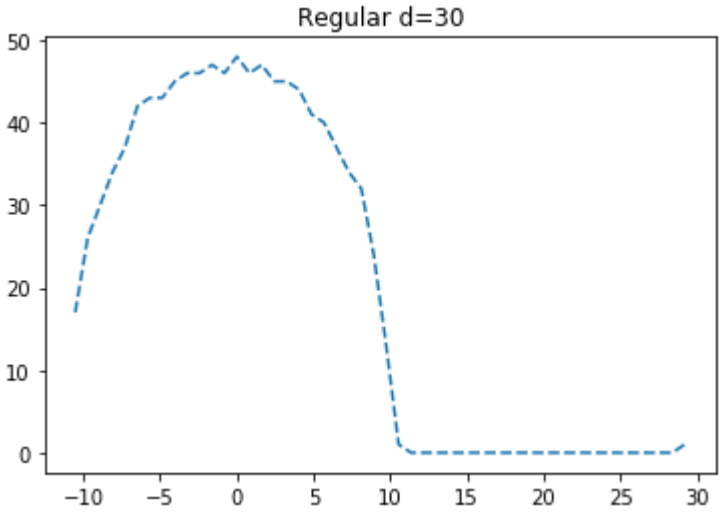
d-Regular Graph

In [10]:
```
for d in (2,3,6,30):
    G = nx.random_regular_graph(d, 1000)
    draw_spectrum(nx.adjacency_spectrum(G), title="Regular d="+str(d))
```


Regular d=2
90
80
70
60
50
40
30
20
10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Regular d=3
25
20
15
10
5
0
−3
−2
−1
0
1
2
3

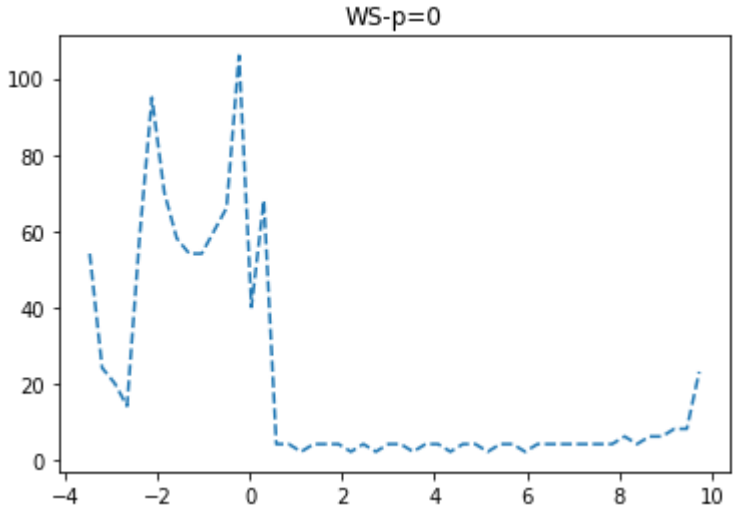

Regular d=6
25
20
15
10
5
0
−4
−2
0
2
4
6



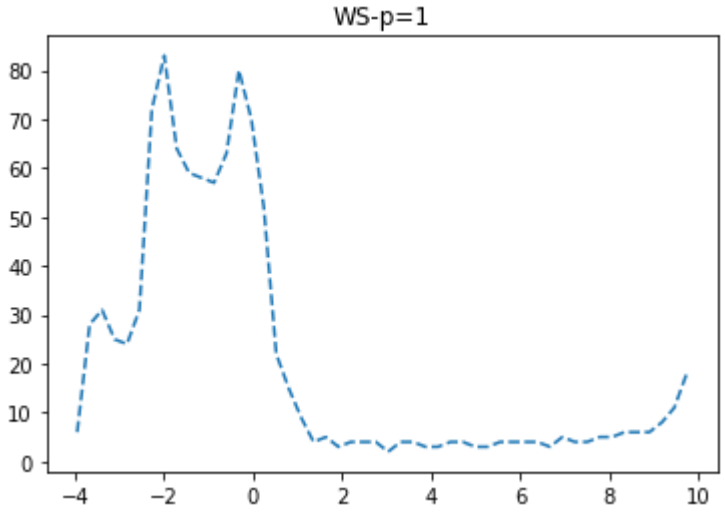
Small-World Graph (WS model)

In [11]:
```
for prob in (.0, .01, .4, .6, .8):
    print(str(int(prob*100)))
    G = nx.watts_strogatz_graph(1000, 10, p=prob)
    draw_spectrum(nx.adjacency_spectrum(G), title="WS-p="+str(int(prob*100)))
```

