2023-06-19_021_Graph-Embeddings-SpectralLaplacian

June 27, 2023

[5]:
```
import numpy as np
import scipy
import imageio

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm

matplotlib.rc('image', interpolation='nearest')
matplotlib.rc('figure',facecolor='white')
matplotlib.rc('image',cmap='viridis')
colors=plt.rcParams['axes.prop_cycle'].by_key()['color']
%matplotlib inline

from matplotlib.animation import FuncAnimation
matplotlib.rc('animation',html='html5')
import colorcet as ccm
from graphplot import *
```

[6]:
```
def getL(A,degnorm=False):
    """Basic implementation of graph Laplacian for undirected graph where␣
↪A_ij=1 if
    there is an edge between edges (i,j), so A is symmetric."""
    # every edge corresponds to an entry -1
    # (will later see: can also weigh edges, as long as weights are symmetric)
    L=-A
    # degree of each vertex
    deg=np.sum(A,axis=1)
    # write the degree onto the diagonal
    np.fill_diagonal(L,deg)
    if degnorm:
        # sometimes, various "normalizations" of the Laplacian work better
        L=np.einsum(L,[0,1],deg**(-0.5),[0],deg**(-0.5),[1],[0,1])


    if degnorm:
        vecscale=deg**(-0.5)
```

```python
    else:
        vecscale=np.ones(deg.shape)

    return L,vecscale

def getEigdat(L):
    # diagonalize Laplacian
    eigdat=np.linalg.eigh(L)
    eigval=eigdat[0]
    eigvec=eigdat[1].transpose().copy()
    return eigval,eigvec
```

1 Spectral embedding with graph Laplacian

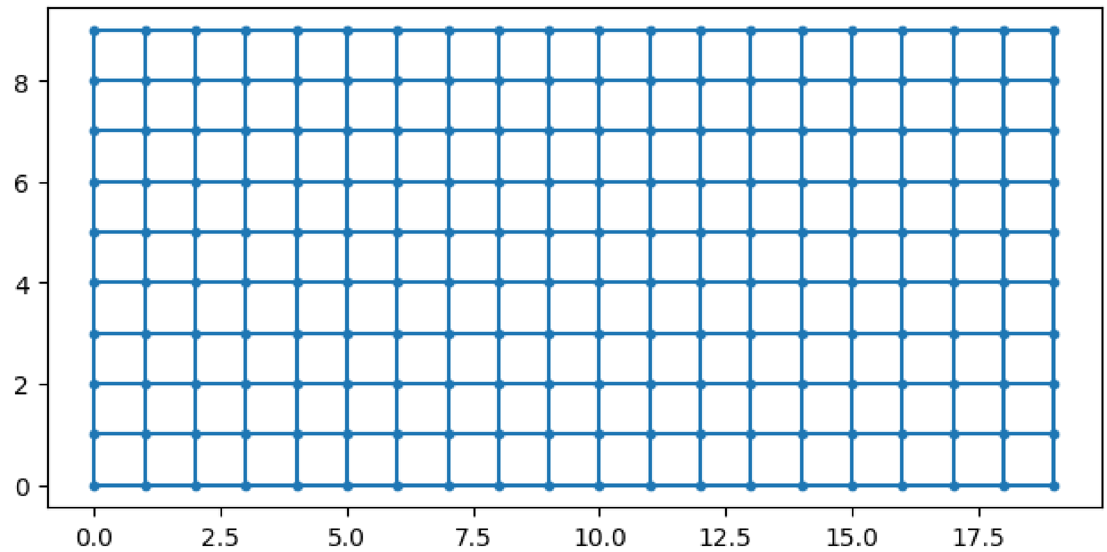
1.0.1 grid graph example

[7]:

```python
# build "rectangular" graphs
# how many rows and cols?
nRows=10
nCols=20
# spatial dimension
dim=2

posList,edgeData=buildGridGraph2d(nRows,nCols,neighbourhood=4)
nPoints=posList.shape[0]
nEdges=edgeData.shape[0]
edgeLengths=np.linalg.norm(posList[edgeData[:,0]]-posList[edgeData[:,1]],axis=1)

# build symmetric adjacency matrix
A=np.zeros((nPoints,nPoints),dtype=np.double)
A[edgeData[:,0],edgeData[:,1]]=1.
A[edgeData[:,1],edgeData[:,0]]=1.
deg=np.sum(A,axis=1)
```

