MA 580 Assignment 2

Kyle Hansen

26 September 2025

AI Use Statement: Here is where I put the AI use statemet

Question 1

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. Prove the following. In your solution use only the materials discussed in the lectures so far and results from the previous homework.

1(a) Show that $\lambda \neq 0$ is an eigenvalue of $\mathbf{A}^T\mathbf{A}$ if and only if it is an eigenvalue of $\mathbf{A}\mathbf{A}^T$.

The nozero eigenvalues of $\mathbf{A}^T\mathbf{A}$ are defined such that, for some $x \neq 0$,

$$(\mathbf{A}^T\mathbf{A})x = \lambda x \tag{1}$$

then it can be shown that this value λ is also an eigenvalue of $\mathbf{A}\mathbf{A}^T$:

$$\begin{aligned} \mathbf{A}^T\mathbf{A}x &= \lambda x \\ \mathbf{A}\mathbf{A}^T\mathbf{A}x &= \lambda \mathbf{A}x \end{aligned} \tag{2}$$

Then, for some $b = \mathbf{A}x$,

$$(\mathbf{A}\mathbf{A}^T)b = \lambda b \tag{3}$$

That is, all nonzero eigenvalues of $\mathbf{A}^T\mathbf{A}$ are also eigenvalues of $\mathbf{A}\mathbf{A}^T$.

The inverse can be shown similarly:

$$\begin{aligned} \mathbf{A}\mathbf{A}^T x &= \lambda x \\ \mathbf{A}^T\mathbf{A}\mathbf{A}^T x &= \lambda \mathbf{A}^T x \end{aligned} \tag{4}$$

Then for some $z = \mathbf{A}^T x$, $\mathbf{A}^T\mathbf{A}z = \lambda z$. Then the matrices $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ share all non-zero eigenvalues.

1(b) Show $\left\|\mathbf{A}^T\right\|_2 = \|\mathbf{A}\|_2$.

This can be shown using the spectral radius, since $\|\mathbf{A}\|_2 = \rho(\mathbf{A}^T\mathbf{A})^{1/2}$, which implies $\left\|\mathbf{A}^T\right\|_2 = \rho(\mathbf{A}\mathbf{A}^T)^{1/2}$.

Since $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ have the same nonzero eigenvalues, then they share the same spectral radius $\rho(\mathbf{A}) = \max|\lambda|$. Then:

$$\begin{aligned} \rho(\mathbf{A}^T\mathbf{A})^{1/2} &= \rho(\mathbf{A}\mathbf{A}^T)^{1/2} \\ \|\mathbf{A}\|_2 &= \left\|\mathbf{A}^T\right\|_2 . \end{aligned} \tag{5}$$

1(c) Show $\left\|\mathbf{A}\mathbf{A}^T\right\|_2 = \left\|\mathbf{A}^T\mathbf{A}\right\|_2 = \|\mathbf{A}\|_2^2$.

Again using the spectral radius, these norms can be rewritten in terms of the largest eigenvalue of the two products:

$$\begin{aligned} \left\|\mathbf{A}^T\mathbf{A}\right\| &= \rho\left(\left(\mathbf{A}^T\mathbf{A}\right)^2\right)^{1/2} \\ \left\|\mathbf{A}\mathbf{A}^T\right\| &= \rho\left(\left(\mathbf{A}\mathbf{A}^T\right)^2\right)^{1/2}, \end{aligned} \tag{6}$$

where $\left(\mathbf{A}^T\mathbf{A}\right)^T = \mathbf{A}^T\mathbf{A}$ and $\left(\mathbf{A}\mathbf{A}^T\right)^T = \mathbf{A}\mathbf{A}^T$.