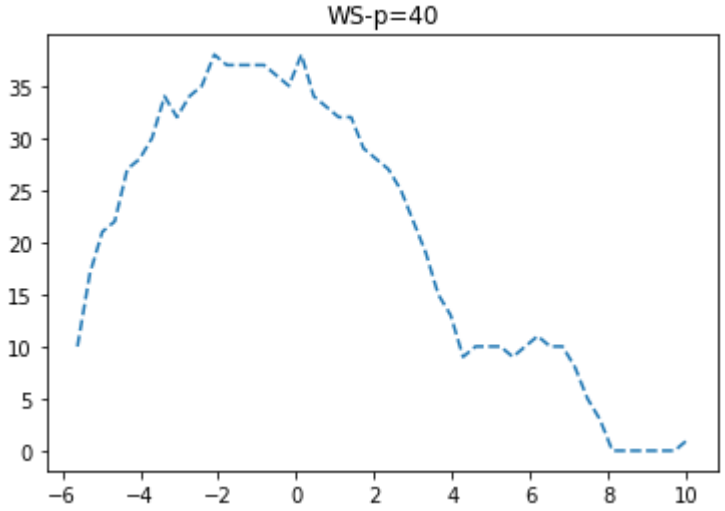
```
0
```


WS-p=0
100
80
60
40
20
0
−4
−2
0
2
4
6
8
10

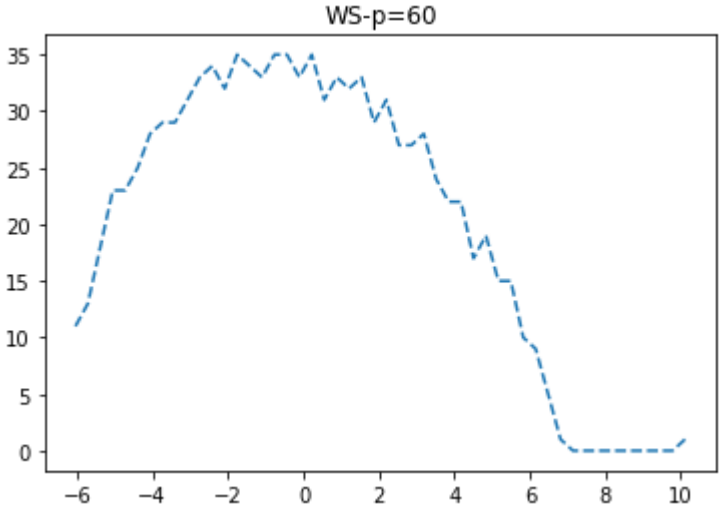
```
1
```


WS-p=1
80
70
60
50
40
30
20
10
0
−4
−2
0
2
4
6
8
10

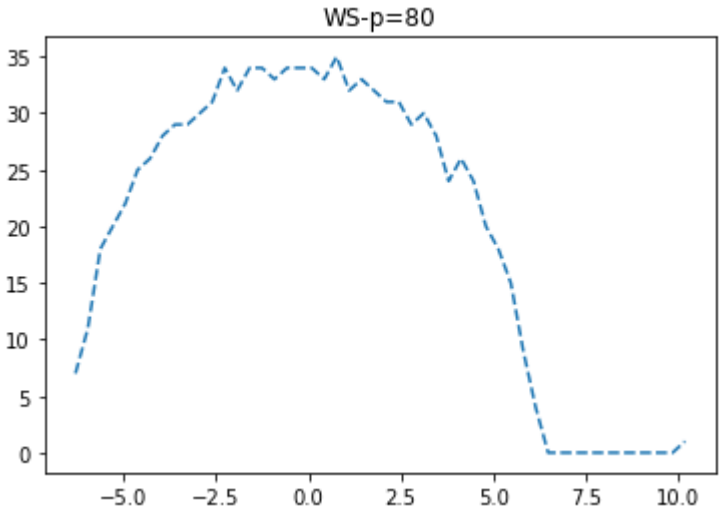
```
40
```


WS-p=40
35
30
25
20
15
10
5
0
−6
−4
−2
0
2
4
6
8
10

```
60
```

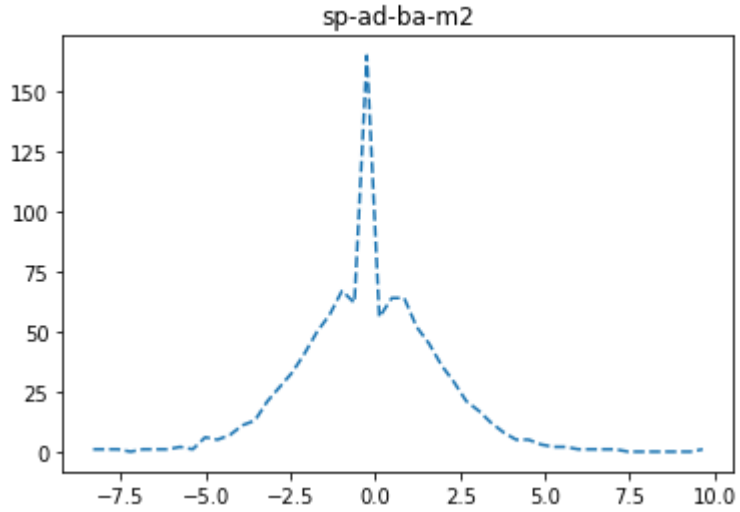


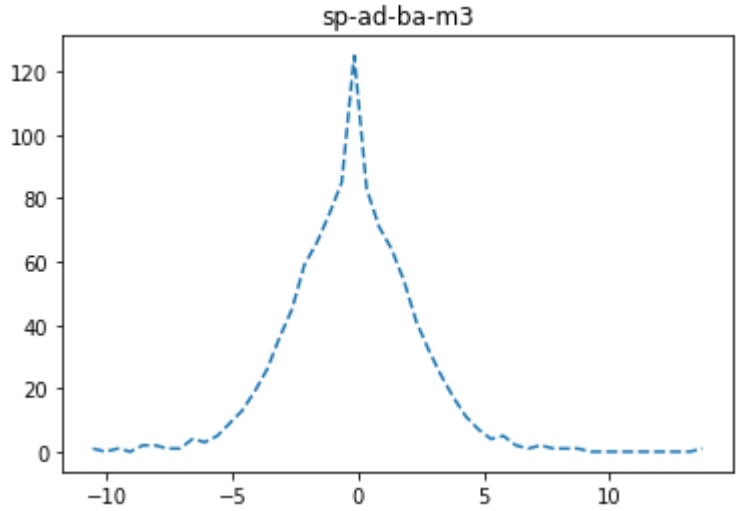

```
80
```

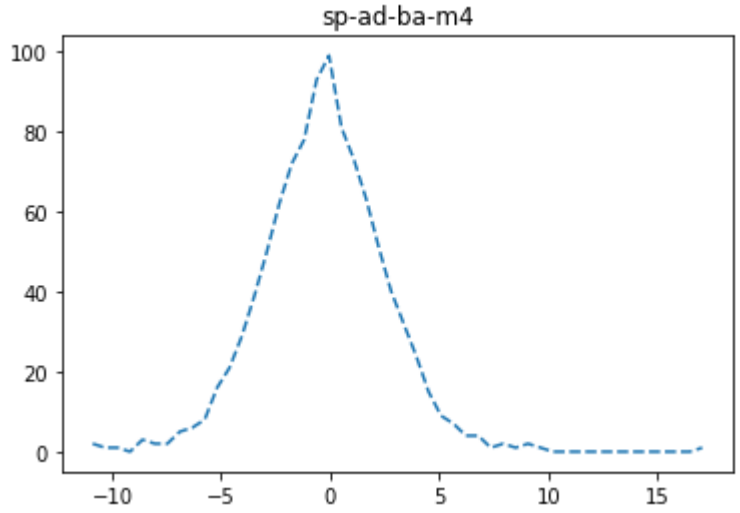


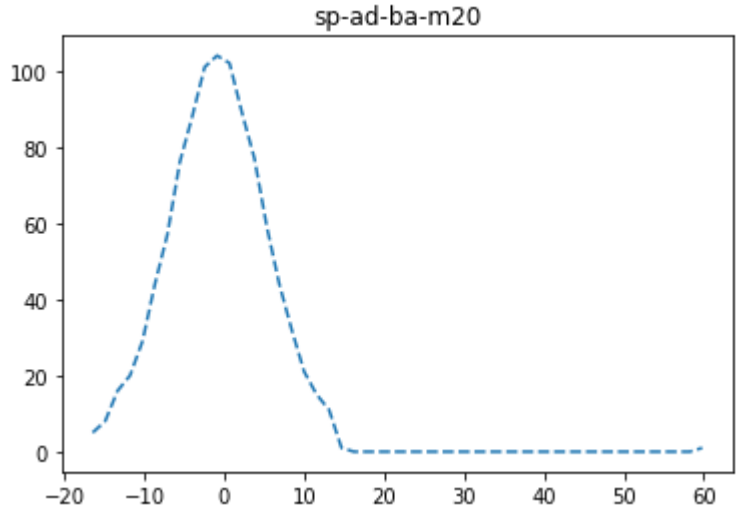
Scale-Free Graph (BA model)

In [12]:
```
for m in (2, 3, 4, 20):
    G = nx.barabasi_albert_graph(1000, m)
    draw_spectrum(nx.adjacency_spectrum(G), title="sp-ad-ba-m"+str(m))
```


sp-ad-ba-m2
150
125
100
75
50
25
0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


sp-ad-ba-m3
120
100
80
60
40
20
0
−10
−5
0
5
10


sp-ad-ba-m4
100
80
60
40
20
0
−10
−5
0
5
10
15



Laplacian Spectrum

Spectral Graph Clustering

In [13]:
```
# build a grape on page 328
G = nx.Graph()
G.add_edges_from([
    (1,2), (2,3), (3,1), (1,4), (2,4),
    (5,6), (5,7),
    (8,9), (9,10), (8,10),
    (11,12), (12,13), (13,14), (14,11), (11,14),
])
L = np.array(nx.laplacian_matrix(G).toarray())
print(L)
```

```
[[ 3 -1 -1 -1  0  0  0  0  0  0  0  0  0  0]
 [-1  3 -1 -1  0  0  0  0  0  0  0  0  0  0]
 [-1 -1  2  0  0  0  0  0  0  0  0  0  0  0]
 [-1 -1  0  2  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  2 -1 -1  0  0  0  0  0  0  0]
 [ 0  0  0  0 -1  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 -1  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2 -1 -1  0  0  0  0]
 [ 0  0  0  0  0  0  0 -1  2 -1  0  0  0  0]
 [ 0  0  0  0  0  0  0 -1 -1  2  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  2 -1  0 -1]
 [ 0  0  0  0  0  0  0  0  0  0 -1  2 -1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 -1  2 -1]
 [ 0  0  0  0  0  0  0  0  0  0 -1  0 -1  2]]
```

In [14]:
```
val, vec = np.linalg.eig(L)
