Project list for Integration Workshop

Colin Clark
Graduate Program in Applied Mathematics,
University of Arizona, Tucson

August 2021

Objectives:

1. Making sure each student has a basic working knowledge of Matlab, Python, Julia, or some other high-level programming language for scientific computing.
2. Making sure each student has a basic working knowledge of LaTeX.
3. Making sure students can use computational approaches to solve math problems (turning mathematics into computer code).
4. Planning, writing, sharing, and discussing algorithms in a group setting.
5. Having fun!

1 The Fredholm Alternative & Least Squares Solutions

Sometimes we focus too much on solving the matrix equation $A\boldsymbol{x} = \boldsymbol{b}$ for situations where A is a square matrix, and we ignore situations where A is not square. In many practical applications, A is an $m \times n$ matrix with $m \neq n$. In this project, you are going to explore what we mean by *solutions to the matrix equation for non-square matrices.*

The Fredholm Alternative

The Fredholm alternative states that for the matrix equation $Ax = b$, exactly one of the following statements is true:

- (Either) There exists an x that solves the matrix equation $Ax = b$.
- (Or) There exists a y that solves $A^\top y = 0$ such that $y^\top b \neq 0$.

Let A_m be the $3 \times m$ matrix (below), and let b' and b'' be the 3×1 vectors (below).

$$A_m = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & m \end{bmatrix}, \qquad b' = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix}, \quad b'' = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix}$$

(a) Plot the vectors that make the columns of A_m for $m = 5$, and use your figure to describe the column space of A_m.

(b) Use pencil-and-paper to verify the Fredholm Alternative for b' and for b''.

(c) Plot the vectors b' and b'' on the same figure from part (a), and use your figure to provide an intuitive explanation for the Fredholm Alternative.

Pseudo-inverses & Least Squares Solutions

When there is no solution to the matrix equation $Ax = b$, we may have to look for the 'next best thing'. Your co-worker suggests the following matrix algebra to find the 'next best thing'.

$$Ax = b \quad \Rightarrow \quad A^\top Ax = A^\top b \quad \Rightarrow \quad x = (A^\top A)^{-1} A^\top b$$

(d) What are the dimensions of the matrix $(A^\top A)$? What are requirements on A for the matrix $(A^\top A)$ to be invertable?

(e) For $m = 2$, find the matrix $A^\dagger := (A_m^\top A_m) A_m^\top$ and compute the vectors $x' := A^\dagger b'$ and $x'' := A^\dagger b''$.

(f) Does $Ax' = b'$? What about $Ax'' = b''$? Can you describe what we mean when we say that $A^\dagger$ gives the 'next best thing'?

(∗) *Bonus:* Use some calculus to verify your answer in part (f).

2 Lagrange Interpolation for Function Approximation

Let $\{(x_k, y_k), \text{for}\, k = 1, \ldots, N\}$ be N points in $\mathbb{R}^2$ that satisfy $x_k \neq x_j$ whenever $k \neq j$. The *Lagrange interpolating polynomial* is defined as

$$L(x) = \sum_{k=1}^{N} \left(\prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \right) y_k. \tag{2.1}$$

In this project, you will determine how well a Lagrange interpolating polynomial can approximate the functions

$$f(x) = (x - .9)(x - .4)(x + .1)(x + .7)(x + .8) \tag{2.2}$$

and

$$g(x) = \frac{1}{1 + 10x^2} \qquad \text{for } -1 \leq x \leq 1 \tag{2.3}$$

(a) Observe that the Lagrange interpolating polynomial is a linear combination of N terms in the form

$$T_k(x) := \left(\prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \right) \tag{2.4}$$

Let $x_1 = -1$, $x_2 = 0$, and $x_3 = 1$. Plot the functions $T_k(x)$ for $k = 1, \ldots, 3$, and describe what you see.

(b) i. Sample $f(x)$ at the points $x_1 = -1$, $x_2 = 0$, and $x_3 = 1$. Compute and plot the Lagrange interpolating polynomial given by these 3 equi-spaced points.

ii. Repeat part (b)(i) with 5 equi-spaced points, and again with 9 equi-spaced points. Plot your results and describe whether your approximation improves with more points.

