Math 151B HW6

March 8, 2020

1 Math 151B Homework No. 6

- Brandon Loptman
- UID: 604105043
- March 10, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1.0.1 9.3 The Power Method

8.)

[17]:
```
def PowerMethod(A,x,n,tol=10e-4, N=25):
    """
    Implements the Power Method as described by the algorithm in the textbook.
    """
    k = 1

    xp = np.max(abs(x))
    x = x/xp

    while(k <= N):
        #print(k)
        y = A.dot(x)

        yp = np.max(np.abs(y))
        u = yp
        #print(u)

        if(yp == 0):
            print("A has eigenvalue 0, select a new vector x and restart.")
            return x

        err = np.max(np.abs(x-(y/yp)))
```

```
        x = y/yp
        #print(x)

        if(err < tol):
            print("The procedure was successful!")
            return u,x

        k = k + 1

    print("The maximum number of iterations exceeded! The procedure was␣
↪unsuccessful.")
    return u,x
```

(a.)

[ ]:
```
A1 = np.array([[4,2,1],[0,3,2],[1,1,4]])
x1 = np.array([1,2,1])
n1 = 3

u,v = PowerMethod(A1,x1,n)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

(b.)

[118]:
```
A2 = np.array([[1,1,0,0],[1,2,0,1],[0,0,3,3],[0,1,3,2]])
x2 = np.array([1,1,0,1])
n2 = 4

u,v = PowerMethod(A2,x2,n2)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  5.666647958421926
Eigenvetor:  [0.0554278  0.25788625 1.         0.88873413]
```

(c.)

[119]:
```
A3 = np.array([[5,-2,-0.5,1.5],[-2,5,1.5,-0.5],[-0.5,1.5,5,-2],[1.5,-0.5,-2,5]])
x3 = np.array([1,1,0,-3])
n3 = 4

u,v = PowerMethod(A3,x3,n3)

print("Eigenvalue: ", u)
```

```
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  8.996446247045956
Eigenvetor:  [-0.99903998  0.9990224   0.9999824  -1.        ]
```

(d.)

[120]:
```
A4 = np.array([[-4,0,0.5,0.5],[0.5,-2,0,0.5],[0.5,0.5,0,0],[0,1,1,4]])
x4 = np.array([0,0,0,1])
n4 = 4

u,v = PowerMethod(A4,x4,n4)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The maximum number of iterations exceeded! The procedure was unsuccessful.
Eigenvalue:  4.1202713104915745
Eigenvetor:  [0.10396255 0.07624767 0.01441345 1.        ]
```

10.)

[114]:
```
def InversePowerMethod(A,x,n,tol=10e-4,N=25):
    """
    Implements the Inverse Power Method as described by the algorithm in the␣
 ↪textbook.
    """
    q = (np.transpose(x)).dot(A.dot(x))/(np.transpose(x)).dot(x)
    #print("q = ", q)

    k = 1

    xp = np.max(abs(x))
    x = x/xp

    qI = np.identity(n)*q

    while(k <= N):
        #print(k)

        y = np.linalg.solve(A-qI,x)
        #print(y)

        p = np.argmax(np.abs(y))
        yp = y[p]

        u = yp
```

```
        #print("u = ", u)

        err = np.max(np.abs(x-(y/yp)))
        x = y/yp
        #print(x)

        if(err < tol):
            u = (1/yp) + q
            #print("u = ", u)

            print("The procedure was successful!")
            return u,x

        k = k + 1

    print("The maximum number of iterations exceeded! The procedure was unsuccessful.")
    return u,x
```

[128]:
```
A = np.array([[-4,14,0],[-5,13,0],[-1,0,2]])
x = np.array([1,1,1])
n = 3

u,v = InversePowerMethod(A,x,n)
print(u)
print(v)
```

```
The procedure was successful!
6.000171438367904
[ 1.          0.71429796 -0.24995339]
```

(a.)

[122]:
```
u,v = InversePowerMethod(A1,x1,n1)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  5.91952739334497
Eigenvetor:  [1.         0.55480496 0.80991748]
```

(b.)

[123]:
```
u,v = InversePowerMethod(A2,x2,n2)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  2.6459223598633343
Eigenvetor:  [ 0.60757625  1.         -0.32509997  0.03834611]
```

(c.)

[124]:
```
u,v = InversePowerMethod(A3,x3,n3)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  3.999973870192454
Eigenvetor:  [0.99999843 0.99993845 1.         0.99993999]
```

(d.)

[125]:
```
u,v = InversePowerMethod(A4,x4,n4)

print("Eigenvalue: ", u)
print("Eigenvetor: ", v)
```

```
The procedure was successful!
Eigenvalue:  4.105293014436679
Eigenvetor:  [0.06281419 0.08704089 0.01825213 1.        ]
```

1.0.2 9.4 Householder's Method

1.)

[82]:
```
def column_convertor(x):
    """
    Converts 1d array to column vector
    """
    x.shape = (1, x.shape[0])
    return x

def get_norm(x):
    """
    Returns Norm of vector x
    """
    return np.sqrt(np.sum(np.square(x)))

def householder_transformation(v):
    """
    Returns Householder matrix for vector v
    """
    size_of_v = v.shape[1]
    e1 = np.zeros_like(v)
    e1[0, 0] = 1
```

```
    vector = get_norm(v) * e1
    if v[0, 0] < 0:
        vector = - vector
    u = (v + vector).astype(np.float32)
    H = np.identity(size_of_v) - ((2 * np.matmul(np.transpose(u), u)) / np.
 ↪matmul(u, np.transpose(u)))
    return H

def qr_step_factorization(q, r, iter, n):
    """
    Return Q and R matrices for iter number of iterations.
    """
    v = column_convertor(r[iter:, iter])
    Hbar = householder_transformation(v)
    H = np.identity(n)
    H[iter:, iter:] = Hbar
    r = np.matmul(H, r)
    q = np.matmul(q, H)
    return q, r
```

[ ]: