STAT 154/254 Homework 1

Raymond Tsao

TOTAL POINTS

33 / 33

QUESTION 1

1 P1 **3 / 3**

✓ **- 0 pts** *Correct*

- 0.3 pts Not all derivations were shown in reducing the matrix.

- 0.5 pts Didn't provide evidence as to why vector e could not be written as a linear combination of only two vectors.

- 1.5 pts Partial credit for Q1

- 3 pts No work shown for Q1

QUESTION 2

P2 5 pts

2.1 2.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.3 pts Slight error in derivation

- 0.3 pts Correct, but the answer can be simplified further.

- 0.4 pts Said Beta(778,224), but did not address that 0.1 <= p <= 0.9 can be extended to 0 <= p <= 1.

- 1 pts Error in derivation

- 3 pts No work shown for Q2.1

2.2 2.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Error in derivation/formulation

- 1 pts Partial credit for Q2.2

- 2 pts No work shown for Q2.2

QUESTION 3

3 P3 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 0.75 pts Didn't explain how the result can be used to generate a sample of random variables from a given distribution.

- 2 pts Partial credit for Q3

- 4 pts No work shown for Q3

QUESTION 4

P4 6 pts

4.1 4.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q4.1

- 2 pts No work shown for Q4.1

4.2 4.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.3 pts Correct marginal densities, but incorrect conditional density

- 0.3 pts Slight error in calculating marginal densities

- 0.4 pts Error in calculating marginal densities

- 1 pts Partial credit for Q4.2

- 2 pts No work shown for Q4.2

4.3 4.3 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct about x_1 and x_2 being stochastically dependent, but said x_1 and x_2 are correlated.

- 1 pts Partial credit for Q4.3

- 2 pts `No work shown for Q4.3

QUESTION 5

5 P5 **0 / 0**

✓ **- 0 pts** *Optional*

QUESTION 6

6 P6 **5 / 5**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Correct approach, but more work is needed to support your answer or key error in derivation.

- 2.5 pts Partial credit for Q6

- 5 pts No work shown for Q6

QUESTION 7

P7 10 pts

7.1 7.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q7.1

- 2 pts No work shown for Q7.1

7.2 7.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q7.2

- 2 pts No work shown for Q7.2

7.3 7.3 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts More explanation is needed to support your answer.

- 1 pts Partial credit for Q7.3

- 2 pts No work shown for Q7.3

7.4 7.4 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Not all histograms were plotted

- 1 pts Partial credit for Q7.4

- 2 pts No work shown for Q7.4

7.5 7.5 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Not all shapes computed by the Marchenko-Pastur Law are plotted

- 1 pts Partial credit for Q7.5

- 2 pts No work shown for Q7.5

Problem 1

We solve for

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Computing the determinant gives us

$$\begin{vmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{vmatrix} = 1 \cdot \begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix} - 2 \cdot \begin{vmatrix} 3 & 2 \\ 2 & 1 \end{vmatrix} + 3 \cdot \begin{vmatrix} 3 & 1 \\ 2 & 3 \end{vmatrix} = -5 + 2 + 21 = 18$$

The inverse is hence given by

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}^{-1} = \frac{1}{18} \begin{bmatrix} -5 & 7 & 1 \\ 1 & -5 & 7 \\ 7 & 1 & -5 \end{bmatrix} \implies \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{7}{18} \\ -\frac{5}{18} \\ \frac{1}{18} \end{bmatrix}$$

Hence

$$\frac{7}{18} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} - \frac{5}{18} \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} + \frac{1}{18} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Since the determinant is non-zero, the columns are linearly independent, meaning that we cannot write it as a sum of 2 vectors.

1 P1 3 / 3

✓ **- 0 pts** *Correct*

- 0.3 pts Not all derivations were shown in reducing the matrix.

- 0.5 pts Didn't provide evidence as to why vector e could not be written as a linear combination of only two vectors.

- 1.5 pts Partial credit for Q1

- 3 pts No work shown for Q1

Problem 2

1.

The posterior is a truncated beta distribution.
Suppose $p \sim \text{Uniform}[\xi_1, \xi_2]$, then

$$\begin{aligned} f(p|x) &\propto f(x|p)f(p) \\ &\propto \binom{n}{x} p^x (1-p)^{n-x} \frac{\mathbb{1}\{\xi_1 \le p \le \xi_2\}}{0.8} \\ &= \frac{\binom{n}{x} p^x (1-p)^{n-x} \mathbb{1}\{\xi_1 \le p \le \xi_2\}}{\int_{\xi_1}^{\xi_2} \binom{n}{x} p^x (1-p)^{n-x} dp} \end{aligned}$$

Plugging in $n = 1000, x = 777, \xi_1 = 0.1, \xi_2 = 0.9$, we have

$$f(p|x) = \frac{p^{777}(1-p)^{223} \mathbb{1}\{0.1 \le p \le 0.9\}}{\int_{0.1}^{0.9} p^{777}(1-p)^{223} dp}$$

Since f is a truncated beta, its mode is given by

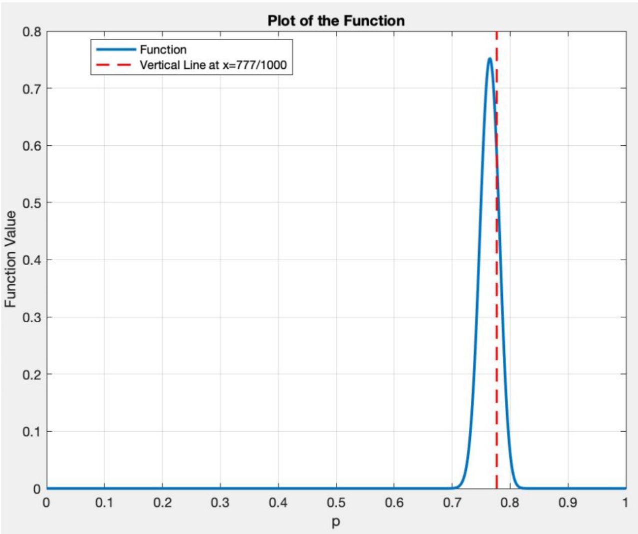
$$\hat{p}_{MAP} = \frac{777}{1000}$$

2.