print("eigenvalue: ", val)       # it is not ordered
print("eigenvector: ", vec)      # the corresponding eigenvector is a column v
ector
```

```
eigenvalue:  [ 4.00000000e+00 -1.11022302e-16  4.00000000e+00  2.00000000e+00
  3.00000000e+00 -2.91433544e-16  1.00000000e+00  3.00000000e+00
 -4.44089210e-16  3.00000000e+00 -2.22044605e-16  2.00000000e+00
  4.00000000e+00  2.00000000e+00]
eigenvector:  [[ 8.66025404e-01 -5.00000000e-01 -9.46547200e-02  1.35973996e-
16
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-2.88675135e-01 -5.00000000e-01  8.43156559e-01  2.26623326e-16
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-2.88675135e-01 -5.00000000e-01 -3.74250919e-01 -7.07106781e-01
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-2.88675135e-01 -5.00000000e-01 -3.74250919e-01  7.07106781e-01
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  -8.16496581e-01  5.77350269e-01  1.57009246e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   4.08248290e-01  5.77350269e-01 -7.07106781e-01  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   4.08248290e-01  5.77350269e-01  7.07106781e-01  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  8.16496581e-01
  -5.77350269e-01  2.93294230e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00 -4.08248290e-01
  -5.77350269e-01 -8.06559133e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00 -4.08248290e-01
  -5.77350269e-01  5.13264903e-01  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  5.00000000e-01  7.07106781e-01
  -5.00000000e-01  2.77555756e-17]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  5.00000000e-01 -2.02962647e-16
   5.00000000e-01 -7.07106781e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  5.00000000e-01 -7.07106781e-01
  -5.00000000e-01  2.86262916e-16]
```

```
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  5.00000000e-01  1.02348685e-16
  5.00000000e-01  7.07106781e-01]]
```