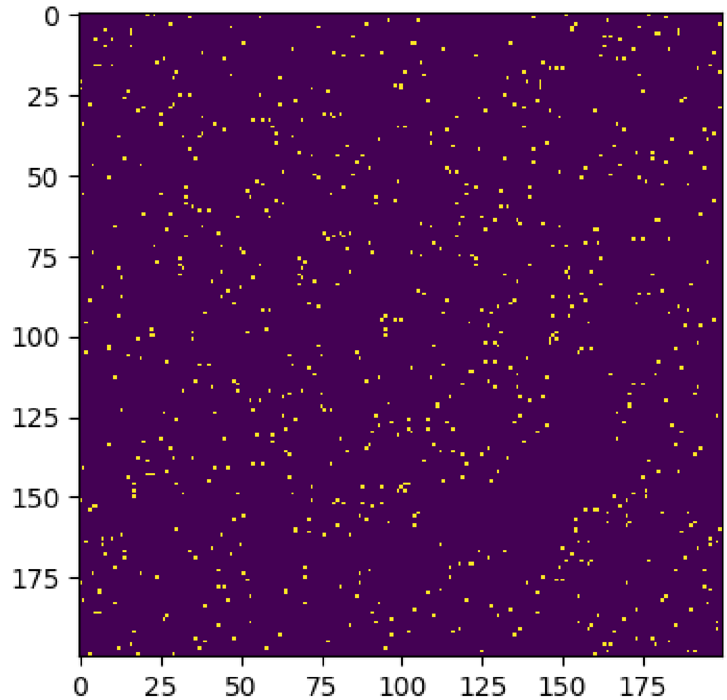
[8]:

```python
fig=plt.figure()
ax=fig.add_subplot(aspect=1.)
vertices(ax,posList,s=10)
edges(ax,posList,edgeData)
plt.tight_layout()
plt.show()
```



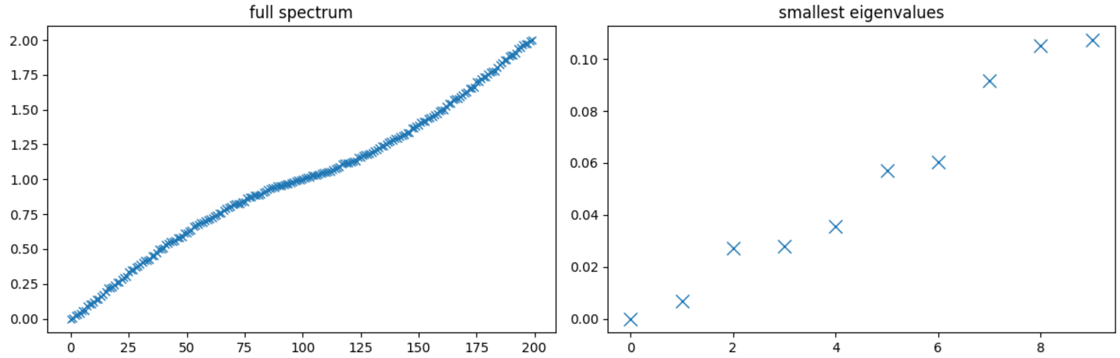
[9]:
```
perm=np.random.permutation(nPoints)
APerm=A[:,perm]
APerm=APerm[perm,:]
```

[12]:
```
%matplotlib inline
fig=plt.figure(figsize=(4,4),facecolor="w")
plt.imshow(APerm)
plt.tight_layout()
plt.show()
#fig.savefig("graph_matrix.png")
```

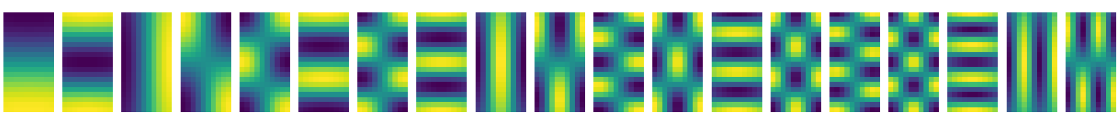


[13]:
```
# construct graph Laplacian (optional: degree-normalization)
degnorm=True
L,vecscale=getL(A,degnorm=degnorm)
eigval,eigvec=getEigdat(L)
```