If we instead only know that $X \in [750, 780]$, then

$$\begin{aligned} f(p|x \in [750, 780]) &\propto f(x \in [750, 780]|p)f(p) \\ &\propto \left[\sum_{k=750}^{780} \binom{1000}{k} p^k (1-p)^{1000-k} \right] \mathbb{1}\{0.1 \le p \le 0.9\} \end{aligned}$$

We can solve for the maximum by plotting the function on Matlab:



Note that the estimated p value becomes smaller.

2.1 2.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.3 pts Slight error in derivation

- 0.3 pts Correct, but the answer can be simplified further.

- 0.4 pts Said Beta(778,224), but did not address that $0.1 <= p <= 0.9$ can be extended to $0 <= p <= 1$.

- 1 pts Error in derivation

- 3 pts No work shown for Q2.1

Problem 2

1.

The posterior is a truncated beta distribution.
Suppose $p \sim \text{Uniform}[\xi_1, \xi_2]$, then

$$\begin{aligned} f(p|x) &\propto f(x|p)f(p) \\ &\propto \binom{n}{x} p^x (1-p)^{n-x} \frac{\mathbb{1}\{\xi_1 \leq p \leq \xi_2\}}{0.8} \\ &= \frac{\binom{n}{x} p^x (1-p)^{n-x} \mathbb{1}\{\xi_1 \leq p \leq \xi_2\}}{\int_{\xi_1}^{\xi_2} \binom{n}{x} p^x (1-p)^{n-x} dp} \end{aligned}$$

Plugging in $n = 1000, x = 777, \xi_1 = 0.1, \xi_2 = 0.9$, we have

$$f(p|x) = \frac{p^{777}(1-p)^{223} \mathbb{1}\{0.1 \leq p \leq 0.9\}}{\int_{0.1}^{0.9} p^{777}(1-p)^{223} dp}$$

Since f is a truncated beta, its mode is given by

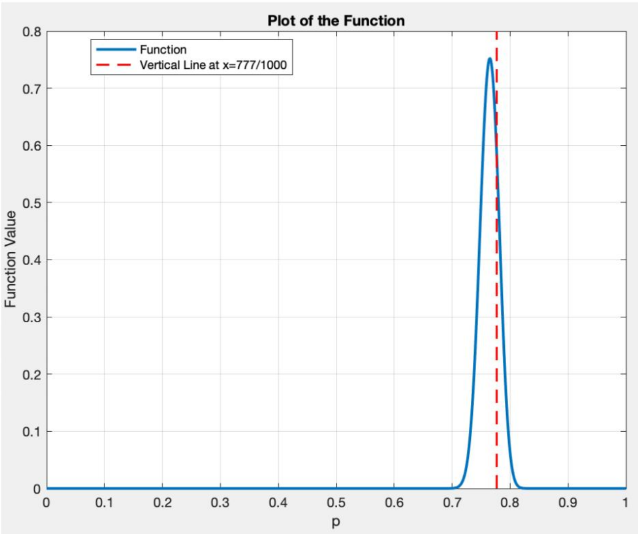
$$\hat{p}_{MAP} = \frac{777}{1000}$$

2.

If we instead only know that $X \in [750, 780]$, then

$$\begin{aligned} f(p|x \in [750, 780]) &\propto f(x \in [750, 780]|p)f(p) \\ &\propto \left[\sum_{k=750}^{780} \binom{1000}{k} p^k (1-p)^{1000-k} \right] \mathbb{1}\{0.1 \leq p \leq 0.9\} \end{aligned}$$

We can solve for the maximum by plotting the function on Matlab:



Note that the estimated p value becomes smaller.

2.2 2.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Error in derivation/formulation

- 1 pts Partial credit for Q2.2

- 2 pts No work shown for Q2.2

Problem 3

Given a realization of U, let

$$x^* = \inf\{x|F(x) = U\}$$

Then

$$\mathbb{P}\{F^{-1}(U) \leq u\} = \mathbb{P}\{x^* \leq u\} = \mathbb{P}\{F(x^*) \leq F(u)\} = \mathbb{P}\{U \leq F(u)\} = \int_0^{F(u)} dx = F(u)$$

This proves that $F^{-1}(U)$ has a distribution with distribution function F.

Hence, given a probability distribution with distribution F, one can generate a sample from the distribution by first generating a sample from uniform distribution (say, u) and then apply the transformation

$$u \to F^{-1}(u)$$

3 P3 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 0.75 pts Didn't explain how the result can be used to generate a sample of random variables from a given distribution.

- 2 pts Partial credit for Q3

- 4 pts No work shown for Q3

Problem 4

a.
We need

$$c\int_{x_1^2+x_2^2\leq 1}\frac{1}{\sqrt{x_1^2+x_2^2}}=1$$

To evaluate the integral, consider polar transformation $x_1 = r\cos\theta, x_2 = r\sin\theta$:

$$\begin{aligned}\int_{x_1^2+x_2^2\leq 1}\frac{1}{\sqrt{x_1^2+x_2^2}} &= \int_0^1\int_0^{2\pi}\frac{1}{r}\cdot r d\theta dr\\ &= 2\pi\end{aligned}$$

Hence

$$c=\frac{1}{2\pi}$$

b.
We first compute the marginal distribution, note that

$$f_{X_2}(x_2)=\frac{1}{2\pi}\int_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}}\frac{1}{\sqrt{x_1^2+x_2^2}}dx_2$$

Let $x_1 = x_2\tan\theta$, then $dx_1 = x_2\sec^2\theta d\theta$ and

$$f_{X_2}(x_2)=\frac{1}{2\pi}\int_{-\alpha}^{\alpha}\frac{1}{x_2\sec^2\theta}x_2\sec^2\theta d\theta=\frac{1}{2\pi}\int_{-\alpha}^{\alpha}\sec\theta d\theta$$

Where

$$\alpha=\tan^{-1}\left(\frac{\sqrt{1-x_2^2}}{x_2}\right)$$

Evaluating the integral, we have

$$\begin{aligned}f_{X_2}(x_2) &= \frac{1}{2\pi}\ln|\sec\theta+\tan\theta|\Big|_{-\alpha}^{\alpha}\\ &= \frac{1}{2\pi}\ln\left|\frac{\sec\alpha+\tan\alpha}{\sec\alpha-\tan\alpha}\right|\\ &= \frac{1}{2\pi}\ln(\sec\alpha+\tan\alpha)^2\\ &= \frac{1}{\pi}\ln\left|\frac{1+\sqrt{1-x_2^2}}{x_2}\right|\end{aligned}$$

By symmetry, we also have

$$f_{X_1}(x_1)=\frac{1}{\pi}\ln\left|\frac{1+\sqrt{1-x_1^2}}{x_1}\right|$$

Hence, the conditional distribution is given by

$$f_{X_1|X_2}(x_1|x_2)=\frac{f_{X_1,X_2}(x_1,x_2)}{f_{X_2}(x_2)}=\frac{1}{2\sqrt{x_1^2+x_2^2}\ln\left|\frac{1+\sqrt{1-x_2^2}}{x_2}\right|}$$

Likewise,

$$f_{X_2|X_1}(x_2|x_1)\frac{1}{2\sqrt{x_1^2+x_2^2}\ln\left|\frac{1+\sqrt{1-x_1^2}}{x_1}\right|}$$

4.1 4.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q4.1

- 2 pts No work shown for Q4.1

Problem 4

a.
We need

$$c\int_{x_1^2+x_2^2\leq 1}\frac{1}{\sqrt{x_1^2+x_2^2}}=1$$

To evaluate the integral, consider polar transformation $x_1 = r\cos\theta, x_2 = r\sin\theta$:

$$\begin{aligned}\int_{x_1^2+x_2^2\leq 1}\frac{1}{\sqrt{x_1^2+x_2^2}} &= \int_0^1\int_0^{2\pi}\frac{1}{r}\cdot r d\theta dr\\ &= 2\pi\end{aligned}$$

Hence

$$c=\frac{1}{2\pi}$$

b.
We first compute the marginal distribution, note that

$$f_{X_2}(x_2)=\frac{1}{2\pi}\int_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}}\frac{1}{\sqrt{x_1^2+x_2^2}}dx_2$$

Let $x_1 = x_2\tan\theta$, then $dx_1 = x_2\sec^2\theta d\theta$ and

$$f_{X_2}(x_2)=\frac{1}{2\pi}\int_{-\alpha}^{\alpha}\frac{1}{x_2\sec^2\theta}x_2\sec^2\theta d\theta=\frac{1}{2\pi}\int_{-\alpha}^{\alpha}\sec\theta d\theta$$

Where

$$\alpha=\tan^{-1}\left(\frac{\sqrt{1-x_2^2}}{x_2}\right)$$

Evaluating the integral, we have

$$\begin{aligned}f_{X_2}(x_2) &= \frac{1}{2\pi}\ln|\sec\theta+\tan\theta|\Big|_{-\alpha}^{\alpha}\\ &= \frac{1}{2\pi}\ln\left|\frac{\sec\alpha+\tan\alpha}{\sec\alpha-\tan\alpha}\right|\\ &= \frac{1}{2\pi}\ln(\sec\alpha+\tan\alpha)^2\\ &= \frac{1}{\pi}\ln\left|\frac{1+\sqrt{1-x_2^2}}{x_2}\right|\end{aligned}$$

By symmetry, we also have

$$f_{X_1}(x_1)=\frac{1}{\pi}\ln\left|\frac{1+\sqrt{1-x_1^2}}{x_1}\right|$$

Hence, the conditional distribution is given by

$$f_{X_1|X_2}(x_1|x_2)=\frac{f_{X_1,X_2}(x_1,x_2)}{f_{X_2}(x_2)}=\frac{1}{2\sqrt{x_1^2+x_2^2}\ln\left|\frac{1+\sqrt{1-x_2^2}}{x_2}\right|}$$

Likewise,

$$f_{X_2|X_1}(x_2|x_1)\frac{1}{2\sqrt{x_1^2+x_2^2}\ln\left|\frac{1+\sqrt{1-x_1^2}}{x_1}\right|}$$

4.2 4.2 **2 / 2**

- ✓ **- 0 pts** *Entirely correct*
- **- 0.3 pts** Correct marginal densities, but incorrect conditional density
- **- 0.3 pts** Slight error in calculating marginal densities
- **- 0.4 pts** Error in calculating marginal densities
- **- 1 pts** Partial credit for Q4.2
- **- 2 pts** No work shown for Q4.2

c.

Since

$$f_{X_1|X_2}(x_1|x_2) \neq f_{X_1}(x_1)$$

It follows that X_1 and X_2 are not independent.

To check whether X_1, X_2 are correlated, we compute the covariance between X_1, X_2

$$\text{Cov}(X_1, X_2) = \mathbb{E}[X_1X_2] - \mathbb{E}[X_1]\mathbb{E}[X_2]$$

We first compute

$$\begin{aligned}
\mathbb{E}[X_1X_2] &= \frac{1}{2\pi}\int_{x_1^2+x_2^2\leq 1} \frac{x_1x_2}{\sqrt{x_1^2+x_2^2}}dx_1dx_2 \\
&= \frac{1}{2\pi}\int_0^1\int_0^{2\pi} \frac{r^2\cos\theta\sin\theta}{r} rdrd\theta \\
&= \frac{1}{2\pi}\left(\int_0^1 r^2\right)\left(\int_0^{2\pi}\sin\theta\cos\theta d\theta\right) \\
&= 0
\end{aligned}$$

We then compute

$$\mathbb{E}[X_1] = \frac{1}{\pi}\int_{-1}^{1} x_1 \ln\left|\frac{1+\sqrt{1-x_1^2}}{x_1}\right| dx_1 = 0$$

The integral is 0 since the integrand is an odd function.
Likewise, $\mathbb{E}[X_2] = 0$
Since

$$\text{Cov}(X_1, X_2) = \mathbb{E}[X_1X_2] - \mathbb{E}[X_1]\mathbb{E}[X_2] = 0$$

it follows that X_1, X_2 are uncorrelated.

4.3 4.3 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct about x_1 and x_2 being stochastically dependent, but said x_1 and x_2 are correlated.

