Scientific computing: Project 2

Tomás Fernández Bouvier

September 20, 2020

a)

```python
def gershgorin(A):
  centres=[]
  radii=[]
  for i in range (np.shape(A)[0]):
    c=sum(abs(A[i, :]))-abs(A[i,i])
    r=sum(abs(A[:, i]))-abs(A[i,i])
    centres.append(A[i,i])
    radii.append(min([c,r]))
  return(centres, radii)
```

Centres	129292.219206	103041.439420	64967.578727	43612.411909	36273.751516
Radii	41286.472107	55632.012643	27975.116399	18532.348737	7870.516137

Centres	37990.099373	24166.971120	11651.158690	13865.080461	5600.547665
Radii	17126.097372	15411.008598	2512.041395	5502.969527	1195.283294

Centres	1173.038787	1760.771353	288.423061	86.896891	13.893346
Radii	341.990248	401.214726	71.417319	2.540294	0.259095

Table 1: Centres and Radii of the Gershgorin's circles obtained through the above algorithm for **K**

b)

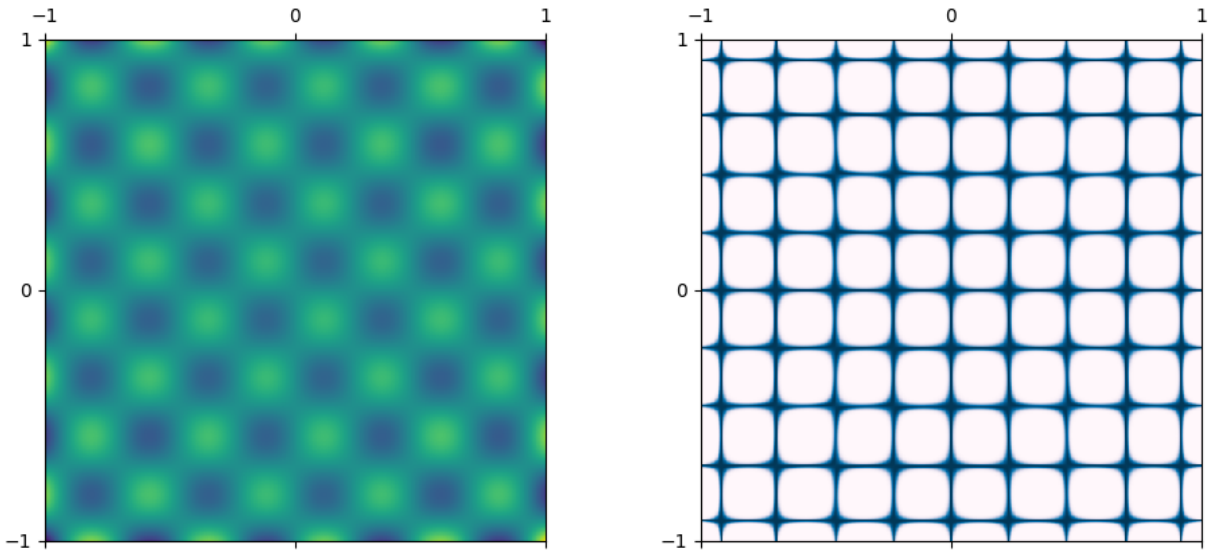
```python
def rayleigh_qt(A,x):
  landa= (x.T@A@x)/(x.T@x)
  return(landa)

def power_iterate(A):
  x=np.random.rand(np.shape(A)[0])
  k=0
  res=2000000
  while(res>10**(-5)):
    rq1=rayleigh_qt(A,x)
    y=A@x.T
```

```
    x= y/max(abs(y))
    rq2=rayleigh_qt(A,x)
    res=abs(rq2-rq1)
    k+=1
  return(x,k,res)
```

Matrix	A1	A2	A3	A4	A5	A6
λ_{max}	4.00	4.00	12.30	16.12	68.64	2.00
iterations	7	9	13	6	6	8
Res (10^{-6})	4.84	4.61	4.78	2.33	7.12	4.90

Table 2: Results obtained using the algorithm above on the example matrixes

The largest eigenvalue of $\mathbf{K}$ is 151362.67, obtained with 42 iterations and with $Res = 6.62 \cdot 10^{-6}$. Below are the respective plots obtained using show_waves and show_nodes applied on the respective eigenvector:



(a) waves (b) nodes

Figure 1: Highest eigenvalue mode

c)

```
def rayleigh_iterate(A, x0, shift0):

  x=x0
  I=np.identity(np.shape(A)[0])
  res= 1
  shift=shift0
  iter=0