(c) Repeat part (b) for the function $g(x)$.

(d) There is a well-known rule of thumb for improving numerical approximations: Increase the sampling density in the regions where the error is greatest. Does this rule of thumb seem to work with the functions f and g?

3 Contour Integration of a Complex-valued Function

(This problem is probably out of reach for students who not yet taken a course in complex analysis. No worries, we will cover this topic during the first few weeks of Math 583A.)

Consider the functions $f : \mathbb{C} \to \mathbb{C}$ and $g : \mathbb{C} \to \mathbb{C}$ defined as

$$f(z) = z^2 \qquad \text{and} \qquad g(z) = z^{-1}$$

In this project, you will write a first-order numerical approximation scheme to compare the integrals of f and of g along two different contours.

(a) Plotting:

i. Make a surface plot showing $Re(f(z))$ and a surface plot showing $Im(f(z))$

ii. Make a surface plot showing $Re(g(z))$ and a surface plot showing $Im(g(z))$

Since g is undefined at $z = 0$ and since it is radially symmetric, the surface plots for g can look scary when done in cartesian co-ordinates. You may try switching to polar coordinates to make prettier plots

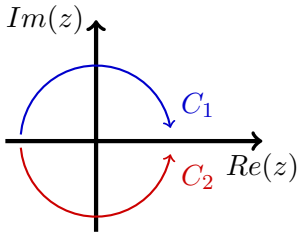
(b) We can approximate the line integral (or contour integral) of a function along a curve by discretizing the curve into $N + 1$ points $z(s) \to \{z_0, z_1, z_2, \ldots, z_N\}$ and then computing the sum:

$$\int_{z_o}^{z_t} f(z)dz \approx \sum_{k=1}^{N} f(z_k)\ (z_k - z_{k-1})$$

Define the two curves:

C_1 : The upper half circle of radius 1 centered at $z = 0$.

C_2 : The lower half circle of radius 1 centered at $z = 0$.



Approximate the line integrals of f and of g along the curves C_1 and C_2.

(∗) *Bonus:* Repeat parts (a) and (b) for the function $h(z) = z^{1/2}$.
Note: You may need to make some 'executive decisions' to make sure your problem is well-posed.

4 Eigenvectors of Circulant Matrices

Define the $n \times n$ matrix A by

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \tag{4.1}$$

In this project, you will explore (and explain) the behavior of the iterative map

$$\boldsymbol{x}^{(k+1)} = (I + \eta A)\, \boldsymbol{x}^{(k)} \tag{4.2}$$

for various initial vectors $\boldsymbol{x}^{(0)}$, and for $\eta = 0.1$.

(a) Create a function that takes in the argument n and returns the $n \times n$ matrix A. *Hint:* The built-in functions `diag(...)` (Matlab), or `np.diag(...)` (Python), or `diagm(...)` (Julia) may be useful.

(b) Let $n = 32$. Create the n dimensional initial vector $\boldsymbol{x}^{(0)} = (1, 0, 0, \dots, 0)$ and plot $\boldsymbol{x}^{(0)}$, $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}, \dots$. Repeat a selection of initial vectors until you can qualitatively describe the behavior of the map $\boldsymbol{x} \to (I + \eta A)\boldsymbol{x}$?

(c) Compute the eigenvectors of $(I+\eta A)$ by using the function `[X,V]=eig(...)` (Matlab), `V,X = np.linalg.eig(...)` (Python), or `V,X=eigen(...)` (Julia).

(d) Plot enough of the eigenvectors so you get a sense for what they look like.
Hint: If your software returned complex-valued eigenvectors, it may be helpful to plot real and imaginary components separately, for example, in Julia, this would be `plot(real(X[k,:]))` and `plot(imag(X[k,:]))`.

(∗) *Bonus:* Propose a closed form representation for the eigenvectors. Can you prove that every symmetric circulant matrix has eigenvectors in this form?