- 1 pts Partial credit for Q4.3

- 2 pts `No work shown for Q4.3

In [24]:
```python
import math
import numpy as np
import matplotlib.pyplot as plt
```

Problem 6

Let A be the matrix

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

By Cayley–Hamilton theorem, A satisfies its own characteristic polynomial $p(\lambda)$. </br> Let $\lambda_1, \lambda_2, \lambda_3$ be its eigenvalue, then

$$p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)(\lambda - \lambda_3)$$

By comparing coefficients, we see that

$$\begin{cases} a = -(\lambda_1 + \lambda_2 + \lambda_3) \\ b = \lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1 \\ c = -\lambda_1\lambda_2\lambda_3 \end{cases}$$

In [2]:
```python
# Create the matrix A, 3X3 identity matrix, and zero matrix
A = np.matrix([[1, -1, 1], [-1, 0, -1], [1, -1, -1]])
A
```

Out[2]:
```
matrix([[ 1, -1,  1],
        [-1,  0, -1],
        [ 1, -1, -1]])
```

In [3]:
```python
# Compute the eigenvalues of A
eigenvalues = np.linalg.eigvals(A)
```

In [4]:
```python
eigenvalues
```

Out[4]:
```
array([ 2.21431974, -0.53918887, -1.67513087])
```

In [5]:
```python
# Compute the solutions a, b, c
a = -(eigenvalues[0] + eigenvalues[1] + eigenvalues[2])
b = eigenvalues[0] * eigenvalues[1] + eigenvalues[1] * eigenvalues[2] + eigenvalues[2] *
c = -eigenvalues[0] * eigenvalues[1] * eigenvalues[2]
print(f"The solutions are a = {a}, b = {b}, c = {c}")
```

```
The solutions are a = 1.9984014443252818e-15, b = -3.999999999999996, c = -1.99999999999
99998
```

In [6]:
```python
# Testing whether computed a, b, c are indeed solutions
I = np.identity(3)
print(np.linalg.matrix_power(A, 3) + a * np.linalg.matrix_power(A, 2) + b * A + c * I)
```

```
[[ 1.04360964e-14 -7.54951657e-15  5.77315973e-15]
 [-7.54951657e-15  4.21884749e-15 -3.99680289e-15]
 [ 5.77315973e-15 -3.99680289e-15  1.99840144e-15]]
```

Hence, the computed solution is given by
$a = 1.9984014443252818e-15, b = -3.999999999999996, c = -1.9999999999999998$. </br>

5 P5 **0 / 0**

✓ **- 0 pts** *Optional*

In [24]:
```python
import math
import numpy as np
import matplotlib.pyplot as plt
```

Problem 6

Let A be the matrix

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

By Cayley–Hamilton theorem, A satisfies its own characteristic polynomial $p(\lambda)$. </br> Let $\lambda_1, \lambda_2, \lambda_3$ be its eigenvalue, then

$$p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)(\lambda - \lambda_3)$$

By comparing coefficients, we see that

$$\begin{cases} a = -(\lambda_1 + \lambda_2 + \lambda_3) \\ b = \lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1 \\ c = -\lambda_1\lambda_2\lambda_3 \end{cases}$$

In [2]:
```python
# Create the matrix A, 3X3 identity matrix, and zero matrix
A = np.matrix([[1, -1, 1], [-1, 0, -1], [1, -1, -1]])
A
```

Out[2]:
```
matrix([[ 1, -1,  1],
        [-1,  0, -1],
        [ 1, -1, -1]])
```

In [3]:
```python
# Compute the eigenvalues of A
eigenvalues = np.linalg.eigvals(A)
```

In [4]:
```python
eigenvalues
```

Out[4]:
```
array([ 2.21431974, -0.53918887, -1.67513087])
```

In [5]:
```python
# Compute the solutions a, b, c
a = -(eigenvalues[0] + eigenvalues[1] + eigenvalues[2])
b = eigenvalues[0] * eigenvalues[1] + eigenvalues[1] * eigenvalues[2] + eigenvalues[2] *
c = -eigenvalues[0] * eigenvalues[1] * eigenvalues[2]
print(f"The solutions are a = {a}, b = {b}, c = {c}")
```

```
The solutions are a = 1.9984014443252818e-15, b = -3.999999999999996, c = -1.99999999999
99998
```

In [6]:
```python
# Testing whether computed a, b, c are indeed solutions
I = np.identity(3)
print(np.linalg.matrix_power(A, 3) + a * np.linalg.matrix_power(A, 2) + b * A + c * I)
```

```
[[ 1.04360964e-14 -7.54951657e-15  5.77315973e-15]
 [-7.54951657e-15  4.21884749e-15 -3.99680289e-15]
 [ 5.77315973e-15 -3.99680289e-15  1.99840144e-15]]
```

Hence, the computed solution is given by
$a = 1.9984014443252818e-15, b = -3.999999999999996, c = -1.9999999999999998$. </br>

Plugging in these into the equation indeed gives us the zero matrix. </br> Once rounded, the solutions are $a = 0, b = -4, c = -2$, which is the true value if calculated by hand.

Problem 7

Problem (1), (2)

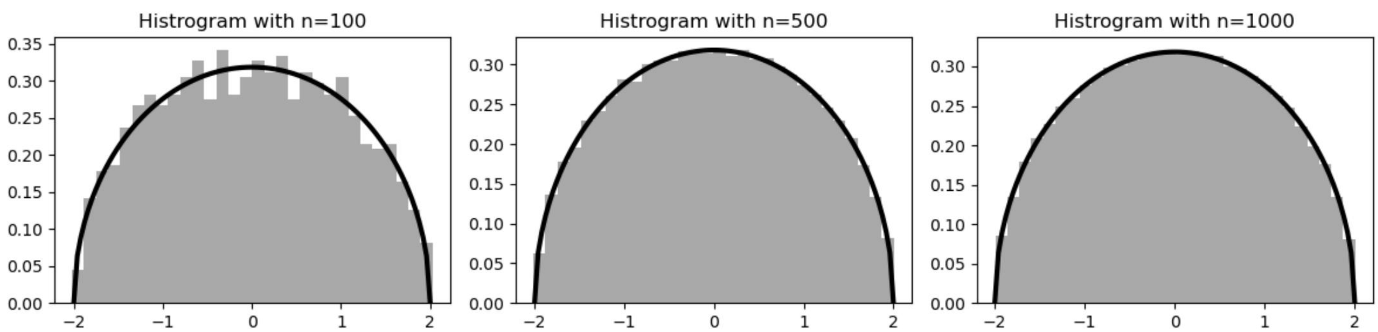
In the following problem, due to computational issues, we restrict matrix size to 100, 500, and 1000

In [8]:
```
# M_W takes in matrix size as argument and returns a random matrix that's scaled by sqrt
def MW(n):
    diagonal = np.random.normal(loc=0, scale=np.sqrt(2), size=n)
    upper_triangular = np.random.normal(size=(n, n))
    upper_triangular = np.triu(upper_triangular, k=1)
    W = upper_triangular + upper_triangular.T
    np.fill_diagonal(W, diagonal)
    return W / n ** 0.5
MW(3)
```

Out[8]:
```
array([[ 0.84697741, -0.64765219,  0.47926734],
       [-0.64765219,  0.24978478, -1.16776298],
       [ 0.47926734, -1.16776298,  0.21837789]])
```

In [9]:
```
# Define the Wigner semi-circle density
def wigner(x):
    return np.sqrt(4 - x ** 2) / (2 * math.pi)
```

In [11]:
```
# Plot a histogram of the eigenvalues of M_W with n=100, 500, 1000, along with the Wigne
def plot_MW():
    fig, axes = plt.subplots(1, 3, figsize=(12, 3))
    matrix_size = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        size = matrix_size[i]
        eigenvalues = []
        for j in range(num_samples):
            mw = MW(size)
            for e in np.linalg.eigvals(mw):
                eigenvalues.append(e)
        x = np.linspace(-2, 2, 100)
        y = wigner(x)
        axes[i].plot(x, y, color="black", linewidth=3.0)
        axes[i].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
        axes[i].set_title(f"Histrogram with n={size}")
    plt.tight_layout()
    plt.show()
plot_MW()
```



Note that the distribution of the eigenvalues follows the same density as the Wiegner semi-circle

6 P6 **5 / 5**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Correct approach, but more work is needed to support your answer or key error in derivation.

- 2.5 pts Partial credit for Q6

- 5 pts No work shown for Q6

Plugging in these into the equation indeed gives us the zero matrix. </br> Once rounded, the solutions are $a = 0, b = -4, c = -2$, which is the true value if calculated by hand.

Problem 7

Problem (1), (2)

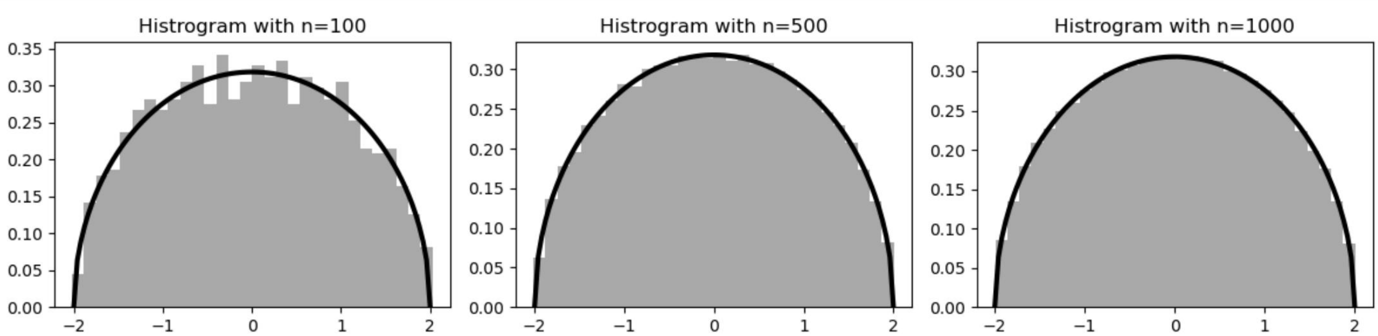
In the following problem, due to computational issues, we restrict matrix size to 100, 500, and 1000

In [8]:
```
# M_W takes in matrix size as argument and returns a random matrix that's scaled by sqrt
def MW(n):
    diagonal = np.random.normal(loc=0, scale=np.sqrt(2), size=n)
    upper_triangular = np.random.normal(size=(n, n))
    upper_triangular = np.triu(upper_triangular, k=1)
    W = upper_triangular + upper_triangular.T
    np.fill_diagonal(W, diagonal)
    return W / n ** 0.5
MW(3)
```

Out[8]:
```
array([[ 0.84697741, -0.64765219,  0.47926734],
       [-0.64765219,  0.24978478, -1.16776298],
       [ 0.47926734, -1.16776298,  0.21837789]])
```

In [9]:
```
# Define the Wigner semi-circle density
def wigner(x):
    return np.sqrt(4 - x ** 2) / (2 * math.pi)
```

In [11]:
```
# Plot a histogram of the eigenvalues of M_W with n=100, 500, 1000, along with the Wigne
def plot_MW():
    fig, axes = plt.subplots(1, 3, figsize=(12, 3))
    matrix_size = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        size = matrix_size[i]
        eigenvalues = []
        for j in range(num_samples):
            mw = MW(size)
            for e in np.linalg.eigvals(mw):
                eigenvalues.append(e)
        x = np.linspace(-2, 2, 100)
        y = wigner(x)
        axes[i].plot(x, y, color="black", linewidth=3.0)
        axes[i].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
        axes[i].set_title(f"Histrogram with n={size}")
    plt.tight_layout()
    plt.show()
plot_MW()
```



Note that the distribution of the eigenvalues follows the same density as the Wiegner semi-circle

7.1 7.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q7.1

- 2 pts No work shown for Q7.1

Plugging in these into the equation indeed gives us the zero matrix. </br> Once rounded, the solutions are $a = 0, b = -4, c = -2$, which is the true value if calculated by hand.

Problem 7

Problem (1), (2)

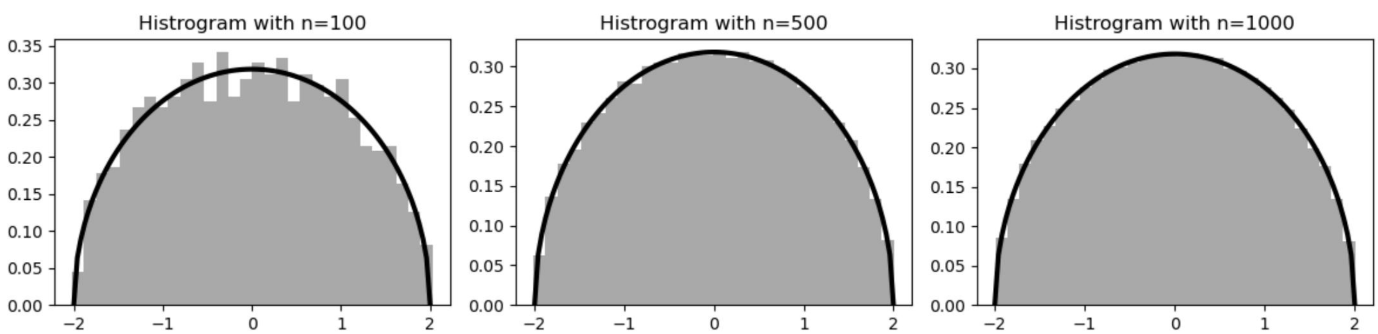
In the following problem, due to computational issues, we restrict matrix size to 100, 500, and 1000