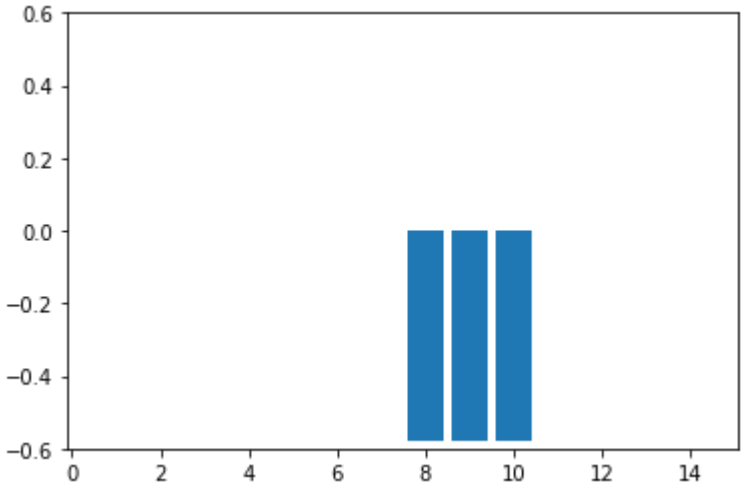
In [15]:
```
idx = val.argsort()        # order w and return its indices
idx
```

Out[15]:
```
array([ 8,  5, 10,  1,  6, 13,  3, 11,  7,  9,  4,  2, 12,  0])
```

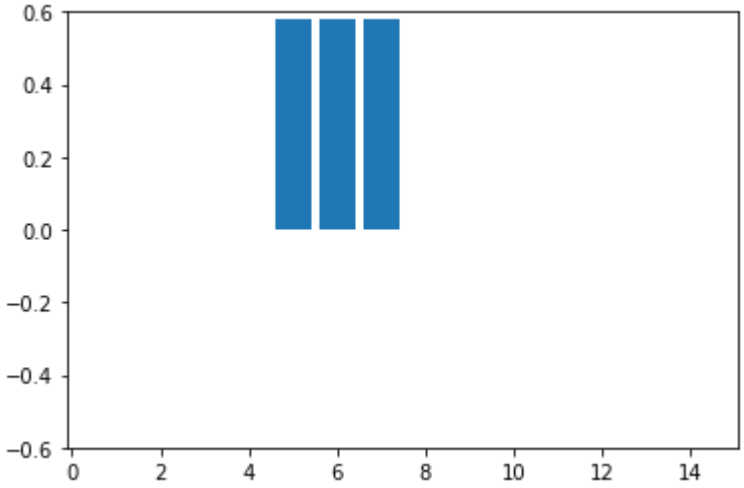
In [16]:
```
w = val[idx]
v = vec[:,idx]
```

In [17]:
```
num_clusters = 4
nodes = list(G.nodes())
for i in range(num_clusters):
    print("lambda_%d = "%i, w[i])
    print(v[:,i])
    plt.bar(nodes, v[:,i])
    plt.ylim((-0.6,0.6))
    plt.show()
```

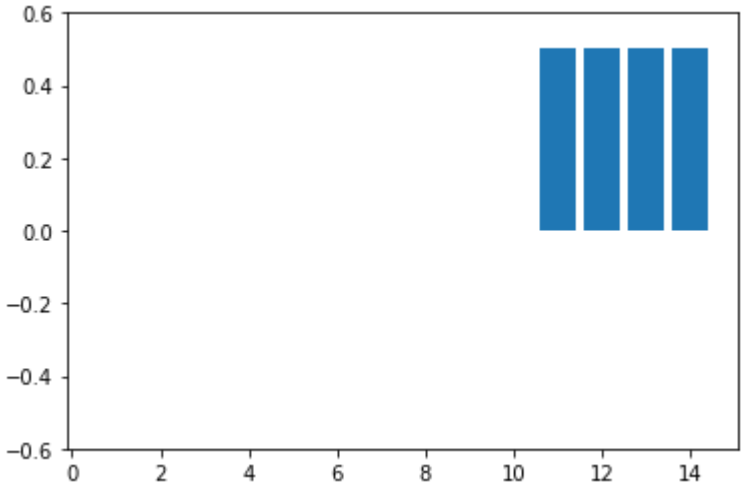
```
lambda_0 =  -4.440892098500626e-16
[ 0.          0.          0.          0.          0.          0.
  0.         -0.57735027 -0.57735027 -0.57735027  0.          0.
  0.          0.        ]
```



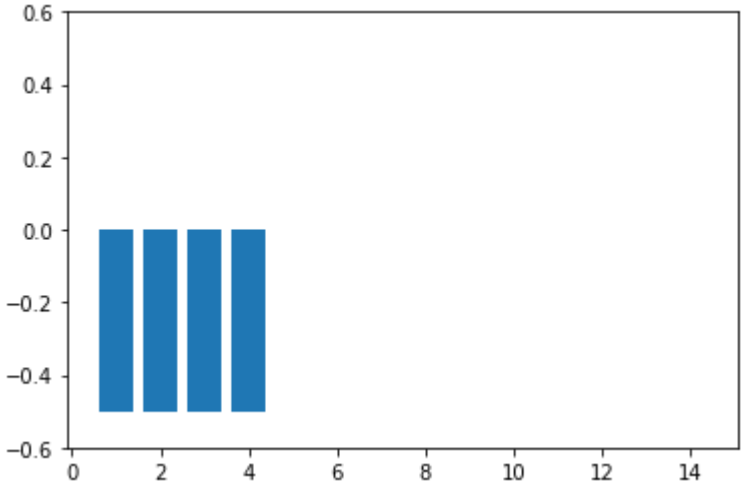

```
lambda_1 =  -2.914335439641036e-16
[0.         0.         0.         0.         0.57735027 0.57735027
 0.57735027 0.         0.         0.         0.         0.
 0.         0.        ]
```




```
lambda_2 =  -2.220446049250313e-16
[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.5 0.5 0.5 0.5]
```




```
lambda_3 =  -1.1102230246251565e-16
[-0.5 -0.5 -0.5 -0.5  0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
```