(∗) *(Bonus:)* Let C be an (arbitrary) circulant matrix. One of the (many) expressions

for the k^{th} eigenvector of a circulant matrix is

$$\boldsymbol{v}_k = \begin{pmatrix} e^{(2\pi i)0k/n} \\ e^{(2\pi i)1k/n} \\ e^{(2\pi i)2k/n} \\ \vdots \\ e^{(2\pi i)(n-1)k/n} \end{pmatrix} \tag{4.3}$$

with corresponding eigenvalues given by the product of c, the first row of C, with the eigenvector,

$$\lambda_k = cv_k \tag{4.4}$$

Verify that $\boldsymbol{x}_k$ as defined in equation (4.3) is an eigenvector of an aribitrary circulant matrix, and that the corresponding eigenvalue is λ_k, as give by equation (4.4).

(e) The $n \times n$ *Fourier Matrix* is the $n \times n$ matrix whose columns are the eigenvectors $\boldsymbol{v}_0$, $\boldsymbol{v}_1$,..., $\boldsymbol{v}_{n-1}$. Write a function that takes argument n and returns the $n \times n$ Fourier matrix, F.

(f) Without any further computation, and by only considering eigenvectors and their corresponding eigenvalues, describe the evolution of the iterative map in equation (4.2) for an initial vector in the form

$$\boldsymbol{x}^{(0)} = \big(\cos(2\pi j/n) + 5\cos(4 \cdot 2\pi j/n), \text{ for } j = 1, ..., n \big).$$

Repeat these steps for the $n \times n$ matrix B, defined by

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & -1 & 0 \end{bmatrix} \tag{4.5}$$

5 First-order ODE Integrators

Consider the differential equation

$$\begin{cases} \dot{x}(t) = -y, & x(0) = 1 \\ \dot{y}(t) = \quad x, & y(0) = 0 \end{cases} \tag{5.1}$$

Use pencil-and-paper to solve the ODE. Make a quiver plot. Sketch solutions.

In this project, you will build a numerical solver for differential equations like this one. You will test your solver on this ODE, and then use it to solve a more interesting ODE. For this problem, discretize the time interval $0 \le t \le T$ into $n+1$ points. That is, set $\Delta t = T/n$ and $t_k = k\,\Delta t$, so $t = \{t_0, t_1, \ldots, t_n\}$.

Forward Differences:

(a) Use what's called a *first-order forward difference* approximation to the derivatives $\dot{x}$ and $\dot{y}$ at each t as follows:

$$\dot{x}(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \qquad \text{and} \qquad \dot{y}(t) \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}. \tag{5.2}$$

Substitute equation (5.2) into equation (5.1) and show that we can approximate the differential equation by the discrete time-stepping process:

$$\begin{cases} x_{k+1} = x_k - (\Delta t) y_k, & x_0 = 1 \\ y_{k+1} = y_k + (\Delta t) x_k, & y_0 = 0 \end{cases} \tag{5.3}$$

(b) Implement equation 5.3 for $0 \le t \le 1$ and plot the trajectory given by the solution.

(c) Is your approximation qualitatively correct? Will it remain qualitatively correct on $0 \le t \le T$ as T gets very large? Explain.

Hint: It may be useful to rewrite the system of difference equations in matrix form,

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & -\Delta t \\ \Delta t & 1 \end{bmatrix} \boldsymbol{x}_k, \quad \boldsymbol{x}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and to analyze the eigenvalues of the matrix.

Backward Differences:

(d) We could have solved this problem by a different approach. We could also have used what's called a *first-order backward difference*:

$$\dot{x}(t) \approx \frac{x(t) - x(t-\Delta t)}{\Delta t} \qquad \text{and} \qquad \dot{y}(t) \approx \frac{y(t) - y(t\Delta t)}{\Delta t}. \tag{5.4}$$

Show that the backward difference gives the approximation

$$\begin{cases} x_k = x_{k-1} - (\Delta t) y_k, & x_0 = 1 \\ y_k = y_{k-1} + (\Delta t) x_k, & y_0 = 0 \end{cases} \tag{5.5}$$

which can be rewritten as

$$\boldsymbol{x}_k = \left(\begin{bmatrix} 1 & \Delta t \\ -\Delta t & 1 \end{bmatrix} \right)^{-1} \boldsymbol{x}_{k-1}, \quad \boldsymbol{x}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

(e) Implement equation (5.5) for $0 \leq t \leq 1$ and plot the trajectory given by the solution.