In [8]:
```
# M_W takes in matrix size as argument and returns a random matrix that's scaled by sqrt
def MW(n):
    diagonal = np.random.normal(loc=0, scale=np.sqrt(2), size=n)
    upper_triangular = np.random.normal(size=(n, n))
    upper_triangular = np.triu(upper_triangular, k=1)
    W = upper_triangular + upper_triangular.T
    np.fill_diagonal(W, diagonal)
    return W / n ** 0.5
MW(3)
```

Out[8]:
```
array([[ 0.84697741, -0.64765219,  0.47926734],
       [-0.64765219,  0.24978478, -1.16776298],
       [ 0.47926734, -1.16776298,  0.21837789]])
```

In [9]:
```
# Define the Wigner semi-circle density
def wigner(x):
    return np.sqrt(4 - x ** 2) / (2 * math.pi)
```

In [11]:
```
# Plot a histogram of the eigenvalues of M_W with n=100, 500, 1000, along with the Wigne
def plot_MW():
    fig, axes = plt.subplots(1, 3, figsize=(12, 3))
    matrix_size = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        size = matrix_size[i]
        eigenvalues = []
        for j in range(num_samples):
            mw = MW(size)
            for e in np.linalg.eigvals(mw):
                eigenvalues.append(e)
        x = np.linspace(-2, 2, 100)
        y = wigner(x)
        axes[i].plot(x, y, color="black", linewidth=3.0)
        axes[i].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
        axes[i].set_title(f"Histrogram with n={size}")
    plt.tight_layout()
    plt.show()
plot_MW()
```



Note that the distribution of the eigenvalues follows the same density as the Wiegner semi-circle

7.2 7.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Q7.2

- 2 pts No work shown for Q7.2

distribution.

Problem (3)

The matrix

$$A = \frac{1}{n} A A^T$$

Has size $m \times m$ and represent the covariance matrix of the m features.

Problem (4), (5)

In [25]:
```
# M_A takes in matrix size as argument and returns a random matrix that's scaled by 1/n
def MA(m, n):
    A = np.random.randn(m, n)
    M_A = np.dot(A, A.T)
    return M_A / n
MA(3, 4)
```

Out[25]:
```
array([[ 2.02943112,  0.03099195, -0.98938068],
       [ 0.03099195,  1.06486093,  0.19160984],
       [-0.98938068,  0.19160984,  1.35968015]])
```

In [26]:
```
def marchenko(x, m, n):
    a = (1 - np.sqrt(m / n)) ** 2
    b = (1 + np.sqrt(m / n)) ** 2
    numerator = np.sqrt((b-x) * (x-a))
    denominator = 2 * math.pi * x
    return numerator / denominator
```

In [28]:
```
# Plot a histogram of the eigenvalues of M_A with n=100, 500, 1000, along with the March
def plot_MA():
    fig, axes = plt.subplots(3, 3, figsize=(12, 8))
    m = [100, 500, 1000]
    n = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        for j in range(3):
            m_size, n_size = m[i], n[i]
            eigenvalues = []
            for k in range(num_samples):
                ma = MA(m_size, n_size)
                for e in np.linalg.eigvals(ma):
                    eigenvalues.append(e)
            x = np.linspace(0.1, 4, 100)
            y = marchenko(x, m_size, n_size)
            axes[i, j].plot(x, y, color="black", linewidth=3.0)
            axes[i, j].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
            axes[i, j].set_title(f"Histrogram with shape {(m_size, n_size)}")
    plt.tight_layout()
    plt.show()
plot_MA()
```

7.3 7.3 **2 / 2**

- ✓ **- 0 pts** *Entirely correct*
- **- 0.5 pts** More explanation is needed to support your answer.
- **- 1 pts** Partial credit for Q7.3
- **- 2 pts** No work shown for Q7.3

distribution.

Problem (3)

The matrix

$$A = \frac{1}{n} A A^T$$

Has size $m \times m$ and represent the covariance matrix of the m features.

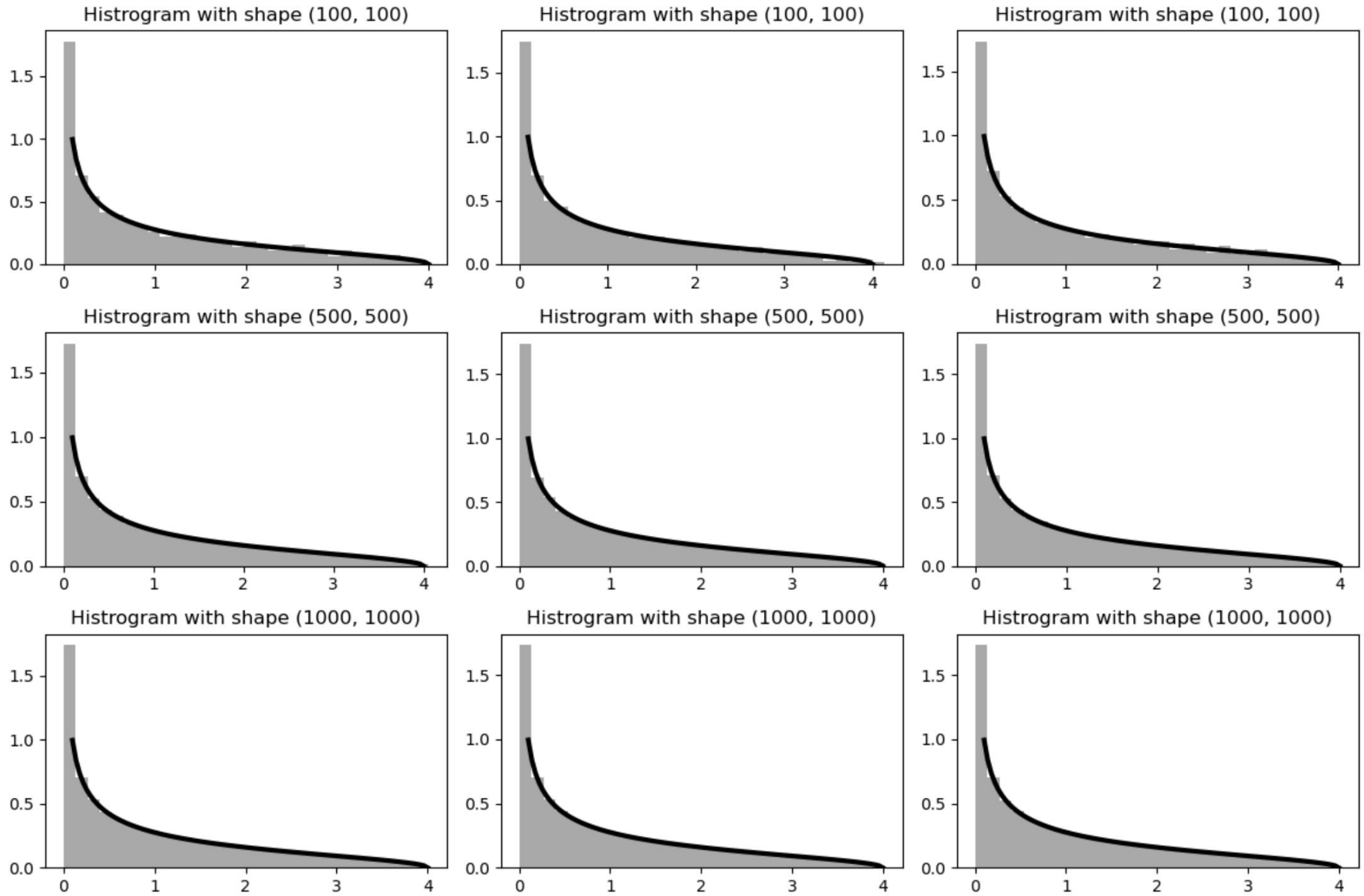
Problem (4), (5)

In [25]:
```
# M_A takes in matrix size as argument and returns a random matrix that's scaled by 1/n
def MA(m, n):
    A = np.random.randn(m, n)
    M_A = np.dot(A, A.T)
    return M_A / n
MA(3, 4)
```

Out[25]:
```
array([[ 2.02943112,  0.03099195, -0.98938068],
       [ 0.03099195,  1.06486093,  0.19160984],
       [-0.98938068,  0.19160984,  1.35968015]])
```

In [26]:
```
def marchenko(x, m, n):
    a = (1 - np.sqrt(m / n)) ** 2
    b = (1 + np.sqrt(m / n)) ** 2
    numerator = np.sqrt((b-x) * (x-a))
    denominator = 2 * math.pi * x
    return numerator / denominator
```

In [28]:
```
# Plot a histogram of the eigenvalues of M_A with n=100, 500, 1000, along with the March
def plot_MA():
    fig, axes = plt.subplots(3, 3, figsize=(12, 8))
    m = [100, 500, 1000]
    n = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        for j in range(3):
            m_size, n_size = m[i], n[i]
            eigenvalues = []
            for k in range(num_samples):
                ma = MA(m_size, n_size)
                for e in np.linalg.eigvals(ma):
                    eigenvalues.append(e)
            x = np.linspace(0.1, 4, 100)
            y = marchenko(x, m_size, n_size)
            axes[i, j].plot(x, y, color="black", linewidth=3.0)
            axes[i, j].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
            axes[i, j].set_title(f"Histrogram with shape {(m_size, n_size)}")
    plt.tight_layout()
    plt.show()
plot_MA()
```



Note that the distribution of the eigenvalues follows the same density as the Marchenko-Pastur density

Exporting file to pdf

In [9]:
```
import plotly.express as px
!pip install Pyppeteer
!pyppeteer-install
```

```
Collecting Pyppeteer
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
     |████████████████████████████████| 83 kB 1.5 MB/s eta 0:00:011
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (1.4.4)
Requirement already satisfied: certifi>=2021 in /Users/raymondtsao/.local/lib/python3.1
0/site-packages (from Pyppeteer) (2023.5.7)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in /Users/raymondtsao/anaconda3/lib/p
ython3.10/site-packages (from Pyppeteer) (4.64.1)
Requirement already satisfied: websockets<11.0,>=10.0 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (10.4)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: importlib-metadata>=1.4 in /Users/raymondtsao/anaconda3/l
ib/python3.10/site-packages (from Pyppeteer) (4.11.3)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (1.26.14)
Requirement already satisfied: zipp>=0.5 in /Users/raymondtsao/anaconda3/lib/python3.10/
site-packages (from importlib-metadata>=1.4->Pyppeteer) (3.11.0)
Installing collected packages: pyee, Pyppeteer
Successfully installed Pyppeteer-1.0.2 pyee-8.2.2
WARNING: You are using pip version 23.1.2; however, version 23.2.1 is available.
You should consider upgrading via the '/Users/raymondtsao/anaconda3/bin/python -m pip in
stall --upgrade pip' command.
[INFO] Starting Chromium download.
100%|██████████████████████████████████████| 86.8M/86.8M [00:04<00:00, 18.3Mb/s]
```

```
[INFO] Beginning extraction
[INFO] Chromium extracted to: /Users/raymondtsao/Library/Application Support/pyppeteer/l
ocal-chromium/588429
```

In [ ]:

7.4 7.4 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Not all histograms were plotted

- 1 pts Partial credit for Q7.4

- 2 pts No work shown for Q7.4

distribution.

Problem (3)

The matrix

$$A = \frac{1}{n} AA^T$$

Has size $m \times m$ and represent the covariance matrix of the m features.

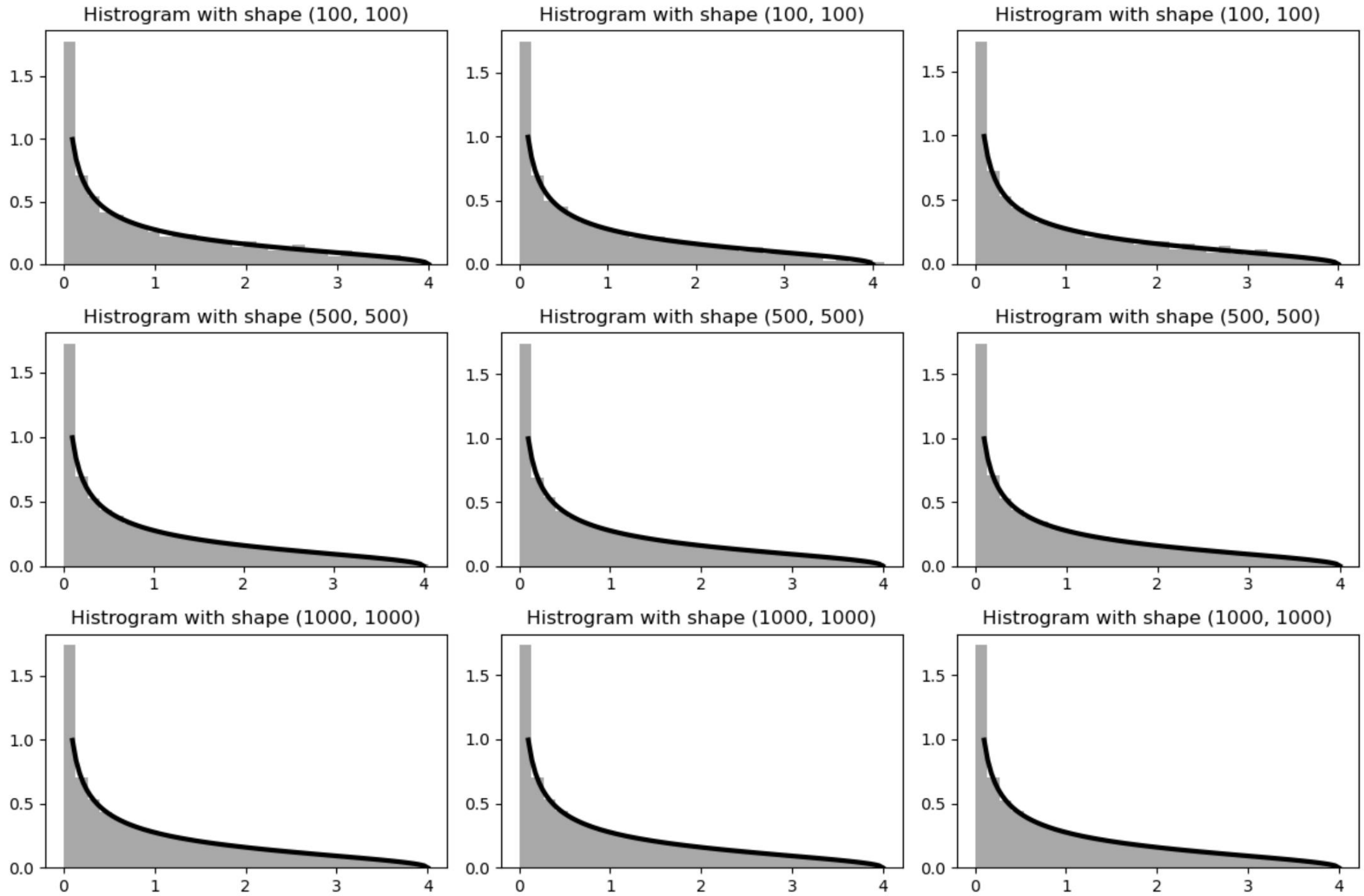
Problem (4), (5)

In [25]:
```
# M_A takes in matrix size as argument and returns a random matrix that's scaled by 1/n
def MA(m, n):
    A = np.random.randn(m, n)
    M_A = np.dot(A, A.T)
    return M_A / n
MA(3, 4)
```

Out[25]:
```
array([[ 2.02943112,  0.03099195, -0.98938068],
       [ 0.03099195,  1.06486093,  0.19160984],
       [-0.98938068,  0.19160984,  1.35968015]])
```

In [26]:
```
def marchenko(x, m, n):
    a = (1 - np.sqrt(m / n)) ** 2
    b = (1 + np.sqrt(m / n)) ** 2
    numerator = np.sqrt((b-x) * (x-a))
    denominator = 2 * math.pi * x
    return numerator / denominator
```

In [28]:
```
# Plot a histogram of the eigenvalues of M_A with n=100, 500, 1000, along with the March
def plot_MA():
    fig, axes = plt.subplots(3, 3, figsize=(12, 8))
    m = [100, 500, 1000]
    n = [100, 500, 1000]
    num_samples = 10
    for i in range(3):
        for j in range(3):
            m_size, n_size = m[i], n[i]
            eigenvalues = []
            for k in range(num_samples):
                ma = MA(m_size, n_size)
                for e in np.linalg.eigvals(ma):
                    eigenvalues.append(e)
            x = np.linspace(0.1, 4, 100)
            y = marchenko(x, m_size, n_size)
            axes[i, j].plot(x, y, color="black", linewidth=3.0)
            axes[i, j].hist(eigenvalues, bins=30, density=True, color="#A9A9A9")
            axes[i, j].set_title(f"Histrogram with shape {(m_size, n_size)}")
    plt.tight_layout()
    plt.show()
plot_MA()
```



Note that the distribution of the eigenvalues follows the same density as the Marchenko-Pastur density

Exporting file to pdf

In [9]:
```
import plotly.express as px
!pip install Pyppeteer
!pyppeteer-install
```

```
Collecting Pyppeteer
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
     |████████████████████████████████| 83 kB 1.5 MB/s eta 0:00:011
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (1.4.4)
Requirement already satisfied: certifi>=2021 in /Users/raymondtsao/.local/lib/python3.1
0/site-packages (from Pyppeteer) (2023.5.7)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in /Users/raymondtsao/anaconda3/lib/p
ython3.10/site-packages (from Pyppeteer) (4.64.1)
Requirement already satisfied: websockets<11.0,>=10.0 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (10.4)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: importlib-metadata>=1.4 in /Users/raymondtsao/anaconda3/l
ib/python3.10/site-packages (from Pyppeteer) (4.11.3)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in /Users/raymondtsao/anaconda3/li
b/python3.10/site-packages (from Pyppeteer) (1.26.14)
Requirement already satisfied: zipp>=0.5 in /Users/raymondtsao/anaconda3/lib/python3.10/
site-packages (from importlib-metadata>=1.4->Pyppeteer) (3.11.0)
Installing collected packages: pyee, Pyppeteer
Successfully installed Pyppeteer-1.0.2 pyee-8.2.2
WARNING: You are using pip version 23.1.2; however, version 23.2.1 is available.
You should consider upgrading via the '/Users/raymondtsao/anaconda3/bin/python -m pip in
stall --upgrade pip' command.
[INFO] Starting Chromium download.
100%|██████████████████████████████████████| 86.8M/86.8M [00:04<00:00, 18.3Mb/s]
```

```
[INFO] Beginning extraction
[INFO] Chromium extracted to: /Users/raymondtsao/Library/Application Support/pyppeteer/local-chromium/588429
```

In [ ]:

```

```

7.5 7.5 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Not all shapes computed by the Marchenko-Pastur Law are plotted

- 1 pts Partial credit for Q7.5

- 2 pts No work shown for Q7.5