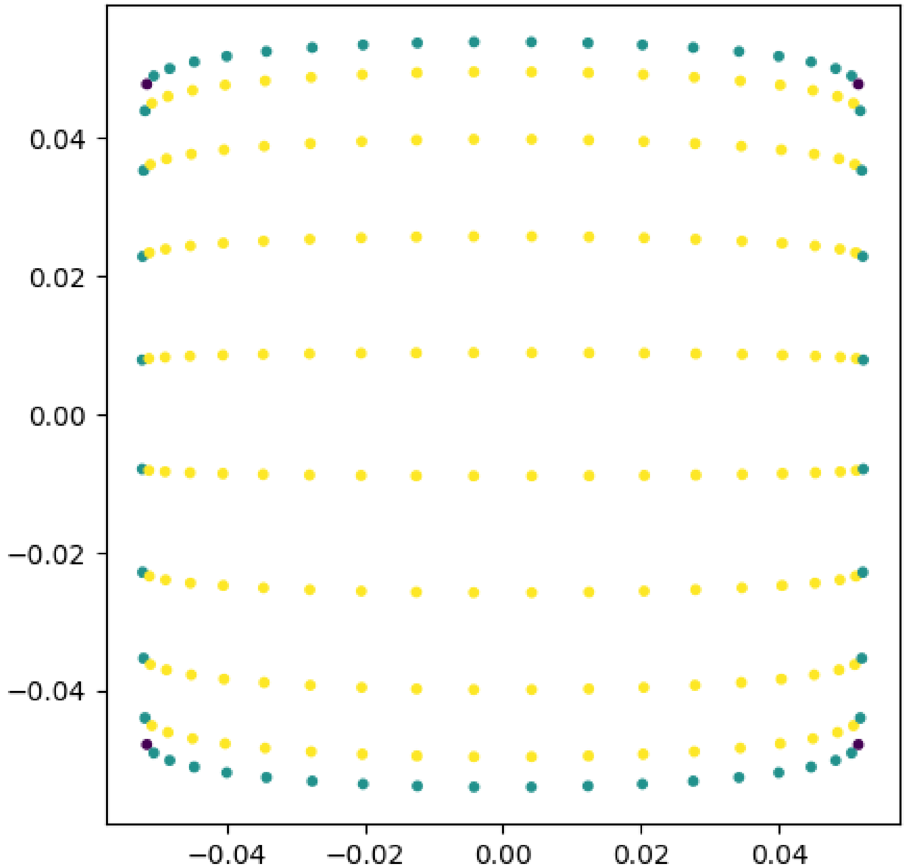
[14]:
```
# spectrum of Laplacian
fig=plt.figure(figsize=(12,4))
fig.add_subplot(1,2,1)
plt.title("full spectrum")
plt.plot(eigval,lw=0,marker="x")

fig.add_subplot(1,2,2)
plt.title("smallest eigenvalues")
plt.plot(eigval[:10],lw=0,marker="x",markersize=10)

plt.tight_layout()
plt.show()
```