(f) Is your approximation qualitatively correct? Will it remain qualitatively correct on $0 \leq t \leq T$ as T gets very large? Explain.

Bonus:

(g) (The really fun stuff) Use your ODE solver to solve $\dot{x} = -y$, $\dot{y} = \sin x$.

6 Random Walk on a Grid

A 'particle' takes a random walk on a 4×8 grid. At each step of the walk, the particle may move up, down, left, or right with equal probability. The particle continues this process until it exits the grid. If the particle exits the top of the grid, the particle scores 1 point for the walk. If the particle exits the left, bottom or right sides of the grid, the particle scores 0 points for the walk. In this project, you will compute the expected score of the particle as a function of the starting coordinates.

Hint: For each starting point, you may estimate the expected score of the particle by simulating many, many random paths and taking their average score.

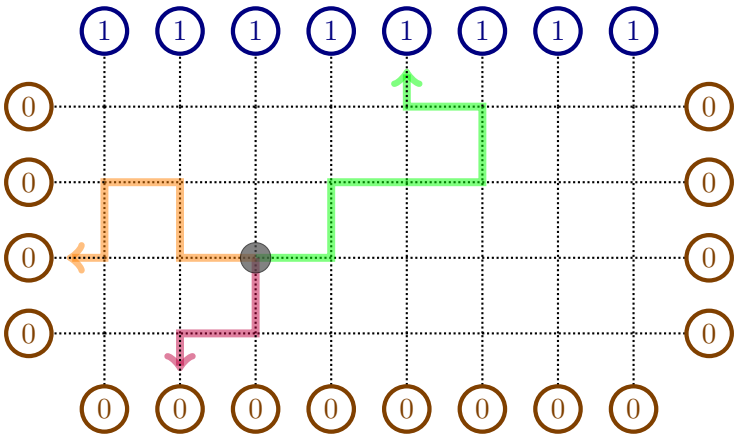


Illustration: The figure shows three sample paths of the random walk that all begin at the initial point (3,2). For the red path, the particle (randomly) takes the steps $(\downarrow, \leftarrow, \downarrow)$, and exits the grid at the bottom scoring zero points for the walk. The other paths are the results of the steps:

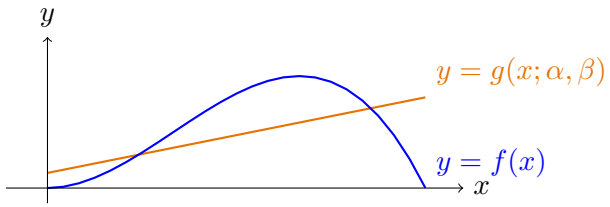
Steps: $(\downarrow \leftarrow \downarrow)$	(See red path)	Exit: Bottom	Score: 0
Steps: $(\leftarrow \uparrow \leftarrow \downarrow \leftarrow)$	(See orange path)	Exit: Left	Score: 0
Steps: $(\rightarrow \uparrow \rightarrow \rightarrow \uparrow \leftarrow \uparrow)$	(See green path)	Exit: Top	Score: 1

Bonus: In this project, we have computed the solution to a problem by simulating many random events. This 'indirect' approach is always computationally inefficient, so it is often worth looking for a 'direct' approach. For this problem, there *is* a method of computing the solution directly. Can you find it?

7 Bias and Variance for Learning in 1-D

In this project, you will compare the difference between *approximation* (finding the best possible candidate to model a function) and *learning* (finding the best possible candidate given limited information). You will also analyze the extra error that is incurred by using limited information, and you will explore one possible way of reducing this error.

Approximation:
Let $f(x) = x^2(1-x)$ where $0 \leq x \leq 1$. We wish to find the 'best' linear model (or approximation) for f. That is, for functions in the form $g(x;\alpha,\beta) = \alpha + \beta x$, we wish to find the parameters α and β that minimize the error we would expect to incur if we used the function g to model (or approximate) the function f.



