Sergio Garcia Tapia
Numerical Linear Algebra, Lloyd Trefethen and David Bau III
Lecture 9: MATLAB
June 27, 2024

Lecture 9: MATLAB

Exercise 1. (a) Run the six-line MATLAB program of Experiment 1 to produce a plot of approximate Legendre polynomials.

(b) For $k = 0, 1, 2, 3$, plot the difference on the on the 257-point grid between these approximations and the exact polynomials (7.11). How big are the errors, and how are they distributed?

(c) Compare these results with what you get with grid spacings $\Delta x = 2^{-\nu}$ for other values of ν. What power of Δx appears to control the convergence?

Solution:

(a) I used Python's Numpy and Matplotlib libraries to do the QR factorization and produce the plot ion Figure 1. See `./01-legendre-polynomials/plot_legendre.py`, whose code follows:

```
import numpy as np
from numpy import linalg
import matplotlib
import matplotlib.pyplot as plt


def approximate_legendre(k : int = 3, n : int = 7) -> np.array:
    """
        Uses a Vandermonde of size 2 ** n to approximate the Legendre
            polynomials
        through degree k on the interval [-1, 1]
    """
    if k < 0:
        raise ValueError('k must be a nonegative degree')
    if (n <= 0):
        raise ValueError('n must be a positive exponent')

    # Discretize interval [-1, 1]
    x = np.linspace(-1, 1, 2 ** n)

    # Construct Vandermonde matrix
    A = np.column_stack(tuple(x ** i for i in range(k + 1)))

    # Compute reduced QR factorization
```

```
    Q, R  = linalg.qr(A, mode='reduced')

    # Scale by last row
    scale_row = 1 / Q[-1, :]
    Q_scaled = Q @ np.diagflat(scale_row)

    # Free memory
    del scale_row, Q

    return Q_scaled

if __name__ == '__main__':
    # Compute approximate and exact legendre polynomials on [-1, 1]
    n = 8
    approx_legendre = approximate_legendre(n=n)

    # Compute exact Legend Polynomials
    x = np.linspace(-1, 1, 2 ** n)
    exact_legendre = np.column_stack((
        x ** 0,
        x ** 1,
        (3/2) * (x ** 2) - 1/2,
        (5/2) * (x ** 3) - (3/2) * (x ** 1)
    ))

    # Plot
    matplotlib.use('qtagg')
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle(f"Approximating Legendre Polynomials with Vandermonde
        Matrix, $n={n}$")

    # Plot approximations
    ax1.plot(approx_legendre)
    ax1.legend(('$k=0$', '$k=1$', '$k=2$', '$k=3$'))
    ax1.set_title("Discrete Legendre Approximation\nwith QR $[-1, 1]$")

    # Plot error in approximations
    ax2.plot(exact_legendre - approx_legendre)
    ax2.set_title("Error in Discrete Legendre\nApproximation with QR on
        $[-1, 1]$")
    ax2.legend(('$k=0$', '$k=1$', '$k=2$', '$k=3$'))

    plt.show()
```

(b) See Figure 2 for a plot of the error, which I've computed as the difference of the exact values of the Legendre Polynomials on the same grid points minus the values of the approximation. The error is distributed symmetrically about 0 for each polyno-

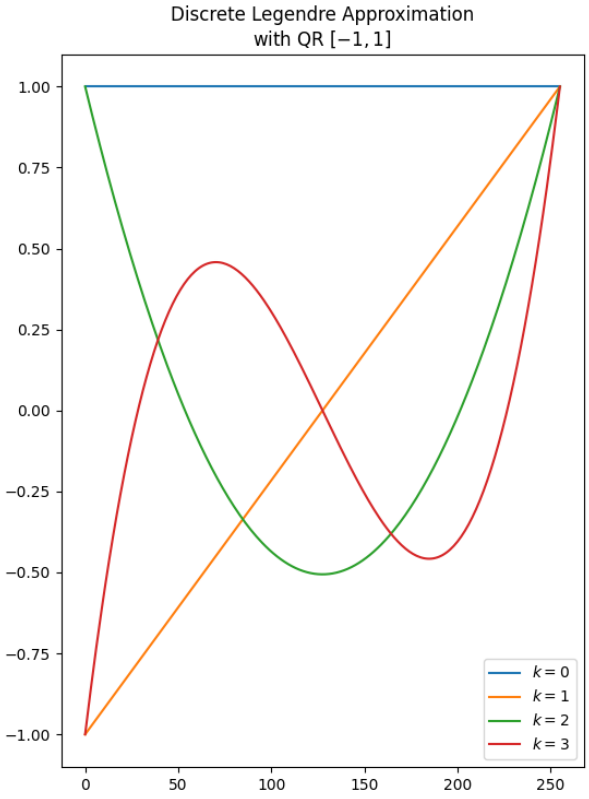

Figure 1: Exercise 9.1a: Discrete Legendre Polynomials Approximation for degrees 0, 1, 2, 3, on the Interval $[-1, 1]$ by finding the QR factorization of a Vandermonde matrix using 256 grid points.

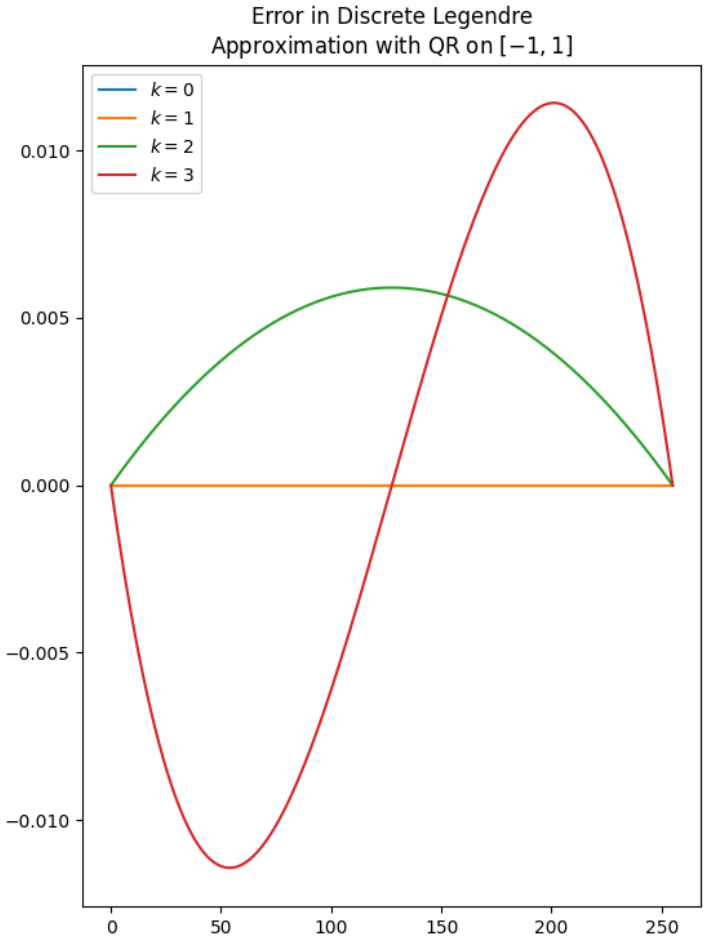

Figure 2: Exercise 9.1b: Plot of error in approximation to Legendre Polynomials given in Figure 1, using 256 grid points. Error is computed as the $P_{k,exact}(x) - P_{k,approx}$.

mial. Let P_k denote the kth degree Legendre polynomial. The polynomials P_0 and P_1 match their approximations exactly. The errors in P_2 and P_3 show a symmetric graph. The error is greater away from their endpoints and near the relative extrema of the functions.

(c) I observed that an increase in the power of Δx by 1 resulted in halving the maximum error across all polynomials.

Exercise 2. In Experiment 2, the singular values of A match the diagonal elements of a QR factor R approximately. Consider now a very different example. Suppose $Q = I$ and $A = R$, the $m \times m$ (a *Toeplitz matrix*) with 1 on the main diagonal, 2 on the first superdiagonal, and 0 everywhere else.

(a) What are the eigenvalues, determinant, and rank of A?

(b) What is A^{-1}?

(c) Give a nontrivial upper bound on σ_m, the mth singular value for A. You are welcome to use MATLAB for inspiration, but the bound you give should be justified analytically. (*Hint*: Use part (b).)

This problem illustrates that you cannot always infer much about the singular values of a matrix form its eigenvalues or from the diagonal entries of a QR factor R

Solution:

(a) The *superdiagonal* of a matrix is the set of elements above the elements comprising the diagonal (source: Wolfram). Thus the matrix in question looks like:

$$A = R = \begin{bmatrix} 1 & 2 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 2 & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 2 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

Since this matrix is upper-triangular, its eigenvalues are the values of its diagonal entries, all of which are 1. The determinant is the product of the diagonal entries, which is also 1. Finally, the rank of A is m, because all of its eigenvalues are nonzero.

(b) The inverse is:

$$A^{-1} = \begin{bmatrix} 1 & -2 & 4 & -8 & \cdots \\ 0 & 1 & -2 & 4 & \cdots \\ \vdots & 0 & 1 & -2 & \cdots \\ \vdots & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & 0 & \ddots \end{bmatrix}$$

In other words, the inverse has 1s on the diagonal, -2 on the first super diagonal, 4 on the next superdiagonal, -8 on the next superdiagonal, and so on.

(c) Note that if $A = U\Sigma V^*$ is a singular value decomposition of A, then $A^{-1} = V\Sigma^{-1}U^*$ is a singular value decomposition of A^{-1}. In other words, σ_j is a nonzero singular value of A if and only if $\frac{1}{\sigma_j}$ is a singular value of A^{-1}.

Since $A \in \mathbb{C}^{m\times m}$ is invertible, all of its singular values are nonzero, so $\sigma_j > 0$ for $j \in \{1, \ldots, m\}$. The singular values $\sigma_1, \ldots, \sigma_m$ of A are conventionally listed in descending order, meaning that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$. Thus, σ_1 is the value of the largest singular value of A and σ_m is the value of the smallest singular value of A.

By contrast, $\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_m}$ are the singular values of A^{-1}, but this list is given in ascending order. In other words, the largest singular value of A^{-1} is $\frac{1}{\sigma_m}$. By Theorem 5.3,

$\|A\|_2 = \sigma_1$, where $\|\cdot\|_2$ is vector-induced matrix 2-norm, and σ_1 is the largest singular value of A. Thus, applying the same Theorem to A^{-1}, we conclude that $\|A^{-1}\|_2 = \frac{1}{\sigma_m}$.

Recall from Exercise 3.3(d) that $\|A^{-1}\|_\infty \leq \sqrt{m}\ \|A^{-1}\|_2$, where $\|\cdot\|_\infty$ is the vector-induced matrix ∞-norm. Let $(a_k^{-1})^*$ denote the k-th row of A^{-1}, and recall also from Example 3.4 that $\|A^{-1}\|_\infty = \max_{1\leq k\leq m}\|(a_k^{-1})^*\|_1$, where $\|\cdot\|_1$ is the vector 1-norm. Clearly the first row of A^{-1} has the largest 1-norm. Then

$$\begin{aligned}
\frac{1}{\sigma_m^2} &= \|A^{-1}\|_2{}^2 \\
&\geq \frac{1}{m}\left\|A^{-1}\right\|_\infty{}^2 \\
&= \frac{1}{m}\cdot\max_{1\leq k\leq m}\|(a_k^{-1})^*\|_1 \\
&= \frac{1}{m}\cdot\|(a_1^{-1})^*\|_1 \\
&= \frac{1}{m}\left(1^2 + |-2| + |4|^2 + \cdot + |(-2)^{m-1}|\right)^2 \\
&= \frac{1}{m}\left(\cdot\sum_{j=0}^{m-1} 2^j\right)^2 \\
&= \frac{1}{m}(2^m-1)^2
\end{aligned}$$

Rearranging and taking square roots on both sides, we get $\frac{1}{m}(2^m-1) \geq \sigma_m$, and hence $\frac{1}{m}2^m$ is an upper bound of σ_m.

Exercise 3. (a) Write a MATLAB program that sets up a 15×40 matrix with entries 0 everywhere except for the values 1 in the positions indicated in the picture below. The upper-leftmost 1 is in position $(2, 2)$, and the lower-rightmost 1 is in position $(13, 39)$. This picture was produced with the command `spy(A)`

(b) Call `svd` to compute the singular values of A, and print the results. Plot these numbers using both `plot` and `semilogy`. What is the mathematically exact rank of A? How does this show up in the computed singular values?

(c) For each i from 1 to rank(A), construct the rank-i matrix B that is the best approximation to A in the 2-norm. Use the command `pcolor(B)` with `colormap(gray)` to create images of these various approximations?