AMATH 584 Assignment 7

Tyler Chen

Exercise 1

Write a routine to generate an m by m matrix with a given 2-norm condition number. You can make your routine a `function` in Matlab that takes two input arguments – the matrix size m and the desired condition number `condno` – and produces an m by m matrix A with the given condition number as output:

```
function A = matgen(m, condno)
```

Form A by generating two random orthogonal matrices U and V and a diagonal matrix Σ with $\sigma_{jj} = \text{condno}^{-(j-1)/(m-1)}$, and setting $A = U\Sigma V^*$. [Note that the largest diagonal entry of Σ is 1 and the smallest is condno^{-1}, so the ratio is `condno`.] You can generate a random orthogonal matrix in Matlab by first generating a random matrix, `Mat = randn(m)`, and then computing its QR decomposition, `[Q,R] = qr(Mat)`. The matrix `Q` is then a random orthogonal matrix. You can check the condition number of the matrix you generate by using the function `cond` in Matlab. Turn in a listing of your code.

For `condno`$= (1,\ 10^4,\ 10^8, 10^{12}, 10^{16})$, use your routine to generate a random matrix `A` with condition number `condno`. Also generate a random vector `xtrue` of length m and compute the product `b = A*xtrue`.

(a) Solve $Ax = b$ using Gaussian elimination with partial pivoting. This can be done in Matlab by typing `x = A\b`. Determine the 2-norm of the error `norm(x - xtrue)/norm(xtrue)` in your computed solution and explain how this is related to the condition number of A. Compute the 2-norm of the residual, `norm(b-A*x)/(norm(A)*norm(x))`. Does the algorithm for solving $Ax = b$ appear to be backward stable (at least for this problem); that is, is the computed solution the exact solution to a nearby problem?

(b) Solve $Ax = b$ by inverting A and multiplying by the inverse: `Ainv = inv(A); x = Ainv*b`. Again look at relative errors and residuals. Does this algorithm appear to be backward stable?

(c) Finally, solve $Ax = b$ using Cramer's rule (i.e., compute the determinant of A by typing `det(A)` and then compute `x(j)` by replacing column j of A by the right-hand side vector b, computing the determinant of the resulting matrix A_j and finding the ratio: `det(A_j)/det(A)`). Again look at relative errors and residuals and determine whether this algorithm is backward stable.

Turn in a table showing the relative errors and residuals for each of the three algorithms and each of the condition numbers tested, along with a brief explanation of the results.

Solution

We implement the function `matgen` in python as,

```
def matgen(m,condno):
    [U,X] = np.linalg.qr(np.random.randn(m,m))
    [V,X] = np.linalg.qr(np.random.randn(m,m))
    S = np.diag(condno**((1-np.linspace(1,m,m))/(m-1)))
    return U@S@V
```

We implement the methods of solving as:

```
def exercise_1():
    ge_err,inv_err,cr_err = [],[],[]
    m = 20
    for condno in [0,4,8,12,16]:

        A = matgen(m,10**condno)
        xtrue = np.random.rand(m)
        b = A@xtrue

        x_ge = np.linalg.solve(A,b)
        ge_err.append([condno, np.linalg.norm(x_ge-xtrue)/np.linalg.norm(
            xtrue),
                np.linalg.norm(b-A@x_ge)/(np.linalg.norm(A)*np.linalg.norm(
                    x_ge))])

        Ainv =  np.linalg.inv(A)
        x_inv = Ainv@b
        inv_err.append([condno, np.linalg.norm(x_inv-xtrue)/np.linalg.norm(
            xtrue),
                np.linalg.norm(b-A@x_inv)/(np.linalg.norm(A)*np.linalg.norm(
                    x_inv))])

        detA = np.linalg.det(A)
        x_cr = np.zeros(m)
        for j in range(m):
            A_j = copy.deepcopy(A)
            A_j[:,j] = b
            x_cr[j] = np.linalg.det(A_j)/detA
        cr_err.append([condno, np.linalg.norm(x_cr-xtrue)/np.linalg.norm(
            xtrue),
                np.linalg.norm(b-A@x_cr)/(np.linalg.norm(A)*np.linalg.norm(
                    x_cr))])

    return [ge_err,inv_err,cr_err]
```

Note that the outputs are the condition number, the normalized error, and the normalized residual.

(a) Note that the linear solver in numpy is implemented using LAPACK routine _gesv (Gaussian elimination with partial pivoting).

The algorithm appears backward stable since the residuals are all order ϵ_{mach}.

```
[[0,   4.9733030843277515e-16, 1.0709777379963362e-16],
 [4,   1.3628659511587549e-13, 1.1343275878528474e-16],
 [8,   6.9160795579062393e-10, 2.0119792028960311e-17],
 [12,  2.999596419709301e-06,  1.7083267125104297e-17],
 [16,  0.017776866915952674,   6.8866641918714602e-17]],
```

(b) The algorithm appears not backward stable since the residual gets large as the condition number increases.

```
[[0,   5.6668973422827896e-16, 1.1959722892476256e-16],
 [4,   3.3270648768047658e-13, 2.9990752835841747e-14],
 [8,   5.5312036797396992e-10, 1.8815131322505332e-11],
```

```
 [12,  7.8752782311348258e-06, 2.1784856442503186e-07],
 [16,  2.3924681879507204,     0.0013630213010579159]],
```

(c) Note that for too large of m the determinant is not properly calculated.

The algorithm appears not backward stable since the residual gets large as the condition number increases.

```
[[0,   7.4032045694590438e-16, 1.7619297304857857e-16],
 [4,   1.0791108088755479e-13, 1.344502372418256e-14],
 [8,   5.7468386383603836e-10, 1.0701321230827662e-10],
 [12,  4.6049547088035518e-06, 1.0994510094470615e-06],
 [16,  0.029069352525193739,   0.0041372463048906759]]
```

Exercise 2

In Matlab, form a 60 by 60 matrix A with 1's on the main diagonal and in the last column, with -1's below the main diagonal, and with 0's everywhere else, as in (22.4) on p. 165 in the text. Compute the 2-norm condition number of A: `cond(A)`. Set a random vector x of length 60: `x = randn(60,1)`. Compute `b = A*x`.

(a) Solve the linear system $Ax = b$ using Gaussian elimination with partial pivoting by typing `x_ge = A\b`. Compute the 2-norm of the difference between the computed vector `x_ge` and the true solution `x` generated previously.

(b) Solve the linear system $Ax = b$ using the QR factorization of A: `[Q,R] = qr(A); x_qr = R\(Q'*b)`. Compute the 2-norm of the difference between the computed vector `x_qr` and the true solution `x`. Explain the difference in accuracy between the two computed solutions `x_ge` and `x_qr`.

(c) By hand, factor the 5 by 5 matrix in (22.4) on p. 165 using *complete* pivoting, so that $PAQ = LU$. What is the growth factor ρ in (22.2)? Would you expect to be able to solve a 60 by 60 linear system of this form to high relative accuracy (on a computer that satisfies the usual assumptions of IEEE arithmetic) using Gaussian elimination with complete pivoting? Explain why or why not.

Solution

(a,b) We implement this problem in python as,

```
def exercise_2():
    m=60
    A=np.tril(np.full((m,m),-1),-1)+np.identity(m)
    A[:,m-1]=np.full(m,1)

    x=np.random.randn(m,1)
    b=A@x

    x_ge = np.linalg.solve(A,b)

    [Q,R]=np.linalg.qr(A)
    x_qr = np.linalg.solve(R,Q.T@b)

    return [np.linalg.cond(A,2),np.linalg.norm(x-x_ge,2),np.linalg.norm
        (x-x_qr,2)]
```

This gives sample output,

```
[26.803535522538006, 16.8776468687335, 7.2434045018894103e-15]
```

Clearly using the QR factorization gives a far more accurate answer.

As explained in the book the growth factor of A is 2^{m-1}. As such, a huge amount of precision (roughy 60 bits) is lost. This means we are trying to calculate x_{ge} with only 4 bits of precision for some entries. If we examine the entires of $x_{ge} - x$ we find that they are on the order of 10^{-16} for the first entries, but on the order of 10^0 by the last entries. This aligns with the growth of entries in the LU factorization of A. A plot of the entries of $x - x_{ge}$ vs the index is shown in Figure 1. Clearly there is a exponential relationship between the error and the index, just as in the last row of U from the factorization of A.

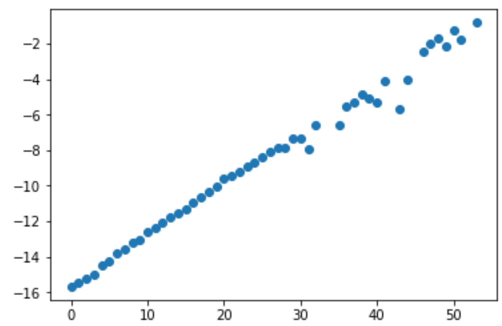

Figure 1: $\log((x - x_{ge})_i)$ vs. i for $i = 1, 2, ..., m$

However, QR factorization is backward stable, and so as expected we have a much lower error.

(c) We start with the 5×5 matrix below.

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

We do not require pivoting at the first step. We perform a row operation giving,

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & & & 1 & \\ 1 & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 0 & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{bmatrix}$$

We now pivot to move a "2" to the pivot position and apply another row operation,

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & -1 & & 1 & \\ & -1 & & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 0 & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & & & & 1 \\ & & 1 & & \\ & & & 1 & \\ & 1 & & & \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & -1 & 1 & -2 \\ 0 & 0 & -1 & -1 & -2 \end{bmatrix}$$

We pivot again to move a "-2" to the pivot position, and apply another row operation,

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -1 & 1 & \\ & & -1 & & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & -1 & 1 & -2 \\ 0 & 0 & -1 & -1 & -2 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & & & 1 \\ & & & 1 & \\ & & 1 & & \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 & 1 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & -1 & -2 \end{bmatrix}$$

We pivot again and apply another row operation,

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 & 1 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & -1 & -2 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & & 1 \\ & & & 1 & \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix}$$

Therefore,

$$U = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & & & & \\ & & & & 1 \\ & & 1 & & \\ & & & 1 & \\ & 1 & & & \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & & & 1 \\ & & & 1 & \\ & & 1 & & \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & & 1 \\ & & & 1 & \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ & 1 & & & \end{bmatrix}$$

$$\begin{aligned} L &= \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & & & 1 & \\ 1 & & & & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & -1 & & 1 & \\ & -1 & & & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -1 & 1 & \\ & & -1 & & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & -1 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & 1 & 1 & & \\ -1 & 1 & 1 & 1 & \\ -1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

We finally verify that,

$$AQ = LU$$

The growth factor is 2.

Based on the repeated structure of the steps after the first, it is clear that any size matrix of this form will have a similar $PAQ = LU$ decomposition. Therefore, for a larger matrix the growth factor would also be two.

Since the growth factor is constant, then we expect Gaussian elimination to be backward stable, and that our results be accurate.

Exercise 23.1

Let A be a nonsingular square matrix and let $A = QR$ and $A^*A = U^*U$ be QR and Cholesky factorizations, respectively, with the usual normalizations $r_{jj}, u_{jj} > 0$. Is it true or false that $R = U$.

Solution

We have,

$$A^*A = (QR)^*(QR) = R^*Q^*QR = R^*R$$

Since R is upper triangular and $r_{jj} > 0$ this is a Cholesky decomposition.

Obviously $A^*A = (A^*A)^*$ and for $u \neq 0$, $u^*(A^*A)u = (u^*A^*)(Au) = (Au)^*(Au) = \|Au\|^2 > 0$. Thus A^*A is Hermetian positive definite and therefore has a unique Cholesky decomposition. This proves $R = U$. □

Exercise 24.1

For each of the following statements, prove it is true or give an example to show it is false. Throughout, $A \in \mathbb{C}^{m\times m}$ unless otherwise indicated, and "ew" stands for eigenvalue.

(a) If λ is an ew of A and $\mu \in \mathbb{C}$, then $\lambda - \mu$ is an ew of $A - \mu I$.

(b) If A is real and λ is an ew of A, then so is $-\lambda$.

(c) If A is real and λ is an ew of A, then so is $\overline{\lambda}$.

(d) If λ is an ew of A and A is nonsingular, then λ^{-1} is an ew of A^{-1}.

(e) If all the ew's of A are zero, then $A = 0$.

(f) If A is hermetian and λ is an ew of A, then $|\lambda|$ is a singular value of A.

(g) If A is diagonalizable and all its ew's are equal, then A is diagonal.

Solution

We use the following equivalent statements:

- λ is an eigenvalue of A
- $\det(A - \lambda I) = 0$.
- λ is a root of $p_A(z) = \det(A - zI)$

These are mostly trivial proofs, so I do not restate the above equivalences in each problem to exactly match the wording of the problem statement.

(a) True. $\det((A - \mu I) - (\lambda - \mu)I) = \det(A - \mu I - \lambda I - (-\mu I)) = \det(A - \lambda I) = 0$

(b) False. Consider $A = [1] \in \mathbb{R}^{1\times 1}$. Clearly $\det(A - 1I) = 0$ but $\det(A - (-1)I) = 2 \neq 0$.

(c) True. If A is real then $p_A(z)$ has real coefficients. Therefore, by the fundamental theorem of algebra, if λ is a root of p_A, then so is $\overline{\lambda}$.

(d) True. $Av = \lambda v \iff \lambda^{-1}A^{-1}Av = \lambda^{-1}A^{-1}\lambda v \iff \lambda^{-1}v = \lambda^{-1}\lambda A^{-1}v \iff A^{-1}v = \lambda^{-1}v$

(e) False. Consider $A = [[0,1],[0,0]]$ with $p_A(z) = z^2$ so all eigenvalues are zero. However, clearly $A \neq 0$.

(f) True. A Hermetian matrix is unitary diagonalizable as $A = Q\Lambda Q^*$. Let S be the diagonal matrix with $S_{i,i} = \operatorname{sign}(\Lambda_{i,i})$. Then $A = (SQ)(S\Lambda)Q^* = (SQ)|\Lambda|Q^*$ is an SVD of A. This proves the singular values of A are the absolute values of the eigenvalues of A.

(g) True. All eigenvalues of A equal means $A = \lambda I$. If A is unitarily diagonalizable, then there is some D, diagonal, and Q, unitary, such that $D = QAQ^* = Q(\lambda I)Q^* = \lambda QQ^* = \lambda I = A$.

Exercise 24.2

Here is Gerschgorin's theorem, which holds for any $m \times m$ matrix A: Every eigenvalue of A lies in at least one of the m circular disks in the complex plane with centers a_{ii} and radii $\sum_{j \neq i} |a_{ij}|$. Moreover, if n of these disks form a connected domain that is disjoint from the other $m - n$ disks, then there are precisely n eigenvalues of A within this domain.

(c) Give estimates based on Gerschgorian's theorem for the eigenvalues of

$$A = \begin{bmatrix} 8 & 1 & 0 \\ 1 & 4 & \epsilon \\ 0 & \epsilon & 1 \end{bmatrix}, \qquad |\epsilon| < 1$$

(d) Find a way to establish the tighter bound $|\lambda_3 - 1| \leq \epsilon^2$ on the smallest eigenvalue of A.

Solution

(c) Let $\mathcal{D}(c, r) = \{z : |z - c| \leq r\}$ be the closed disk of radius r centered at c. Then there is exactly one eigenvalue in each of the following three disks as no two disks intersect.

$$\mathcal{D}(8, 1) \qquad \mathcal{D}(4, 1 + |\epsilon|) \qquad \mathcal{D}(1, |\epsilon|)$$

Since A is symmetric (and therefore Hermetian), we know all eigenvalues are real. We therefore take the part of the real axis contained in the above disks. This corresponds to the closed intervals,

$$[7, 9] \qquad [3 - |\epsilon|, 5 + |\epsilon|] \qquad [1 - |\epsilon|, 1 + |\epsilon|]$$

(d) Define,

$$Q = \begin{bmatrix} \epsilon^{-1} & & \\ & \epsilon^{-1} & \\ & & 1 \end{bmatrix} \qquad Q^{-1} = \begin{bmatrix} \epsilon & & \\ & \epsilon & \\ & & 1 \end{bmatrix}$$

Then,

$$QAQ^{-1} = \begin{bmatrix} 8 & 1 & \\ 1 & 4 & 1 \\ & \epsilon^2 & 1 \end{bmatrix}$$

Since QAQ^{-1} is a similarity transform of A, QAQ^{-1} and A share eigenvalues. In particular, this means the eigenvalues of QAQ^{-1} are real. Now note that since $|\epsilon| < 1$ the Gershgorin row disks are disjoint. Therefore, the smallest eigenvalue of QAQ^{-1} is in the interval $[1 - \epsilon^2, 1 + \epsilon^2]$.

This proves that the smallest eigenvector of A satisfies $|\lambda_3 - 1| \leq \epsilon^2$.

Exercise 6

By hand, find a Householder reflector Q and an upper Hessenberg matrix H such that $Q^*AQ = H$, where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Solution

ince A is 3×3 it takes only Householder reflector to take A to an upper Hessenberg matrix.

First let,

$$x = A_{2:3,1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Now let,

$$v = \operatorname{sign}(x_1) \left\| x \right\|_2 e_1 + x = 1\sqrt{2}e_1 + x = \begin{bmatrix} \sqrt{2} + 1 \\ 1 \end{bmatrix}$$

Then,

$$Q = \begin{bmatrix} I_1 & 0 \\ 0 & I_2 - 2\frac{vv^*}{v^*v} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 - \frac{2\left(\sqrt{2}+1\right)^2}{\left(\sqrt{2}+1\right)^2+1} & -\frac{2\left(\sqrt{2}+1\right)}{\left(\sqrt{2}+1\right)^2+1} \\ 0 & -\frac{2\left(\sqrt{2}+1\right)}{\left(\sqrt{2}+1\right)^2+1} & 1 - \frac{2}{\left(\sqrt{2}+1\right)^2+1} \end{bmatrix}$$

Finally,

$$H = Q^*AQ = \begin{bmatrix} 1 & -5/\sqrt{2} & 1/\sqrt{2} \\ -\sqrt{2} & 5/2 & -1/2 \\ 0 & 1/2 & -1/2 \end{bmatrix}$$