For this problem, we will define the error as

$$\text{Error}(x;\alpha,\beta) = \Big(f(x) - g(x;\alpha,\beta)\Big)^2 \quad \text{and} \quad \text{TotalError}(\alpha,\beta) = \int_0^1 \text{Error}(x;\alpha,\beta)dx.$$

Use pencil-and-paper (or Mathematica, WolframAlpha, etc) to find a simplified expression for the error between f and g as a function of the parameters α and β. Find the values of α and β that minimize the total error and find the corresponding total error.

Learning:
In many practical applications, we only have imperfect knowledge of f, and therefore it is impossible to know the *best possible* parameters α, β. Often, we must commit to a choice of sub-optimal parameters that are chosen after only a small glimpse of f. In this project, you will estimate and explore the extra error incurred when using an imperfect choice of parameters.
Terminology: For each value of x, the *bias* of a model is defined as the the difference between $\bar{g}(x;\alpha,\beta)$, the expected (or average) value of $g(x;\alpha,\beta)$, and the true value of f.

$$\text{Bias}(x) = \bar{g}(x;\alpha,\beta) - f(x) \tag{7.1}$$

Terminology: For each value of x, the *variance* of a model is defined as the expected (or average) value of the square of the difference between the value of $g(s;\alpha,\beta)$ and the expected

value of $g(x;\alpha,\beta)$.

$$\text{Variance}(x) = \left\langle \big(g(x;\alpha,\beta) - \bar{g}(x;\alpha,\beta)\big)^2 \right\rangle \tag{7.2}$$

Implement the following process to estimate how well a linear function that is parameterized by only two random data points can be used to model f.

1. Create the variable `xx` by discretizing the interval $0 \le x \le 1$ into 101 points. That is, set `xx`= (0, 0.01, 0.02, ..., 1). Plot $f(x)$ in blue.
2. Generate two random data points on $x_1, x_2 \in [0, 1]$.
3. Evaluate $y_1 = f(x_1)$ and $y_2 = f(x_2)$. Plot (x_1, y_1) and (x_2, y_2) as two orange dots on the same figure as f.
4. Using **only** (x_1, y_1) and (x_2, y_2), compute a 'best guess' for parameters α and β.
5. Use your best guess parameters α and β to evaluate the linear model $g(x;\alpha,\beta)$ for each $x \in$ `xx` and record the results. Plot $g(x;\alpha,\beta)$ in orange on the same figure.

Repeat steps 1-5 several times and save their plots.

Repeat steps 1-5 1000 times (without plotting) and record the results in a 101×1000 array. For each x, compute $\bar{g}(x)$, the average value predicted by the linear model $g(x;\alpha,\beta)$ and plot it on the same figure as a plot of $f(x)$. On the same figure, plot $g_5(x)$, $g_{25}(x)$, $g_{75}(x)$ and $g_{95}(x)$, the 5^{th}, 25^{th}, 75^{th} and 95^{th} quantiles of the values predicted by the linear model.

Error Reduction for Learning:
Someone proposes that the variance is so big that it makes the linear model unusable. They suggest that we should approximate f by a constant function $h(x;\alpha)$. Repeat the steps above to estimate the bias and the variance of the constant model.

8 Random Triangles in High Dimensions

For a triangle Δ, we will let $\mathcal{Q}$ be a measure of the *quality* of the triangle as defined by

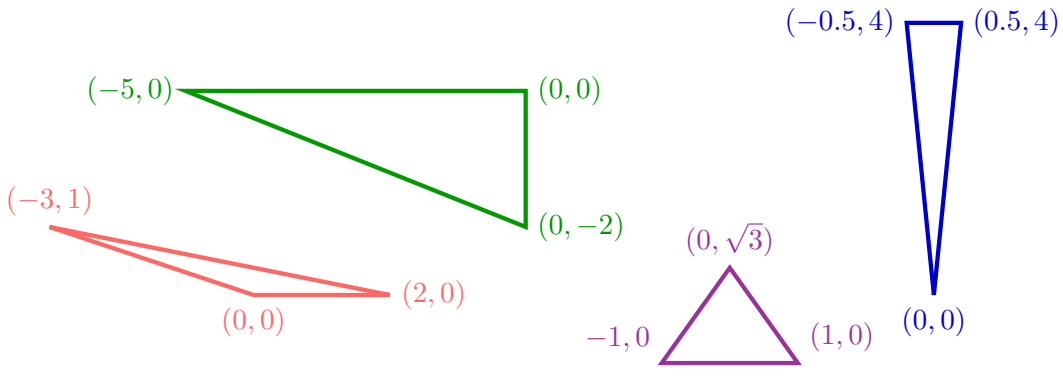
$$\mathcal{Q}(\Delta) = \frac{4\sqrt{3}A}{\ell_1^2 + \ell_2^2 + \ell_3^2} \tag{8.1}$$

where A is the area of the triangle, and ℓ_1, ℓ_2 and ℓ_3 are the lengths of the three sides. Without performing any computations, can you determine the maximum and minimum values that $\mathcal{Q}$ can take? Can you describe what kinds of triangles are associated with high, and with low, values of $\mathcal{Q}$?

In the first part of this project, you will write a function that computes the quality of a triangle in $\mathbb{R}^2$, given the coordinates of its vertices.

- Write a function, `tri_lengths()`, that takes as its arguments the three vertices of a triangle and returns the lengths of the three sides of the triangle.
- Write a function, `tri_angles()`, that takes as its arguments the three vertices of a triangle and returns the angles of the triangle.
- Write a function, `tri_area()`, that takes as its arguments the three vertices of a triangle and returns the area of the triangle.
- Write a function, `tri_score()`, that takes as its arguments the three vertices of a triangle and returns the quality score of the triangle as defined in equation (8.1).
- Test your function `tri_score()` on the four triangles below. As a check on your function, the correct values are:

$$\mathcal{Q}(\Delta) = 0.597\cdots, \qquad \mathcal{Q}(\Delta) = 0.413\cdots,$$
$$\mathcal{Q}(\Delta) = 0.172\cdots, \qquad \mathcal{Q}(\Delta) = 1.$$



Many applied mathematicians regularly work with data sets where each data point is associated with many different features (e.g. each medical patient is associated with their own temperature, heart rate, blood pressure, etc). Sometimes the physical, three-dimensional world can provide useful intuition for the geometry of higher dimensional space. Most of the time, however, we are unable to grasp all the weird ways that the higher dimensional space is 'bigger'.

In the second part of this project, you will compare the expected quality of random triangles in $\mathbb{R}^2$ with that of random triangles in $\mathbb{R}^{10}$.

(a) i. Use a random number generator to create a random point, $\boldsymbol{x} = (x_1, x_2)$, that is normally distributed in $\mathbb{R}^2$. The functions `randn()` (Matlab), `np.random.randn()` (Python), or `randn()` (Julia) may be useful.

ii. Repeat part i. until you have three points, $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{y} = (y_1, y_2)$, and $\boldsymbol{z} = (z_1, z_2)$. Let Δ be the triangle with $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ as its vertices. Use your function `tri_score()` to compute $\mathcal{Q}(\Delta)$.

iii. Repeat parts i. and ii. until you have the quality scores for 100,000 random triangles. Plot a histogram of your data.

(b) Repeat this process for triangles in $\mathbb{R}^{10}$. That is, use a multivariate normal random number generator to generate 3 points in $\mathbb{R}^{10}$, $\boldsymbol{x} = (x_1, x_2, \ldots x_{10})$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_{10})$, $\boldsymbol{z} = (z_1, z_2, \ldots, z_{10})$. Let Δ be the triangle with $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ as its vertices, and compute $\mathcal{Q}(\Delta)$. Repeat for 100,000 random triangles.

(c) Compare the histogram for triangles in $\mathbb{R}^2$ with a histogram for triangles in $\mathbb{R}^{10}$. Provide a geometric explanation for why the random triangles in higher dimensions are much closer to equilateral than the random triangles in lower dimensions