AMATH 584 Assignment 8

Tyler Chen

Exercise 26.1

Theorem 26.1 and its successors in later lectures show that we can compute eigenvalues $\{\tilde{\lambda_k}\}$ of A numerically that are the exact eigenvalues of a matrix $A + \delta A$ with $\|\delta A\| / \|A\| = \mathcal{O}(\epsilon_{\text{mach}})$. Does this mean they are close to the exact eigenvalues $\{\lambda_k\}$ of A? This is a question of eigenvalue perturbation theory.

One can approach such problems geometrically as follows. Given $A \in \mathbb{C}^{m\times m}$ with spectrum $\Lambda(A) \subseteq \mathbb{C}$ and $\epsilon > 0$, define the 2-norm ϵ-pseudospectrum of A, $\Lambda_\epsilon(A)$, to be the set of numbers $z \in \mathbb{C}$ satisfying any of the following conditions:

(i) z is an eigenvalue of $A + \delta A$ for some δA with $\|\delta A\|_2 \leq \epsilon$;
(ii) There exists a vector $u \in \mathbb{C}^m$ with $\|(A - zI)u\|_2 \leq \epsilon$ and $\|u\|_2 = 1$;
(iii) $\sigma_m(zI - A) \leq \epsilon$;
(iv) $\left\|(zI - A)^{-1}\right\|_2 \geq \epsilon^{-1}$

The matrix $(zI - A)^{-1}$ is known as the resolvent of A at z. If z is an eigenvalue of A, we use the convention $\left\|(zI - A)^{-1}\right\|_2 = \infty$. σ_m denotes the smallest singular value.

Prove the above conditions are equivalent.

Use the results of this exercise to show that if A is a normal matrix and $\tilde{\lambda}$ is an eigenvalue of $A + \delta A$ for some δA with $\|\delta A\|_2 \leq \epsilon$, then there is an eigenvalue λ of A such that $|\tilde{\lambda} - \lambda| \leq \epsilon$. Would you say that the problem of computing eigenvalues of a normal matrix is well-conditioned or ill-conditioned?

Solution

(i) $\Rightarrow$ (ii) Suppose z is an eigenvalue of $A + \delta A$ for some δA with $\|\delta A\| \leq \epsilon$. Let u be a corresponding eigenvector with $\|u\| = 1$. Then, $(A+\delta a)u = zu$ so $Au - zu = (A - zI)u = -\delta Au$. Taking the norm of each side we have, $\|(A - zI)u\|_2 = \|-\delta Au\| \leq \|\delta A\| \, \|u\| \leq \epsilon$.

(ii) $\Rightarrow$ (iii) The smallest singular value, σ_m is minimum stretch of a vector under $(zI - A)$. That is, $\|(zI - A)v\| \geq \sigma_m \|v\|$ for all v. Therefore, in particular, $\sigma_m(zI - A) \leq \|(zI - A)u\| / \|u\| = \|(zI - A)u\| \leq \epsilon$.

(iii) $\Rightarrow$ (iv) Recall the largest singular value of a matrix is the reciprocal of the smallest singular value of its inverse. We then have, $\left\|(zI - A)^{-1}\right\|_2 = \sigma_{\max}((zI - A)^{-1}) = \sigma_{\min}(zI - A)^{-1} \geq \epsilon^{-1}$

(iv) $\Rightarrow$ (i) Let u, v be the first left and right singular vectors of $(zI - A)^{-1}$ with singular value $\sigma^{-1} := \left\|(zI - A)^{-1}\right\|$. Then $\|u\| = \|v\| = 1$ and $\sigma \leq \epsilon$ and by definition, $(zI - A)^{-1}v = \sigma^{-1}u$. Rearranging we have, $\sigma v = (zI - A)u$ so $(A - \sigma vu^*)u = Au - \sigma vu^*u = Au - \sigma v = zu$. This means z is an eigenvector of $A - \sigma vu^*$ and $\|\sigma vu^*\| = \sigma \|v\| \, \|u^*\| = \sigma \leq \epsilon$.

This proves the four statements are equivalent. □

Suppose A is a normal matrix and $\tilde{\lambda}$ is an eigenvalue of $(A + \delta A)$ for some δA with $\|\delta A\| \leq \epsilon$.

Since A is normal, it has a unitary diagonalization, $A = U\Lambda U^*$. Then $A - zI = U\Lambda U^* - zUIU^* = U(\Lambda - zI)U^*$. We can write this as an SVD by moving the any negative entries of $\Lambda - zI$ to the U on the left. We write the SVD as, $A = U|\Lambda - zI|U^*$, noting that U might have different signs than before.

By hypothesis, $\tilde{\lambda}$ is an eigenvalue of $A+\delta A$ where $\|\delta A\| \leq \epsilon$. Equivalently, $\sigma_m := \sigma_m(\tilde{\lambda}I - A) \leq \epsilon$. From the SVD above we have, $\sigma_m u = (\tilde{\lambda}I - A)u = U|zI - \Lambda|U^*u$, where u is the last singular vector (the one corresponding to σ_m). So $\sigma_m U^*u = |\tilde{\lambda}I - \Lambda|U^*u$. Then looking at bottom entry of these vectors we have $|\tilde{\lambda} - \lambda_m| = \sigma_m \leq \epsilon$.

Since λ_m is an entry of Λ, λ_m is an eigenvalue of A. The result is then proved. □

> The problem of computing eigenvalues of a normal matrix is well conditioned since a small change in A leads to a similarly small change in the eigenvalues.

Exercise 2

Let

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad A_\epsilon = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \epsilon & 0 & 0 \end{bmatrix}.$$

What are the eigenvalues of A and of A_ϵ? Would you say that the problem of computing the eigenvalues of a matrix A of this form is well-conditioned or ill-conditioned?

Solution

We have,

$$p_A(z) = \det(A - zI) = -z \det \begin{bmatrix} -z & 1 \\ 0 & -z \end{bmatrix} = -z^3$$
$$p_{A_\epsilon}(z) = \det(A_\epsilon - zI) = -z \det \begin{bmatrix} -z & 1 \\ 0 & -z \end{bmatrix} + \epsilon \det \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = -z^3 + \epsilon$$

Therefore the eigenvalues of A are all zero, and the eigenvalues of A_ϵ are $\epsilon^{1/3}, \epsilon^{1/3}e^{2\pi i/3}, \epsilon^{1/3}e^{4\pi i/3}$.

We consider the problem of computing eigenvalues as a map $f : \mathbb{C}^{m\times m} \to \mathbb{C}^m$, where the input is a matrix, and the output is a vector of the eigenvalues.

The relative condition number is not a very useful measure of the conditioning of a problem when the input or output to the unperturbed problem is zero.

The absolute condition number is,

$$\kappa := \lim_{\delta\to 0} \sup_{\|\delta x\|\le\delta} \left(\frac{\|\delta f\|}{\|\delta x\|} \right)$$

In this case, we have $\|\delta A\| = \|A - A_\epsilon\| = \epsilon$ and $\|A\| = 1$. Likewise, we have $\|f(A)\| = \|[0, 0, 0]\| = 0$ and $\|f(\delta A)\| = \left\|[\epsilon^{1/3}, \epsilon^{1/3}e^{2\pi i/3}, \epsilon^{1/3}e^{4\pi i/3}]\right\| = 3^{1/3}\epsilon^{1/3}$. Therefore,

$$\kappa \ge \left(\frac{\|\delta f\|}{\|\delta x\|} \right) = \left(\frac{3^{1/3}\epsilon^{1/3}}{\epsilon} \right) \ge \epsilon^{-2/3}$$

If epsilon is small then this is large. That is, a small change to the problem leads to a large change to the output of the problem. Therefore, the problem of computing eigenvalues this matrix is ill conditioned.

More generally, a matrix with zeros everywhere except for ones on the main superdiagonal will will have eigenvalues 0, and eigenvalues equal to the m-th roots of ϵ when perturbed with an error of ϵ in the bottom left corner. So, similar to this computation, calculating the eigenvalues of these matrices will never be well conditioned.

Exercise 28.1

What happens if you apply the unshifted QR algorithm to an orthogonal matrix? Figure out the answer, and then explain how it relates to Theorem 28.4.

Solution

The unshifted QR algorithm is,

$A^{(0)} = A$
`for` $k = 1, 2, \ldots$:
    $[Q^{(k)}, R^{(k)}]$ = `qr(`$A^{(k-1)}$`)`
    $A^{(k)} = R^{(k)} Q^{(k)}$

Suppose A is orthogonal. Then in the first step we have, $[Q^{(1)}, R^{(1)}] = [Q, I]$ since the QR decomposition of an orthogonal matrix is just itself times the identity. Then at each stage $A^{(k)} = A^{(k-1)}$ so the algorithm will never converge.

If Q is orthogonal then all eigenvalues have magnitude 1. So the theorem is not satisfied since there is not distinct eigenvalue with largest magnitude.

Exercise 4

Consider the matrix

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

Taking A to be a 10 by 10 matrix, try the following:

(a) What information does Gerschgorin's theorem give you about this matrix?

(b) Implement the power method to compute an approximation to the eigenvalue of largest absolute value and its corresponding eigenvector. [Note: Use a random initial vector. If you choose something special, like the vector of all 1's, it may turn out to be orthogonal to the eigenvector you are looking for.] Turn in a listing of your code together with the eigenvalue/eigenvector pair that you computed. Once you have a good approximate eigenvalue, look at the error in previous approximations and comment on the rate of convergence of the power method.

(c) Implement the QR algorithm (without shifts) to take A to diagonal form. You may use Matlab routine `[Q,R] = qr(A);` to perform the necessary QR decompositions. Comment on the rate at which the off-diagonal entries in A are reduced.

(d) Take one of the eigenvalues computed in your QR algorithm and, using it as a shift in inverse iteration, compute the corresponding eigenvector.

Solution

Define,

```
A = np.triu(np.tril(-np.ones((10,10))+3*np.identity(10),1),-1)
```

(a) We know all the eigenvalues are found in the disk: $\{z : |z-2| \leq 2\}$. Since A is symmetric, (and therefore Hermetian) all eigenvalues are real. Therefore they all like within the closed interval $[0, 4]$.

(b) We iterate until the Raleigh quotient changes by less than the specified tolerance (machine epsilon). The code is implemented in python as,

```
def power(A):
    tol=10e-16
    l=[0,1]
    v = np.random.rand(10)
    v = v/np.linalg.norm(v)
    while abs(l[-1] - l[-2]) > tol:
        w = A@v
        v = w/np.linalg.norm(w)
        l.append(v.T@A@v)
    return [l[-1],v]
```

This gives output,

```
[3.9189859472289839,
 array([ 0.1201312144953602, -0.2305301009433062,  0.3222527902846078,
        -0.3878684540194245,  0.4220613062809278, -0.4220612556116857,
         0.3878683180988246, -0.32225261226648  ,  0.2305299373471484,
        -0.1201311172617966])]
```

Figure 1 shows a plot of the log of $|\lambda^{(k)} - \lambda_\infty|$. As expected convergence is linear.

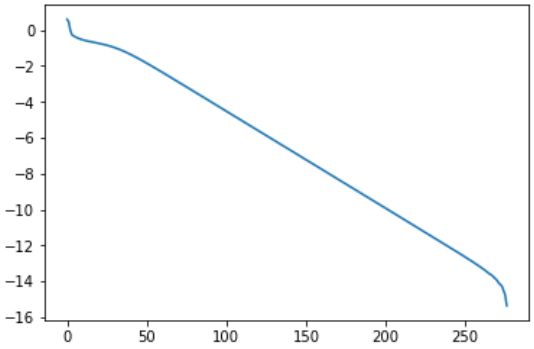

Figure 1: $\log|\lambda^{k+1} - \lambda_\infty|$

(c) We implement the unshifted QR algorithm in python as,

```
def qr(A):
tol=10e-16
A_old = np.identity(len(A))
while abs(A[0,0] - A_old[0,0]) > tol:
    [Q,R] = np.linalg.qr(A)
    A=R@Q

return A
```

Figure 2 shows a plot of the log of the two norm of the entries of the first subdiagonal. It appears the convergence is linear

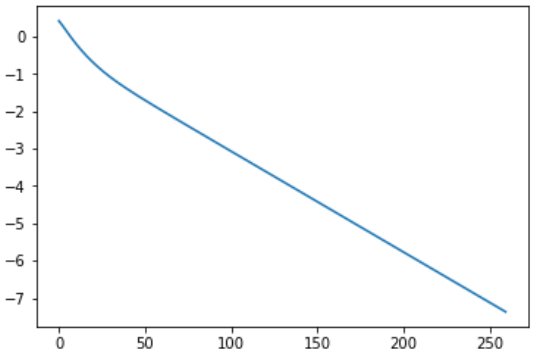

Figure 2: $\log\left\|\left[A^{(k)}_{i,i+1}\right]_{i=1}^{m-1}\right\|$ vs iteration

(d) We implement inverse iteration in python as,

```
def inviter(A,s):
    tol=10e-16
    l=[0,1]
    v = np.random.rand(10)
    v = v/np.linalg.norm(v)
    while abs(l[-1] - l[-2]) > tol:
        w = np.linalg.solve(A-s*np.identity(len(A)),v)
        v = w/np.linalg.norm(w)
        l.append(v.T@A@v)

    return [l[-1],v]
```

Using `s = qr(A)[3] = 2.8308300260037638` we have output,

```
array([ 3.9189859472289963,  3.6825070656623637,  3.309721467890574 ,
        2.8308300260037744,  2.2846296765465772,  1.7153703234534301,
        0.0810140527710047,  0.3174929343376377,  0.6902785321094301,
        1.1691699739962271])
```