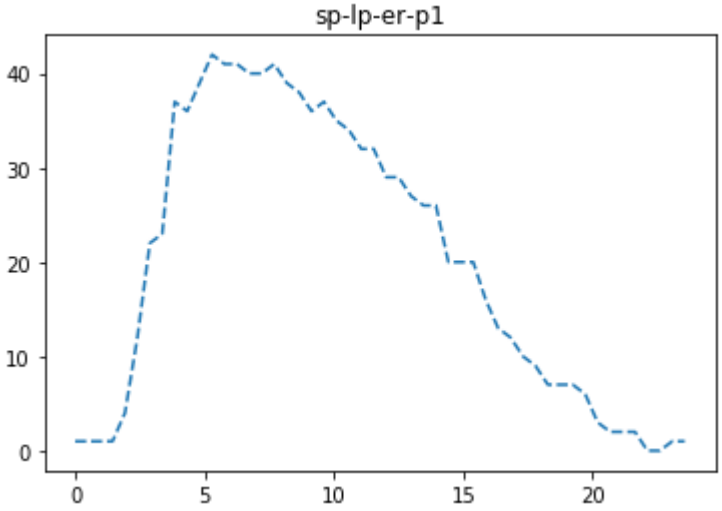


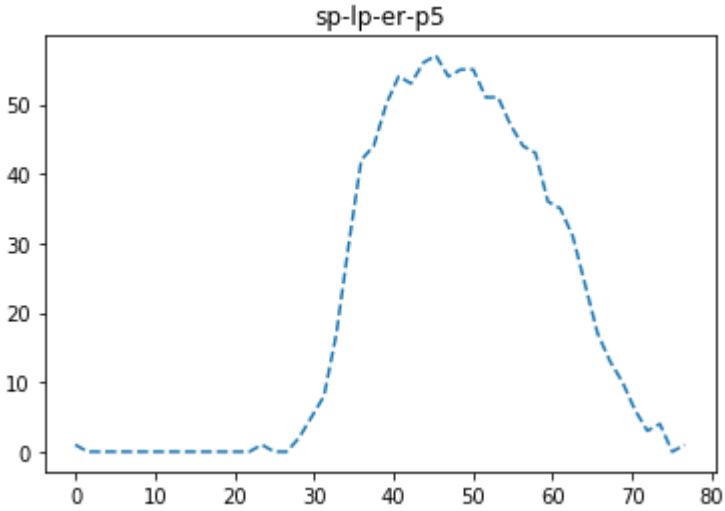
Random Graph (ER random model)

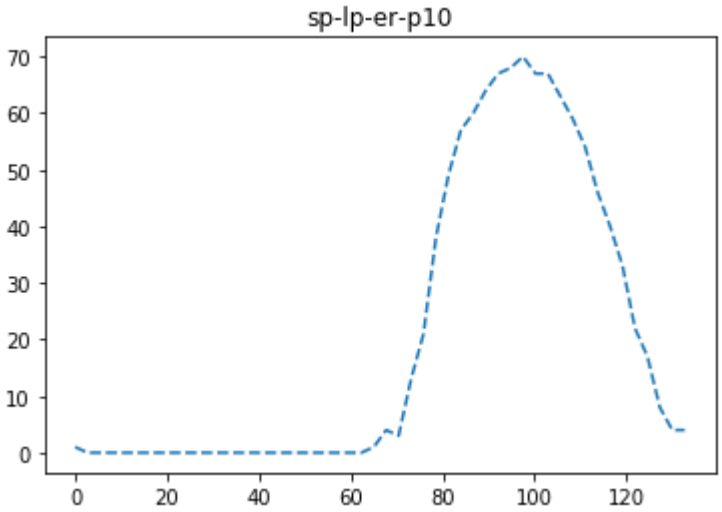
In [18]:
```
w.argsort()[::1]
```

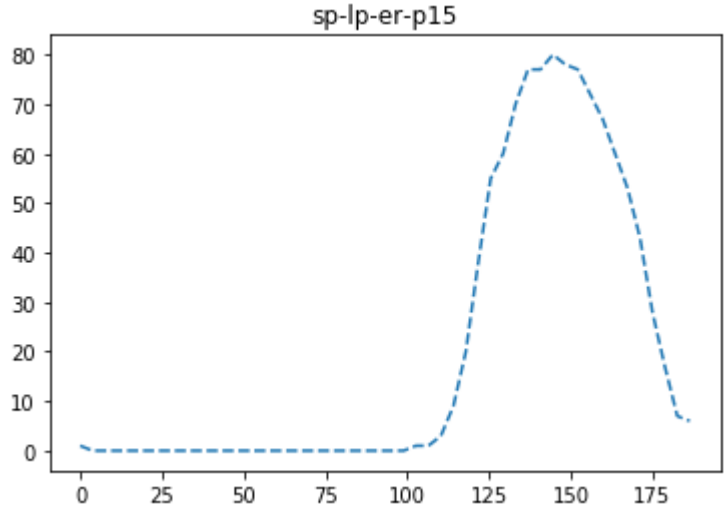
Out[18]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])
```

In [19]:
```
for prob in [.01, .05, .1, .15]:
    G = nx.erdos_renyi_graph(1000, prob)
    draw_spectrum(nx.laplacian_spectrum(G), title="sp-lp-er-p"+str(int(prob*10
0)))
```


sp-lp-er-p1
0
10
20
30
40
0
5
10
15
20

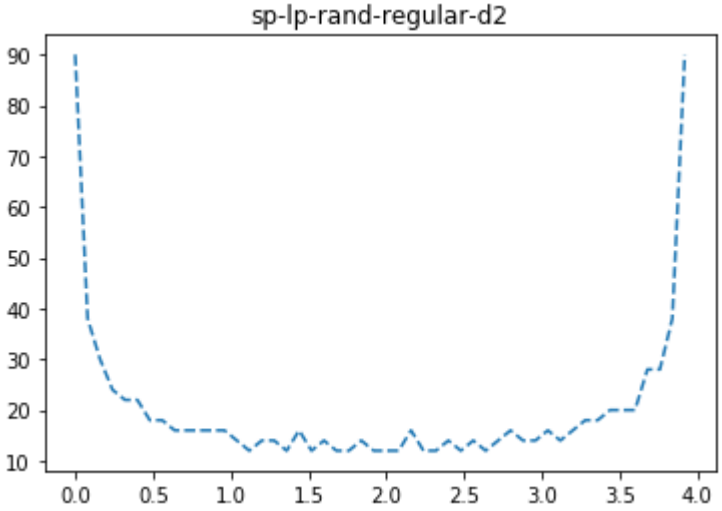

sp-lp-er-p5
0
10
20
30
40
50
0
10
20
30
40
50
60
70
80

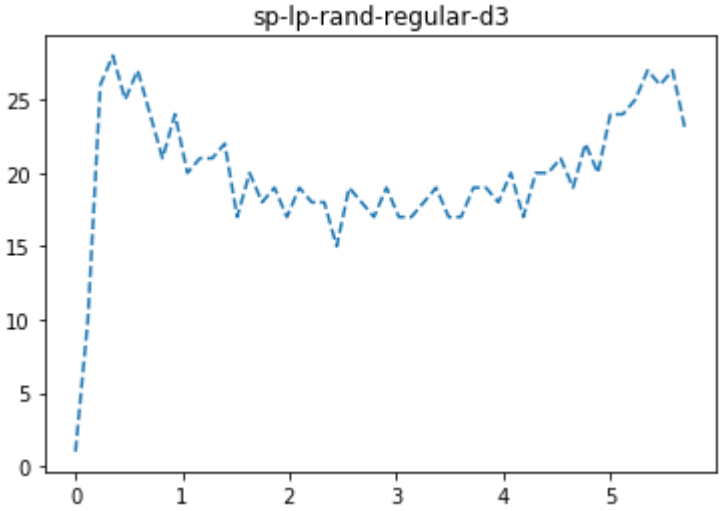

sp-lp-er-p10
0
10
20
30
40
50
60
70
0
20
40
60
80
100
120

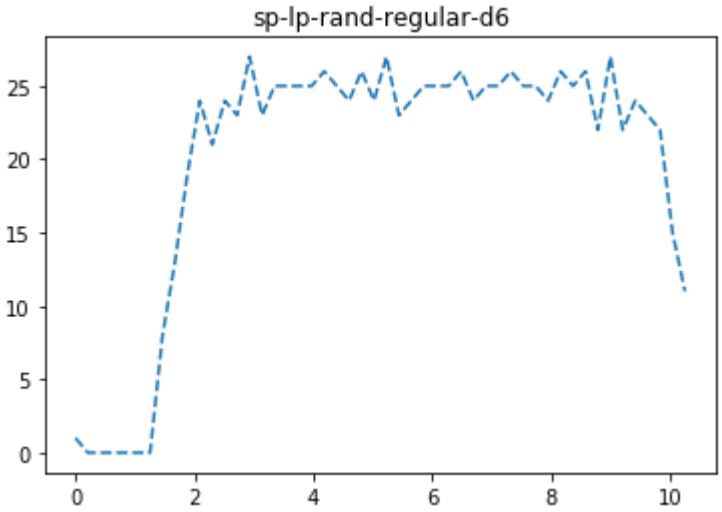


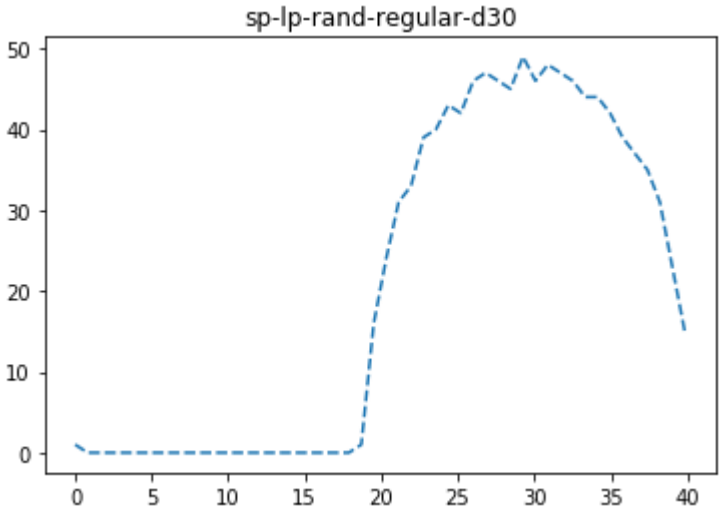
d-Regular Graph

In [20]:
```
for d in (2,3,6,30):
    G = nx.random_regular_graph(d, 1000)
    save_spectrum(nx.laplacian_spectrum(G), title="sp-lp-rand-regular-d"+str(d
))
```


sp-lp-rand-regular-d2
90
80
70
60
50
40
30
20
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

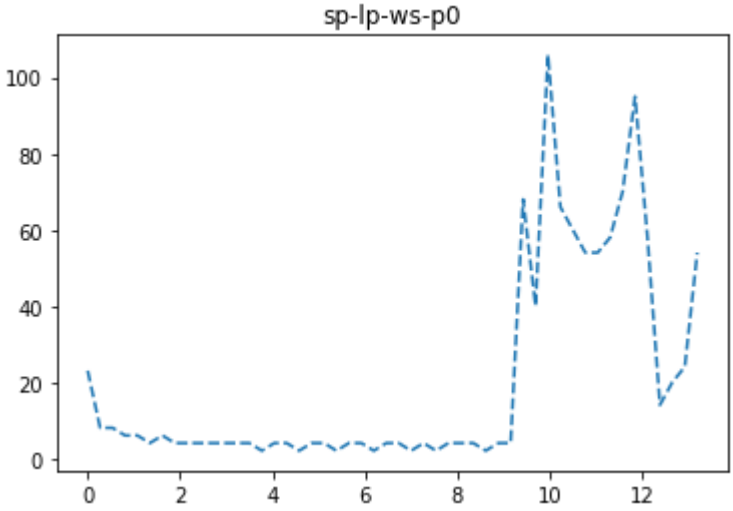

sp-lp-rand-regular-d3
25
20
15
10
5
0
0
1
2
3
4
5

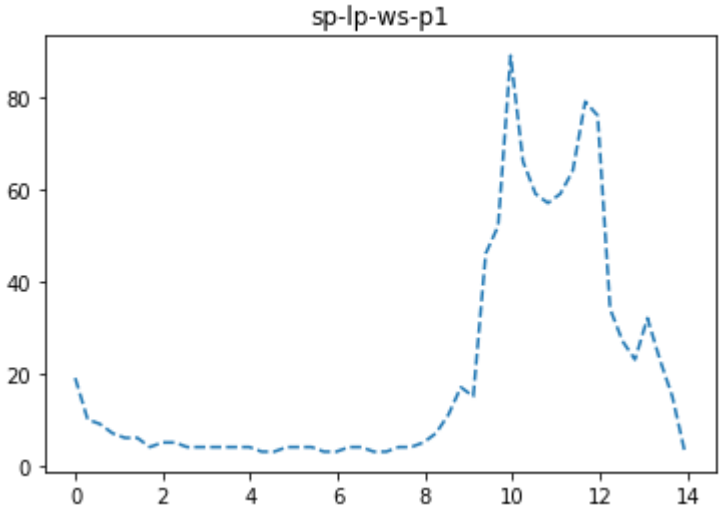

sp-lp-rand-regular-d6
25
20
15
10
5
0
0
2
4
6
8
10

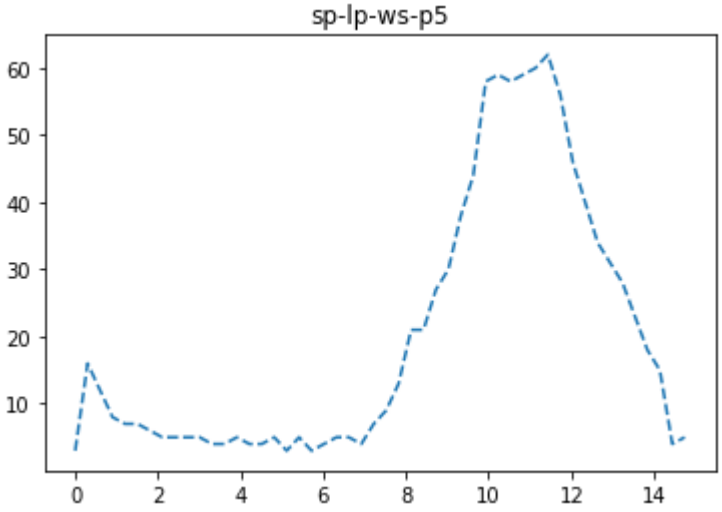


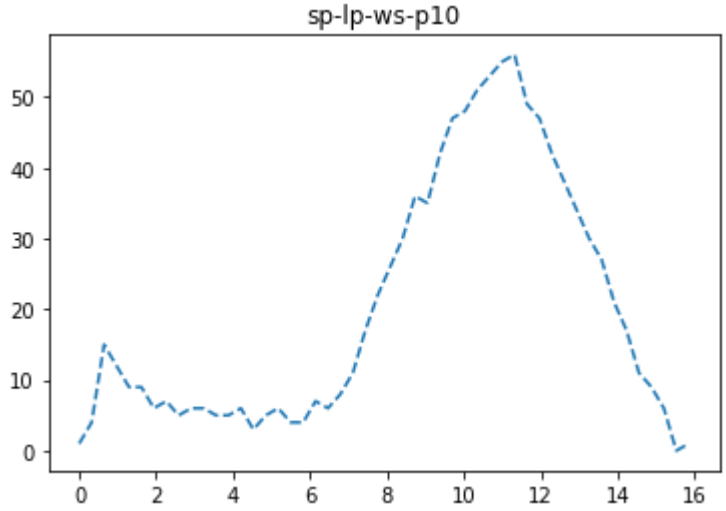
Small-World Graph (WS model)

In [21]:
```
for prob in (.0, .01, .05, .1):
    G = nx.watts_strogatz_graph(1000, 10, p=prob)
    save_spectrum(nx.laplacian_spectrum(G), title="sp-lp-ws-p"+str(int(prob*10