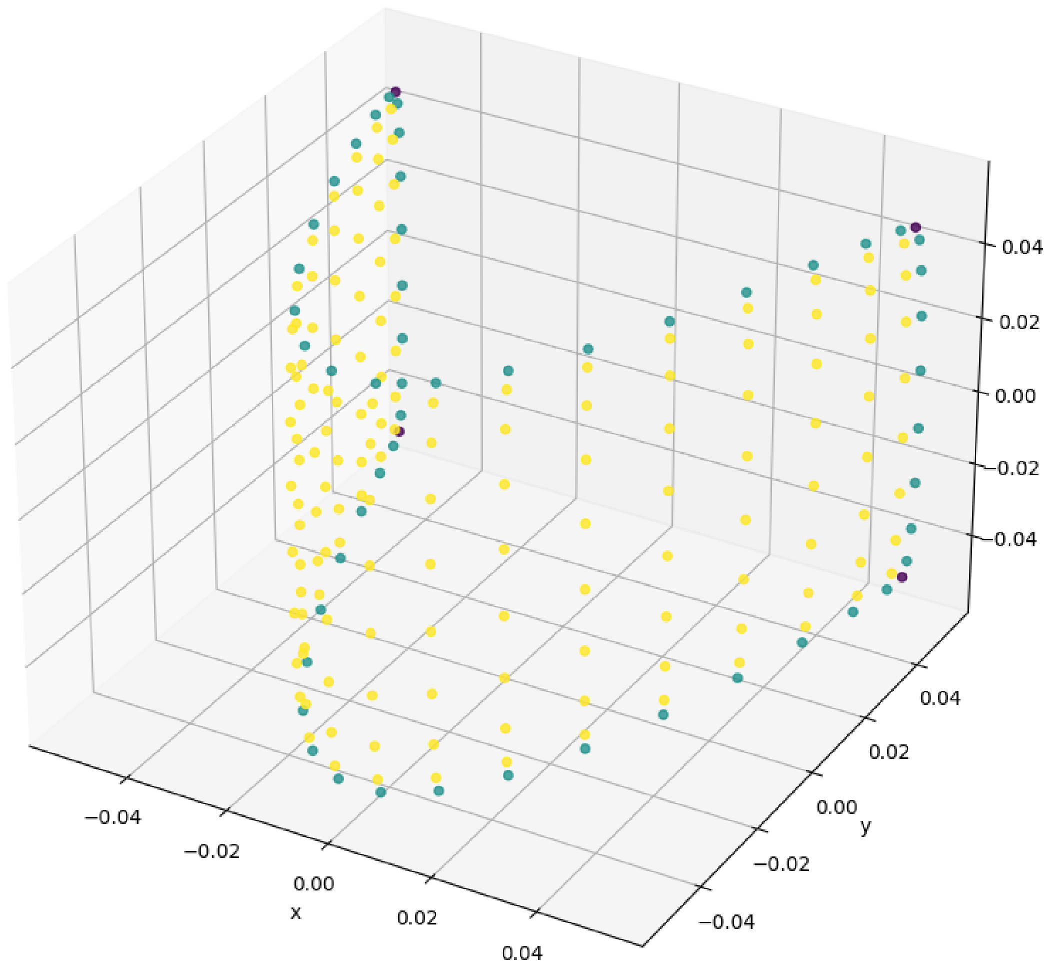
[15]:
```
# for grid graph: visualize eigenfunctions as images
# should see "vibration modes"
%matplotlib inline
sel=range(1,20)
nSel=len(sel)
fig=plt.figure(figsize=(1*nSel,3))
for i in range(nSel):
    fig.add_subplot(1,nSel,i+1)
    vec=eigvec[sel[i]]*vecscale
    plt.imshow(vec.reshape((nCols,nRows)))
    plt.axis("off")
plt.tight_layout()
plt.show()
```



[18]:
```
# 2d embedding
%matplotlib inline
i1=1
i2=3
fig=plt.figure()
fig.add_subplot(aspect=1.)
plt.scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,s=10,marker="o",c=deg)
plt.tight_layout()
plt.show()
```



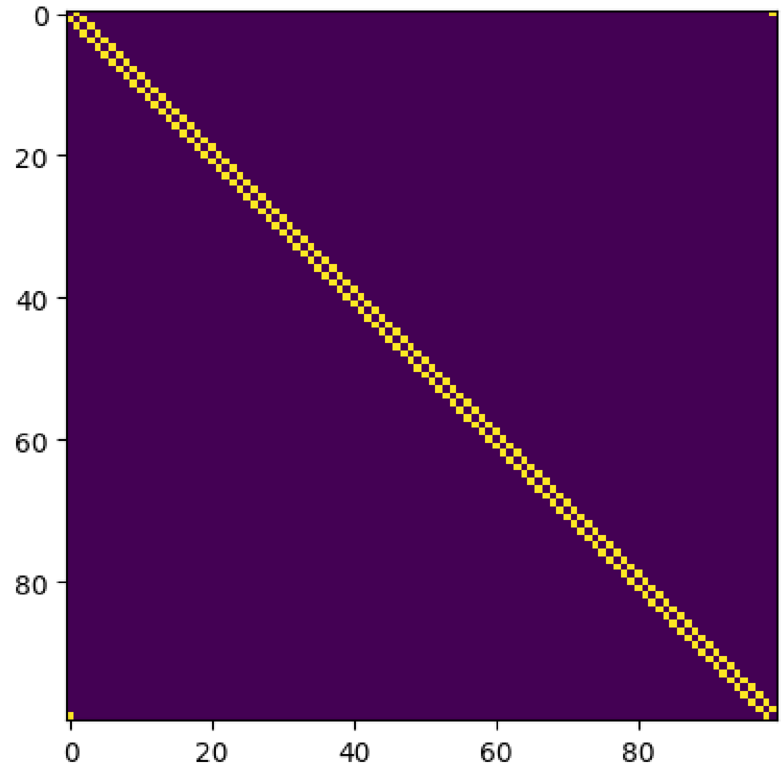

```
[19]: i1,i2,i3=1,2,3
%matplotlib inline
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')

ax.
 ↪scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,eigvec[i3]*vecscale,c=deg,alpha=0.
 ↪8)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
plt.tight_layout()
plt.show()
```



[20]:
```
plt.close()
%matplotlib inline
```

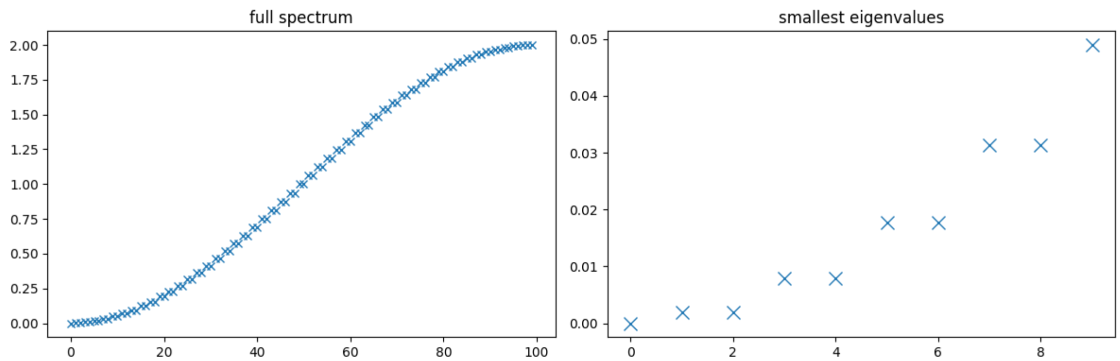
1.0.2 circle graph

[21]:
```
n=100
A=np.zeros((n,n),dtype=np.double)
A[np.arange(n),np.mod(np.arange(n)+1,n)]=1
A[np.mod(np.arange(n)+1,n),np.arange(n)]=1
plt.imshow(A)
plt.show()
```

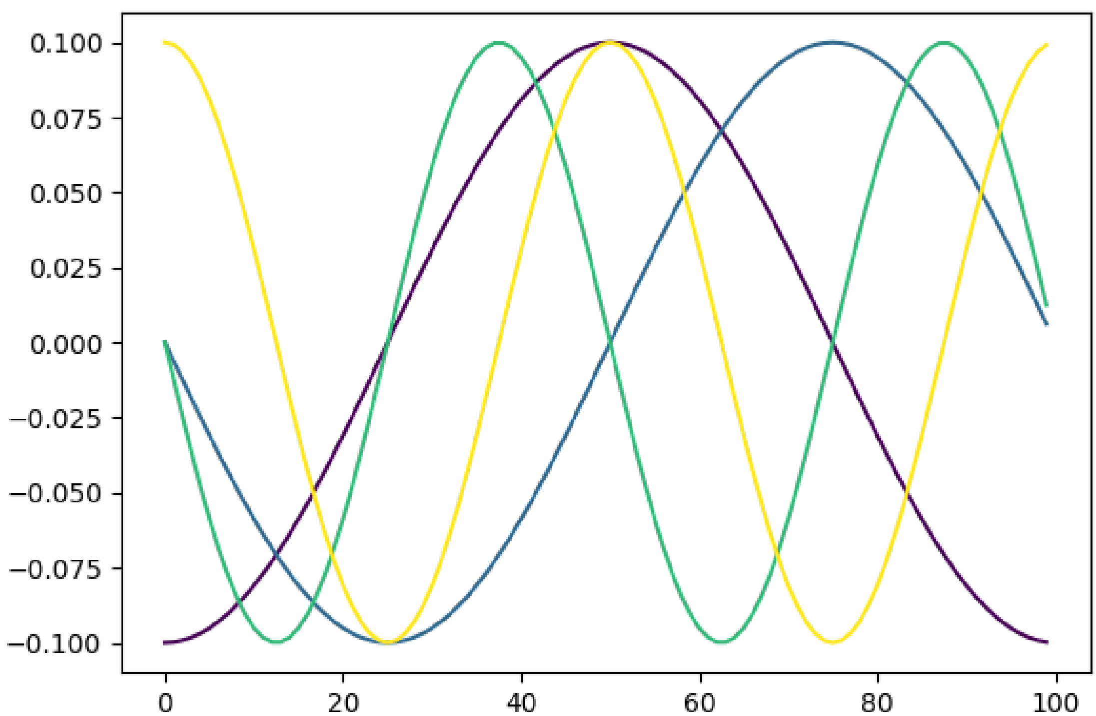


[23]:
```
# construct graph Laplacian (optional: degree-normalization)
degnorm=True
L,vecscale=getL(A,degnorm=degnorm)
eigval,eigvec=getEigdat(L)
```

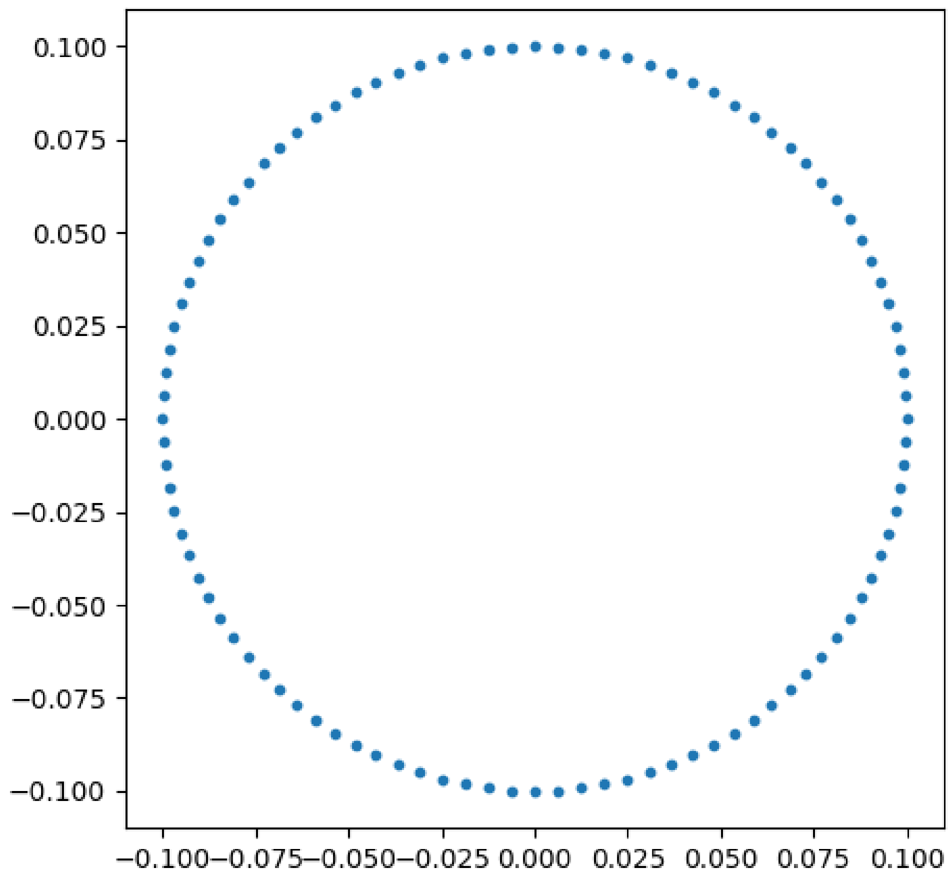
[24]:
```
# spectrum of Laplacian
fig=plt.figure(figsize=(12,4))
fig.add_subplot(1,2,1)
plt.title("full spectrum")
plt.plot(eigval,lw=0,marker="x")

fig.add_subplot(1,2,2)
plt.title("smallest eigenvalues")
plt.plot(eigval[:10],lw=0,marker="x",markersize=10)

plt.tight_layout()
plt.show()
```



[25]:
```python
# for circle graph: visualize modes as 1d functions
# should see "vibration modes"
%matplotlib inline
sel=range(1,5)
nSel=len(sel)
fig=plt.figure(figsize=(6,4))
for i in range(nSel):
    vec=eigvec[sel[i]]*vecscale
    plt.plot(vec,c=cm.viridis(i/(nSel-1)))
plt.tight_layout()
plt.show()
```



[26]:
```
# 2d embedding
%matplotlib inline
i1=1
i2=2
fig=plt.figure()
fig.add_subplot(aspect=1.)
plt.scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,s=10,marker="o")
plt.tight_layout()
plt.show()
```



[ ]:
```
i1,i2,i3=1,2,3
%matplotlib widget
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,eigvec[i3]*vecscale,alpha=0.
↪8)
```

```
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
plt.tight_layout()
plt.show()
```

[ ]:
```
plt.close()
%matplotlib inline
```

1.0.3 3elt example from file

[ ]:
```
# source: https://www.cise.ufl.edu/research/sparse/matrices/AG-Monien/3elt.html
# see also: http://yifanhu.net/GALLERY/GRAPHS/GIF_SMALL/AG-Monien@3elt.html
dat=scio.loadmat("220_Graph-Embeddings_3elt.mat")
ASparse=dat["Problem"][0][0][2]
A=ASparse.toarray()
deg=np.sum(A,axis=1)
```

[ ]:
```
# construct graph Laplacian (optional: degree-normalization)
degnorm=True
L,vecscale=getL(A,degnorm=degnorm)
eigval,eigvec=getEigdat(L)
```

[ ]:
```
# spectrum of Laplacian
fig=plt.figure(figsize=(12,4))
fig.add_subplot(1,2,1)
plt.title("full spectrum")
plt.plot(eigval,lw=0,marker="x")

fig.add_subplot(1,2,2)
plt.title("smallest eigenvalues")
plt.plot(eigval[:10],lw=0,marker="x",markersize=10)

plt.tight_layout()
plt.show()
```

[ ]:
```
# 2d embedding
%matplotlib inline
i1=1
i2=3
fig=plt.figure()
fig.add_subplot(aspect=1.)
plt.scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,s=5,marker="o",c=deg)
plt.tight_layout()
plt.show()
```

```
[ ]: i1,i2,i3=1,2,3
%matplotlib widget
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')

ax.
 ↪scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,eigvec[i3]*vecscale,c=deg,alpha=0.
 ↪8,s=5)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
plt.tight_layout()
plt.show()
```

```
[ ]: plt.close()
%matplotlib inline
```

1.0.4 Alternative weight matrix, based on distances

- this is a little bit artificial, because we do assume that positions of vertices are unknown
- but this provides larger neighbourhoods for each vertex and thus captures more local information
- we visualize how this affects the provided embeddings
- in practice many tricks are available to emulate this situation on real data

```
[ ]: nRows=10
nCols=20
# spatial dimension
dim=2

posList=getPoslistNCube((nCols,nRows),dtype=np.int32)
nPoints=posList.shape[0]

d=np.sum((posList.reshape((-1,1,dim))-posList.
 ↪reshape((1,-1,dim)))**2,axis=2)**0.5


rList=[0.1,1.,10.,100.,1000.]
fig=plt.figure(figsize=(3*len(rList),3*2))

for i,r in enumerate(rList):
    A=np.exp(-d/r)
    for j in range(nPoints):
        A[j,j]=0

    # construct graph Laplacian (optional: degree-normalization)
```

```
    degnorm=True
    deg=np.sum(A,axis=1)

    L,vecscale=getL(A,degnorm=degnorm)
    eigval,eigvec=getEigdat(L)


    i1=1
    i2=2
    fig.add_subplot(2,len(rList),i+1)
    plt.title(r)
    plt.imshow(A)
    plt.axis("off")
    fig.add_subplot(2,len(rList),len(rList)+i+1)
    plt.scatter(eigvec[i1]*vecscale,eigvec[i2]*vecscale,s=10,marker="o")
plt.tight_layout()
plt.show()
```

[ ]:

```

```