Since eigenvalues of a squared matrix are the squared eigenvalues of the original matrix,

$$\begin{aligned} \rho\left(\left(\mathbf{A}^T\mathbf{A}\right)^2\right)^{1/2} &= \rho\left(\mathbf{A}^T\mathbf{A}\right) \\ \rho\left(\left(\mathbf{A}\mathbf{A}^T\right)^2\right)^{1/2} &= \rho\left(\mathbf{A}\mathbf{A}^T\right). \end{aligned} \tag{7}$$

These terms are equal to the squares of Equation 5 from the previous subsection– since $\rho\left(\mathbf{A}\mathbf{A}^T\right)^{1/2} = \rho\left(\mathbf{A}^T\mathbf{A}\right)^{1/2} = \|\mathbf{A}\|_2$, the same is true of their squares, and

$$\left\|\mathbf{A}\mathbf{A}^T\right\|_2 = \left\|\mathbf{A}^T\mathbf{A}\right\|_2 = \|\mathbf{A}\|_2^2 \tag{8}$$

1(d) Suppose $\mathbf{A} \in \mathbb{R}^{n\times n}$ is nonsingular. Show $\kappa_2(\mathbf{A}^T\mathbf{A}) = \kappa_2(\mathbf{A})^2$.

The condition number of of the product is defined as

$$\kappa_2(\mathbf{A}^T\mathbf{A}) = \left\|\mathbf{A}^T\mathbf{A}\right\|_2 \left\|\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\right\|_2 . \tag{9}$$

From the previous section, $\left\|\mathbf{A}\mathbf{A}^T\right\|_2 = \left\|\mathbf{A}^T\mathbf{A}\right\|_2 = \|\mathbf{A}\|_2^2$, so

$$\kappa_2(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_2^2 \left\|\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\right\|_2. \tag{10}$$

Where the inverse term can be expanded as $\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \mathbf{A}^{-1}(\mathbf{A}^T)^{-1}$. Since the inverse of a transpose is the transpose of an inverse, this can be rewritten $\mathbf{A}^{-1}(\mathbf{A}^{-1})^T$. The same property as above can be applied again, since $\left\|\mathbf{A}^{-1}(\mathbf{A}^{-1})^T\right\|_2 = \left\|(\mathbf{A}^{-1})^T\mathbf{A}^{-1}\right\|_2 = \|\mathbf{A}^{-1}\|_2^2$, and

$$\kappa_2(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_2^2 \left\|\mathbf{A}^{-1}\right\|_2^2 = \left(\|\mathbf{A}\|_2 \left\|\mathbf{A}^{-1}\right\|_2\right) = \kappa_2(\mathbf{A}^2) \tag{11}$$

Question 2

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be nonsingular, and suppose $\mathbf{E} \in \mathbb{R}^{n\times n}$ satisfies $\|\mathbf{E}\| \leq \frac{1}{2\|\mathbf{A}^{-1}\|}$, where $\|\cdot\|$ is an induced matrix norm. Then, $\mathbf{A}+\mathbf{E}$ is nonsingular, and

$$\left\|(\mathbf{A}+\mathbf{E})^{-1}\right\| \leq 2\left\|\mathbf{A}^{-1}\right\|.$$

Since $\mathbf{A}$ is nonsingular, the addition can be rewritten

$$\mathbf{A}+\mathbf{E} = \mathbf{A} + \mathbf{A}\mathbf{A}^{-1}\mathbf{E} = \mathbf{A}\left(\mathbf{I} + \mathbf{A}^{-1}\mathbf{E}\right) \tag{12}$$

The condition $\|\mathbf{E}\| \leq 1/2\left\|\mathbf{A}^{-1}\right\|$ can be rewritten (using the Cauchy-Schwarz inequality)

$$\begin{gathered}\|\mathbf{E}\| \leq \frac{1}{2\left\|\mathbf{A}^{-1}\right\|} \\ 2\left\|\mathbf{A}^{-1}\mathbf{E}\right\| \leq 2\left\|\mathbf{A}^{-1}\right\|\|\mathbf{E}\| \leq 1 \\ \left\|\mathbf{A}^{-1}\mathbf{E}\right\| \leq 1/2\end{gathered} \tag{13}$$

Then since $\|\mathbf{A}^{-1}\mathbf{E}\| \leq 1/2$, Fact 2.23 in [1] can be used to show that $(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E})$ is nonsingular and

$$\left\|\left(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E}\right)^{-1}\right\| \leq 2. \tag{14}$$

Since both $\mathbf{A}$ and $(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E})$ are nonsingular and the product of two nonsingular matrices is also nonsingular, then the product $(\mathbf{A}+\mathbf{E}) = \mathbf{A}\left(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E}\right)$ is nonsingular.

Again using the Cauchy-Schwarz inequality and the result from Equation 14,

$$\left\|(\mathbf{A}+\mathbf{E})^{-1}\right\| = \left\|\mathbf{A}^{-1}\left(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E}\right)^{-1}\right\| \leq \left\|\mathbf{A}^{-1}\right\|\left\|\left(\mathbf{I}+\mathbf{A}^{-1}\mathbf{E}\right)^{-1}\right\| \leq 2\left\|\mathbf{A}^{-1}\right\|. \tag{15}$$

Question 3

Let $\|\cdot\|$ be a vector norm. Let $\mathbf{A}$ be a nonsingular matrix. Consider the maximum and minimum magnification, $\mathbf{maxmag}(\mathbf{A})$ and $\mathbf{minmag}(\mathbf{A})$.

Show that

$$\kappa(\mathbf{A}) = \frac{\mathbf{maxmag}(\mathbf{A})}{\mathbf{minmag}(\mathbf{A})}.$$

Begining from the definitions

$$\begin{aligned} \mathbf{minmag}(\mathbf{A}) &= \min \frac{\|\mathbf{A}x\|}{\|x\|} \\ \mathbf{maxmag}(\mathbf{A}) &= \max \frac{\|\mathbf{A}x\|}{\|x\|} \end{aligned} \tag{16}$$

The **minmag** can be redefined using the property that that inverse of a maximum is the maximum of an inverse:

$$\mathbf{minmag}(\mathbf{A}) = \min \frac{\|\mathbf{A}x\|}{\|x\|} = \min \frac{\|b\|}{\|\mathbf{A}^{-1}b\|} = \frac{1}{\max \frac{\|\mathbf{A}^{-1}b\|}{\|b\|}} = \frac{1}{\|\mathbf{A}^{-1}\|} \tag{17}$$

Then, since $\mathbf{maxmag}(\mathbf{A}) = \|\mathbf{A}\|$ (by the interpretation discussed in class), $\mathbf{minmag}(\mathbf{A}) = \|\mathbf{A^{-1}}\|^{-1}$ (by Equation 17), and $\kappa(\mathbf{A}) = \|\mathbf{A}\|\,\|\mathbf{A}^{-1}\|$, the condition can be rewritten:

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\| = \|\mathbf{A}\|\left(\frac{1}{\|\mathbf{A}^{-1}\|}\right)^{-1} = \frac{\mathbf{maxmag}(\mathbf{A})}{\mathbf{minmag}(\mathbf{A})}. \tag{18}$$

Question 4

Computer problem, Python

Computer problem. Consider the following elliptic partial differential equation (PDE).

$$\begin{aligned} -\Delta u(x,y) &= f(x,y) \qquad \text{in } \Omega = (0,1)\times(0,1), \\ u &= 0, \qquad\qquad\quad \text{on } \partial\Omega. \end{aligned} \tag{19}$$

4(a) Develop a routine that given the right-hand-side function f and the number n, returns the coefficient matrix and right-hand side vector for the linear system resulting from the finite-difference discretization of the problem, using sparse matrix storage.

I used the same sample code provided in the appendix, adapted to function in Python. Since the 2D finite difference scheme is given by

$$\Delta u \approx 4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} \tag{20}$$

Since $1/h^2$ appears in every entry of the matrix, it can be factored out for simplicity (now solving instead the system $h^2\mathbf{A}\boldsymbol{x} = h^2\boldsymbol{b}$.

The matrix $\mathbf{A}$ should then have 4 in every diagonal entry, and using the scheme $k = i + (j-1)(n-1)$ to convert the 2D coordinate to an index, -1 should appear one entry to the left and right of the diagonal (corresponding with $i \pm 1$) and, as well as $(n-1)$ entries to the left and right (corresponding with $j \pm 1$). The boundary conditions mean that some rows appear without all 5 coefficients– the matrix is only $(n-1) \times (n-1)$.

The vector b is simply set to $b_k = h^2 u_k$, where $u_k = u_{i,j}$ using the flatting algorithm described in the assignment's appendix.

The matrix and vector assignment appear in the `system_def` function in the Python code.

4(b) Solve the problem with

$$f(x,y) = -2\pi^2(\cos^2(\pi x)\sin^2(2\pi y) - 5\sin^2(\pi x)\sin^2(2\pi y) + 4\sin^2(\pi x)\cos^2(2\pi y)), \tag{21}$$

and with $n = 2^k, k = 3, \ldots, 10$ using Gaussian elimination. Note that for this choice of f the analytic solution is given by,

$$u(x,y) = \sin^2(\pi x)\sin^2(2\pi y).$$

The code is presented in Appendix A. As an example, a numerical solution is shown in Figure 1 and its absolute error is shown in Figure 2.

The results for all run cases are presented in Table 1. The results clearly demonstrate a logarithmic convergence rate, where each doubling of n (or quadrupling of n^2) results in a reduction in error by one quarter. Because of this convergence rate, a linear increase in the number of meshpoints (which results in significantly increased computational cost) has a diminishing effect on error reduction.

4(c) Scipy's `linalg.onenormest` could be used to estimate the one norm $\|\mathbf{A}\|_1$, but does not directly compute the condition number, which requires $\|\mathbf{A}^{-1}\|_1$, or an appropriate estimate.

Matlab provides the function `condest`, which uses an iterative method [2] that computes the gradient of $f(\boldsymbol{x}) = \|\mathbf{A}\boldsymbol{x}\|_1$, in order to converge on an upper bound for $\|\mathbf{A}^{-1}\|_1$. When I tested my own implementation of this algorithm I found its results to be equivalent to directly computing the matrix inverse and using `onenormest`. This iterative estimate appears as `hager_invnorm` in the Python code.

Table 1: Convergence results for two-dimensional Poisson problem

h	no. of unknowns	$\|e_h\|_\infty$	$\|e_{2h}\|_\infty \,/\, \|e_h\|_\infty$
0.125	8^2	$1.72 \cdot 10^{-1}$	0.2298
0.0625	16^2	$3.96 \cdot 10^{-2}$	0.2449
0.03125	32^2	$9.70 \cdot 10^{-3}$	0.2487
0.01563	64^2	$2.41 \cdot 10^{-3}$	0.2499
0.00781	128^2	$6.03 \cdot 10^{-3}$	0.2500
0.00391	256^2	$1.51 \cdot 10^{-4}$	0.2500
0.00195	512^2	$3.77 \cdot 10^{-4}$	0.2500
0.00098	1024^2	$9.42 \cdot 10^{-5}$	–

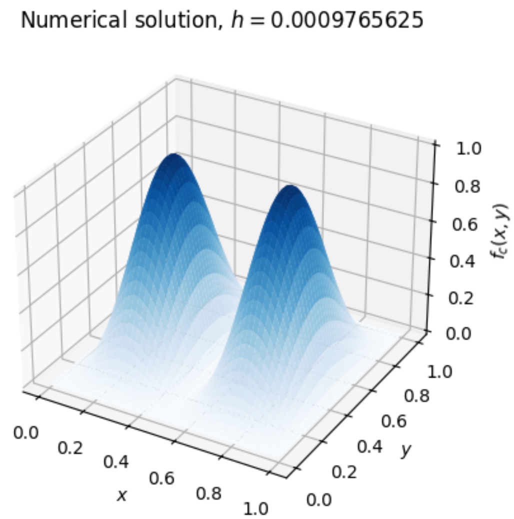

Figure 1: Numerical solution, $n = 2^{10}$

The condition numbers for n up to 2^{10} are shown in Figure 3. κ shows a clear $\kappa \propto n^{\approx 2}$ increase, clearly justifying the need for a more stable algorithm, potentially different than the GE scheme used in this solve.

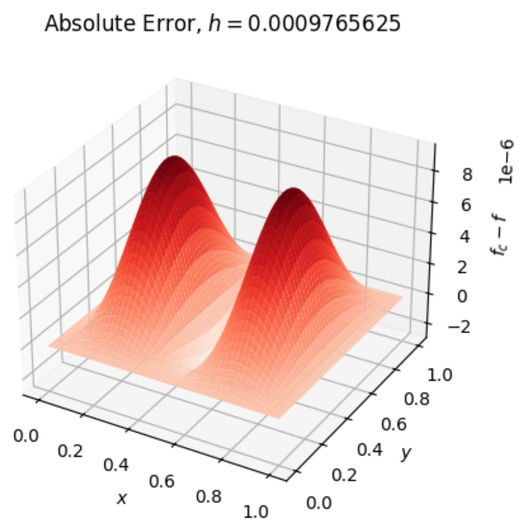

Figure 2: Absolute error, $n = 2^{10}$

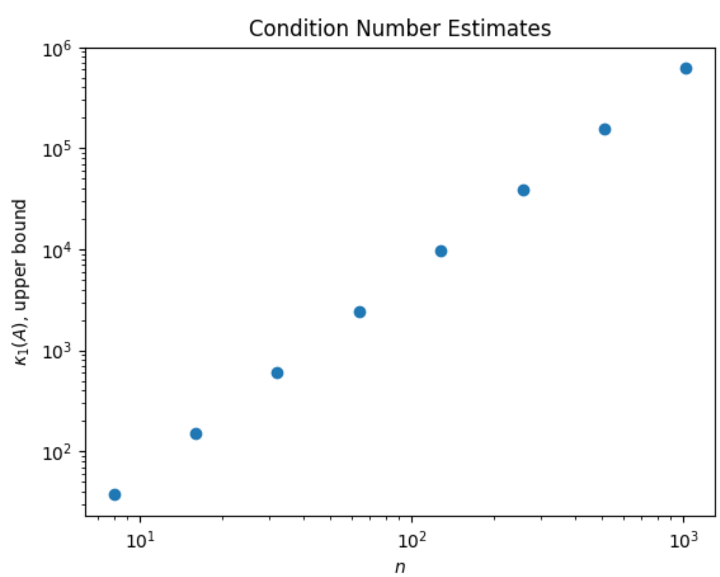

Figure 3: Condition number estimates, $\kappa_1(\mathbf{A})$

References

[1] I. C. Ipsen, *Numerical matrix analysis: Linear systems and least squares*. SIAM, 2009.

[2] W. W. Hager, "Condition estimates," *SIAM Journal on scientific and statistical computing*, vol. 5, no. 2, pp. 311–316, 1984.

Question A Python Code

```
import numpy
# import scipy
import scipy.sparse as sparse
import matplotlib
import matplotlib.pyplot as plt

pi = numpy.pi
# part a -- matrix assignment
def source_function(x, y):
    z = -2*(pi**2)*((numpy.cos(pi*x)**2 * numpy.sin(2*pi*y)**2)
                    -5*( numpy.sin(pi*x)**2 * numpy.sin(2*pi*y)**2)
                    +4*(numpy.sin(pi*x)**2 * numpy.cos(2*pi*y)**2))
    return z

def analytic_solution(x, y):
    return numpy.sin(pi*x)**2 * numpy.sin(2*pi*y)**2

class poisson_system:
    def __init__(self, x, y, A, f):
        self.x_mesh = x
        self.y_mesh = y
        self.coefficients = A
        self.source = f

def unpack_1d_array(a):
    m = int(numpy.sqrt(a.size))
    n = m+1
    A = numpy.zeros((m,m))
    for i in range(0, m):
        for j in range(0, m):
            A[i][j] = a[i+ (j)*(m)]
    return A

def system_def(f, n):
    m = n-1
    I = sparse.eye(m,m)
```

```
    e = numpy.ones(m)
    T = sparse.diags([-e, 4*e, -e], [-1, 0, 1], (m,m))
    S = sparse.diags([-e, -e], [-1, 1], (m,m))
    A = sparse.kron(I, T) + sparse.kron(S, I)

    h = 1/n
    x = numpy.arange(h, 1, h)
    y = numpy.arange(h, 1, h)
    q_vec = numpy.zeros((n-1)**2)

    for i in range(0, (n-1)):
        for j in range(0, (n-1)):
            k = i+ (j)*(n-1)
            q_vec[k] = (h**2)*source_function(x[i], y[j])

    return poisson_system(x, y, A, q_vec)

def hager_invnorm(A):
    # Hager 1984, Higham 1988
    n = A.shape[0]
    x = (1/n)*numpy.ones((n))
    while (True):
        y = sparse.linalg.spsolve(A, x)
        # heaviside instead of sign() so that sign(0) = 1
        xi = 2*numpy.heaviside(y, 1) - 1
        z = sparse.linalg.spsolve(A,xi)
        if (numpy.max(abs(z)) <= z@x):
            return numpy.sum(abs(y))
        x = numpy.zeros(n)
        x[numpy.argmax(z)] = 1


k = numpy.arange(3, 11, 1)
n = 2**k
h = 1/n
e = numpy.zeros(n.shape)
r = e.copy()
cond = e.copy()

for ni in range(0, n.size):
    print("n= "+str(n[ni]))
    system = system_def(source_function, n[ni])
    A = system.coefficients
    x = system.x_mesh
    y = system.y_mesh
```

```
    q = system.source

    u = sparse.linalg.spsolve(A, q)
    U = unpack_1d_array(u)

    K = numpy.zeros(U.shape)
    for i in range(0, K.shape[0]):
        for j in range(0, K.shape[1]):
            K[i][j] = analytic_solution(x[i], y[j])

    e[ni] = numpy.max(numpy.abs(K-U))
    cond[ni] = sparse.linalg.onenormest(A)*hager_invnorm(A)
    print(cond[ni])
    if (ni>0):
        r[ni] = e[ni]/e[ni-1]
    if n[ni] == 2**10:
        x, y = numpy.meshgrid(x, y)

        fig = plt.figure()
        ax = plt.axes(projection="3d")
        ax.plot_surface(x, y,(U-K), cmap=matplotlib.cm.Reds)
        plt.title("Absolute Error, $h="+str(h[ni])+"$")
        ax.set_xlabel("$x$")
        ax.set_ylabel("$y$")
        ax.set_zlabel("$f_c - f$")
        plt.show()
        # plt.savefig("abs_errs")
        plt.close()

        fig = plt.figure()
        ax = plt.axes(projection="3d")
        ax.plot_surface(x, y,U, cmap=matplotlib.cm.Blues)
        plt.title("Numerical solution, $h="+str(h[ni])+"$")
        ax.set_xlabel("$x$")
        ax.set_ylabel("$y$")
        ax.set_zlabel("$f_c(x,y)$")
        plt.show()
        # plt.savefig("num_soln")
        plt.close()

fig = plt.figure()
plt.scatter(n, cond)
plt.title("Condition Number Estimates")
plt.xlabel("$n$")
plt.ylabel("$\\kappa_1(A)$, upper bound")
```

```python
plt.show()
# plt.savefig("condition_linear")
plt.xscale("log")
plt.yscale("log")
plt.show()
# plt.savefig("condition_log")
plt.close()

```