0)))
```


sp-lp-ws-p0
100
80
60
40
20
0
0
2
4
6
8
10
12

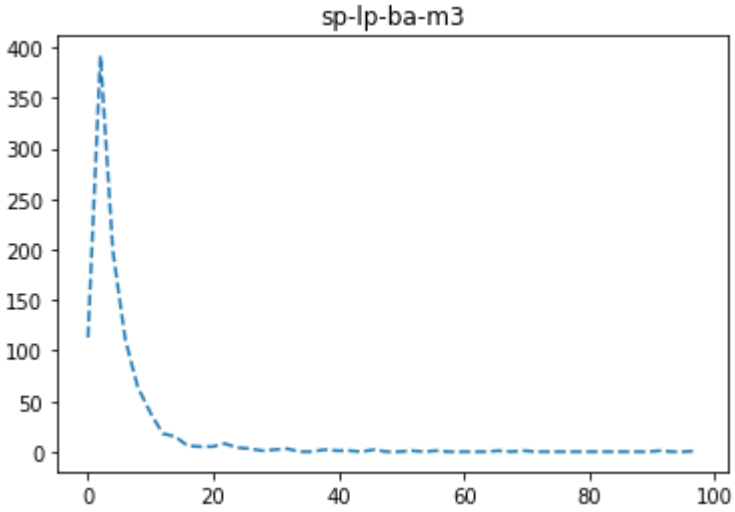

sp-lp-ws-p1
80
60
40
20
0
0
2
4
6
8
10
12
14

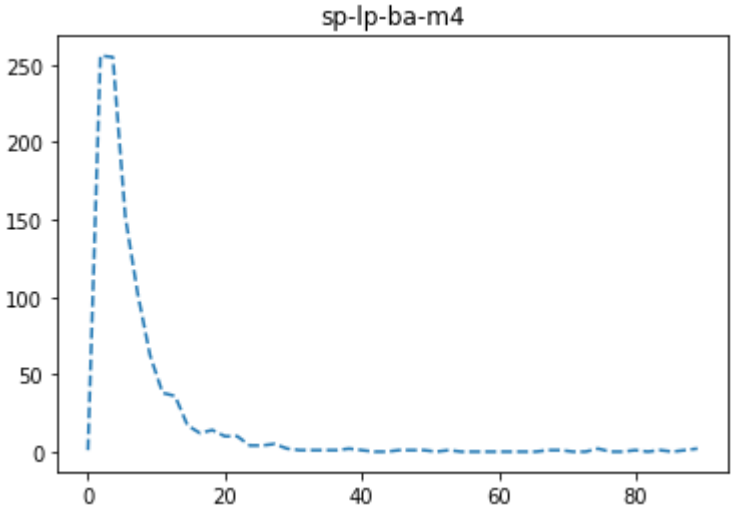

sp-lp-ws-p5
60
50
40
30
20
10
0
2
4
6
8
10
12
14

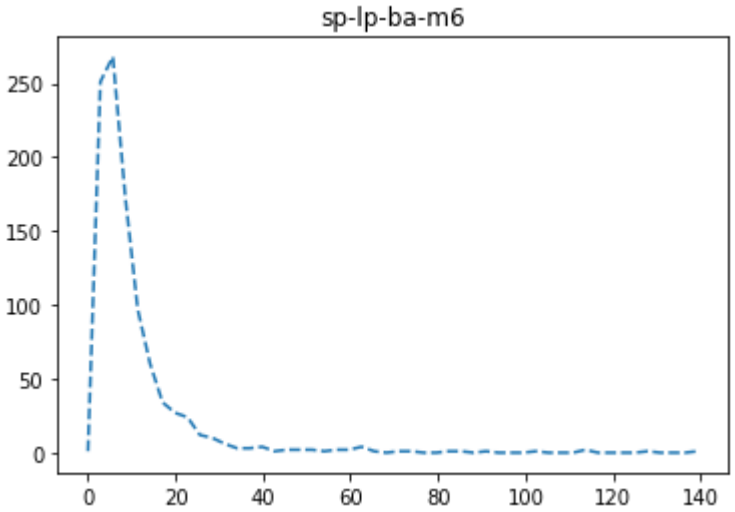


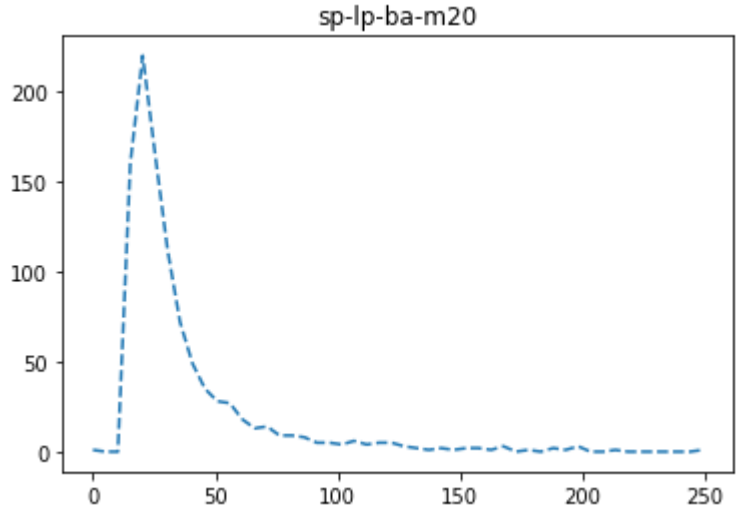
Scale-Free Graph (BA model)

In [22]:
```
for m in (3, 4, 6, 20):
    G = nx.barabasi_albert_graph(1000, m)
    draw_spectrum(nx.laplacian_spectrum(G), title="sp-lp-ba-m"+str(m))
```


sp-lp-ba-m3
400
350
300
250
200
150
100
50
0
0
20
40
60
80
100


sp-lp-ba-m4
250
200
150
100
50
0
0
20
40
60
80


sp-lp-ba-m6
250
200
150
100
50
0
0
20
40
60
80
100
120
140



In [ ]: