Numerical Analysis II: Homework 5

Peter Shaffery

March 1, 2017

1 Problems

Problem 1. Implement the QR iteration algorithm and try it out on a couple of tridiagonal, symmetric, real matrices. What is the complexity of this algorithm?

See Appendix for code (also available on GitHub under /petershaff/numerics2). Tested this script on some random matrices of different sizes and had no problem. Applying the 'does this script work' function (see Append; this calculates the sum-of-squares error, scaled by the matrix dimension, between the eigenvalues calculated with my QR iteration and those calculated using np.eig()) to 100 random matrices with $n = 3, 4, 5$ I got a median, scaled error on the order of 1e-14 for all n.

For an arbitrary matrix the complexity of the Housholder QR decomposition is $\mathcal{O}(n^4)$. To see this consider that calculating u is $\mathcal{O}(n)$, calculating H is $\mathcal{O}(n^2)$ (the outer product and the matrix subtraction are both $\mathcal{O}(n^2)$, so combined the whole thing is just $\mathcal{O}(n^2)$), and then the matrix multiplication at the end is $\mathcal{O}(n^3)$ therefore a single step is $\mathcal{O}(n^3)$. Since we are performing an $\mathcal{O}(n^3)$ operation at each step, and there are $n-1$ steps then the whole thing is $\mathcal{O}(n^3(n-1)) = \mathcal{O}(n^4)$.

Problem 2. Compare QR steps with a shift and without.

For any values of ϵ I used the shift step performed better than the no-shift step (see Appendix). For all values of ϵ I used, the lower left element after one shift step was slightly smaller in magnitude than the same element after a no-shift step. This is because the eigenvalues of the matrix when ϵ is small are close to$\lambda_1 = 2$ and $\lambda_2 = 1$, so shifting with $\mu = 1$ sends the eigenvalues to something near $\lambda_1 = 1$ and $\lambda_2 = 0$. The rate of convergence (for k steps) for the lower left element to 0 in the QR-iteration is $\frac{|\lambda_2|}{|\lambda_1|}^k$, so if $\lambda_2 \approx 0$ then the algorithm converges faster.

2 Appendix

```
import numpy as np
############
#Question 1#
############
def qr_decomp(A):
    if np.shape(A)[0] != np.shape(A)[1]:
        print('Not a square matrix, fool!')
        return([A,A])

    else:
        dim = np.shape(A)[0]
        R = A.copy()
        Q = np.eye(dim)
        for i in range(0,dim-1):
            a = R[:,i].copy()
            a[0:i] = 0
            r = np.zeros(dim)
            r[i] = np.linalg.norm(a,ord=2)
            u = a - r
            H = np.eye(dim) - np.outer(u,u)*2*(1/np.linalg.norm(u,ord=2)**2)
            Q = np.dot(Q,H)
            R = np.dot(H,R)
        return([Q,R])

def qr_iter(A, tol=1e-6):
    err = tol + 1
    L = A
    t = 0
    dim = np.shape(A)[0]
    mu = np.mean(np.diag(A))
    if np.shape(A)[0] != np.shape(A)[1]:
        print('Still not square(, dork.')
        return(A)

    try:
        while err > tol:
            q,r = qr_decomp(L)# - mu*np.eye(dim))
            L = np.dot(r,q)# + mu*np.eye(dim)
            err = np.linalg.norm(np.tril(L,k=-1),ord=np.inf)
            t+=1

    except KeyboardInterrupt:
        return(L)

    return([L,t])

def rand_tridiag(n):
    test = np.zeros([n,n])
    for i in range(0,n):
        if i==0:
            test[i,i] = np.random.rand(1)
            test[i+1,i] = np.random.rand(1)

        elif i < n-1:
            test[i+1,i] = np.random.rand(1)

        else:
            test[i-1,i] = test[i,i-1]
            test[i,i] = np.random.rand(1)

    return(10*test)

def does_this_script_work(n):
    test = rand_tridiag(n)
    eigs = np.sort(np.diag(qr_iter(test)[0]))
    err = np.linalg.norm(eigs - np.sort(np.linalg.eig(test)[0]))
    return(err/n)

############
#Question 2#
############
compare_list = []
for eps in np.arange(1e-4,1e-1,1e-5):
    A = np.array([[2,eps],[eps,2]])
    #Without shift:
    q,r = qr_decomp(A)
    L1 = np.dot(r,q)
    #With shift:
    q,r = qr_decomp(A-np.eye(2))
    L2 = np.dot(r,q) + np.eye(2)

    #Record which did better
    compare_list.append(abs(L1[1,0]) - abs(L2[1,0]))

#Did the shift help at all?
print(any(np.array(compare_list)>0))
```