```

```
  while(res>10**(-5)):
    if(np.linalg.det(A-shift*I)==0.):
      res=0
    else:
      x=np.linalg.solve(A-shift*I,x) # I used a solver from numpy
      x=x/max(abs(x[:]))
      shift= (x.T@A@x)/(x.T@x)
      res= np.linalg.norm(A@x.T-shift*x.T)
    iter+=1
    if(iter==10**4):
      res=0
  return(x, shift, iter, res)
```

My function works for all the matrices and computes the proper eigenvalues even for **A6**:

λ	$\vec{x}$	res (10^{-6})	iter
0	$(0.02759628, -1.00, 0.59551759, 0.20117056, -0.57388993)$	1.1	4
1	$(-0.24773878, 0.00133725, 0.68947447, -1.0.35067647)$	$1.42 \cdot 10^{-6}$	4
2	$(0.67888766, 0.52615739, 1.00, 0.6565257, 0.38364258)$	$1.75 \cdot 10^{-2}$	7
2	$(-0.88398519, -0.58193246, -1.00, -0.56376392, -0.26380064)$	$2.4 \cdot 10^{-7}$	4
2	$(0.85194566, 0.27999458, 1.00, 0.78209554, 0.8649187)$	$5.74 \cdot 10^{-5}$	5

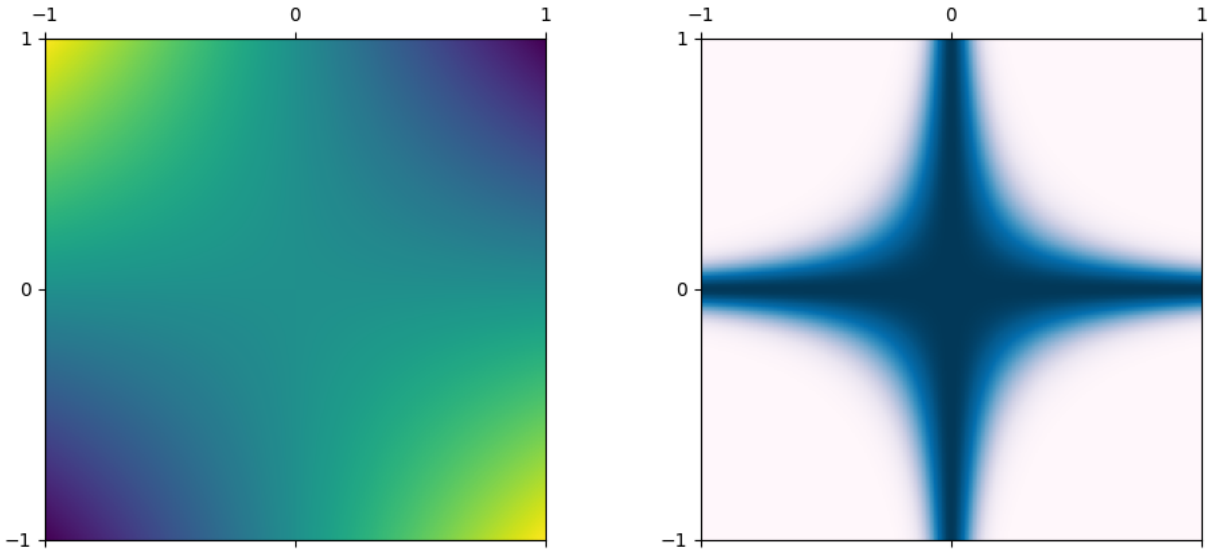
d)

My program was able to find all the eigenvalues and eigenvectors of the matrix **K** under the asumption that there is not multiplicity in it. My initial approach was to calculate the Gershgorin's circles so I would have a starting point for then calculating the eigenvalues and eigenvectors using rayleigh_iterate (see above). The problem is that some circles overlap so some of them would yield the same eigenvalue. Therefore what I deed is to take as a starting point the circle's centres and their edges. I obtained a huge list of possible eigenvalues from which I chose the ones that appeared only once in the list.

```
def find_eigenvalues(A):
  centres, radii= gershgorin(A)
  eval=[]
  evec=[]
  obj=len(centres)
  for i in range(len(centres)):
    dwshift=centres[i]- radii[i]
    upshift=centres[i] + radii[i]
    centreshift=centres[i]
    dwevec, dweval, _, _ =  rayleigh_iterate(A,10*np.random.rand(np.shape(A)[0]),
   dwshift)
    upvec,upval, _, _ = rayleigh_iterate(A,10*np.random.rand(np.shape(A)[0]),
   upshift)
    centrevec,centreval, _, _ = rayleigh_iterate(A,10*np.random.rand(np.shape(A)
   [0]),upshift)

    eval.append(dweval); eval.append(upval); eval.append(centreval)
    evec.append(dwevec); evec.append(upvec); evec.append(centrevec)
  evec=np.array(evec)
  eval2=[]
  evec2=[]
```

```
80
81    for i in range(len(eval)):
82      k=0
83      for j in range(len(eval2)):
84        if(abs(eval[i]-eval2[j])<=10**(-1)):
85          k+=1
86      if(k==0):
87        eval2.append(eval[i])
88        evec2.append(evec[i])
89
90    return(eval2, evec2,obj)
91
92 obj=1; eval2=[];
93 while(len(eval2)!=obj):
94    eval2,evec2, obj=find_eigenvalues(Kmat)
95 evec2=np.array(evec2)
96
97 minvec=evec2[np.argsort(eval2)[0],:]
98 show_waves(minvec, basis_set)
99 show_nodes(minvec, basis_set)
```

This is the figure that I obtain for the eigenvector associated to the lowest eigenvalue:



(a) waves (b) nodes

Figure 2: Lowest eigenvalue mode

I also constructed the matrix $\mathbf{T}$ which column's are the eigenvectors of $\mathbf{K}$ and after checking that the first diagonalises the latter (i.e. $\mathbf{T}^{-1}\mathbf{K}\mathbf{T} = \Lambda$).

```
103 T=[]
104
105 for i in range(np.shape(evec2)[0]):
106    T.append(evec2[np.argsort(eval2)[i], :])
107
```

```
T=(np.array(T)).T

lambdas=np.sort(eval2)

print(np.diag(np.linalg.inv(T)@Kmat@T))
print(lambdas)

show_all_wavefunction_nodes(T,lambdas,basis_set)
```

Finally, I used show_all_wavefunction_nodes() on my results in order to obtain the final plot below:

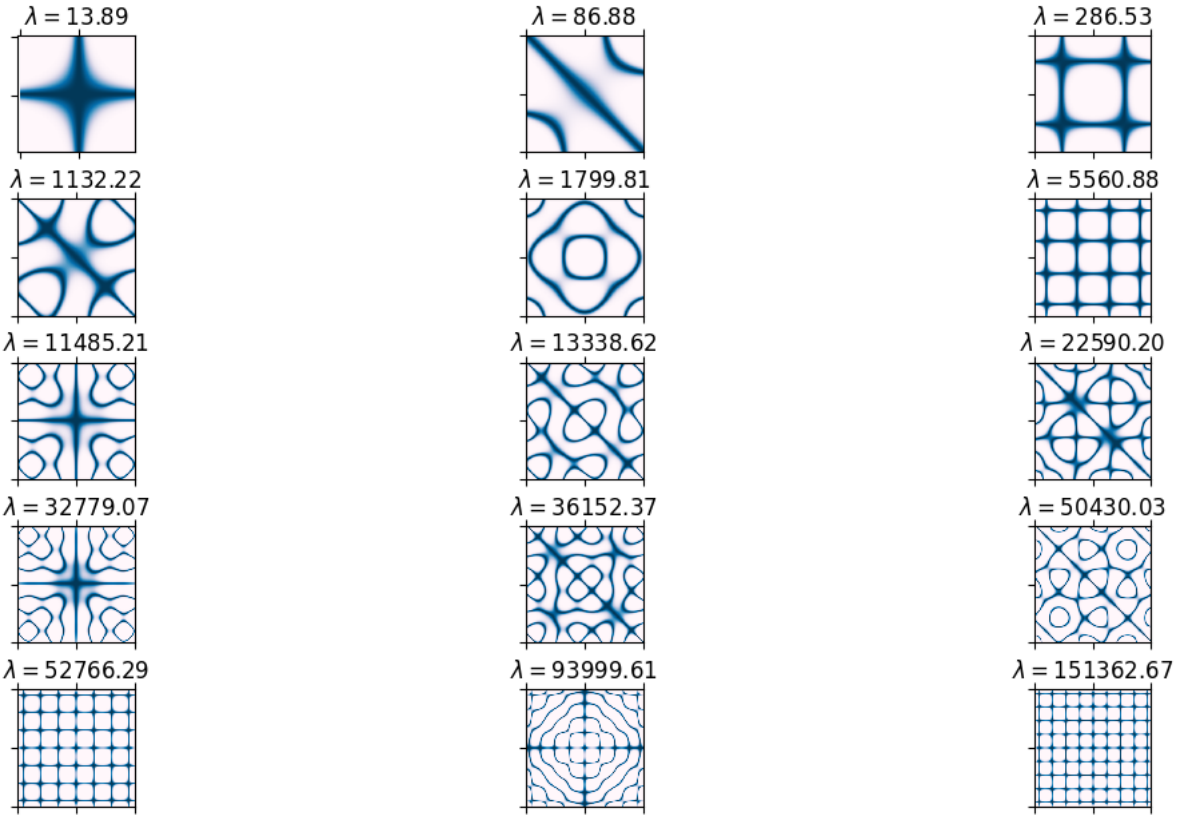


Figure 3: All modes of vibration of the system