Reconstructing the Tracy-Widom Distribution

Connor Moore, 100826701. April 26, 2026

1 Introduction

Random numbers form the basis of various studies in mathematics and the sciences. One strong example of this is *Random Matrix Theory*, or the study of the construction and properties of random matrices. One of the first applications of random matrix theory was proposed by Eugene Wigner (1902-1995) in his application to model the nuclei and spectrum of various heavy atom [1, 2]. It has since found different applications including work in computational mechanics, control theory, systems engineering, and others.

Random matrix theory borrows some distributions from statistics to construct matrices. This includes the Gaussian (normal) distribution, which is used to make various matrices depending on which number system is used. Wigner's work included Orthogonal (real symmetric), Unitary (Complex Hermitian), and Symplectic (Quaternion Hermitian) matrices and their eigenvalues.

This work considers Gaussian orthogonal ensembles (GOEs) specifically and attempts to reconstruct the Tracy-Widom distribution directly by computing eigenvalues. An overview of relevant theory, the necessary methods, and implementation details is provided before results are presented. The distribution is examined as a function of matrix dimension and sample size, and statistical testing is performed to support a Tracy-Widom being observed as opposed to a normal distribution.

2 Relevant Theory

2.1 Random Ensembles

The GOE-based random matrix has some properties that go beyond simply being normally distributed. Namely, the matrix is square, symmetric, and the standard deviation σ is unity except for elements on the diagonal where $\sigma = 2$. The mean μ is considered as zero for all elements. Consider the normal distribution with notation $N(\mu, \sigma)$. Then, the matrix is constructed using:

$$\mathbf{A}(i,j) \sim \begin{cases} N(0,1) & \forall (i \neq j) \\ N(0,2) & \forall (i = j) \end{cases}$$

The diagonal standard deviation of 2 is what makes this ensemble orthogonal. Note that the representation above does not explicitly state the symmetry, but $\mathbf{A}(i,j) = \mathbf{A}(j,i) \, \forall (i,j)$.

2.2 The Tracy-Widom Distributions

Various distributions can be reconstructed from the eigenvalues of a GOE. For example, the case of the global eigenvalues yields a semicircle distribution [1]. Consider the N eigenvalues of a vector $|\lambda_1| \geq |\lambda_2| \geq |\lambda_3\| \geq ... \geq |\lambda_N|$. In this case, λ_1 is considered the *dominant* eigenvalue so long as $|\lambda_1| > |\lambda_2|$. Taking the distribution of the dominant eigenvalues λ_1 yields an entirely different result that is known as the *Tracy-Widom* distribution. This distribution can be parametrized in terms of which division algebra is used in its definition. Wigner's work considered real, complex, and Quaternion cases, shown below in Figure 1:

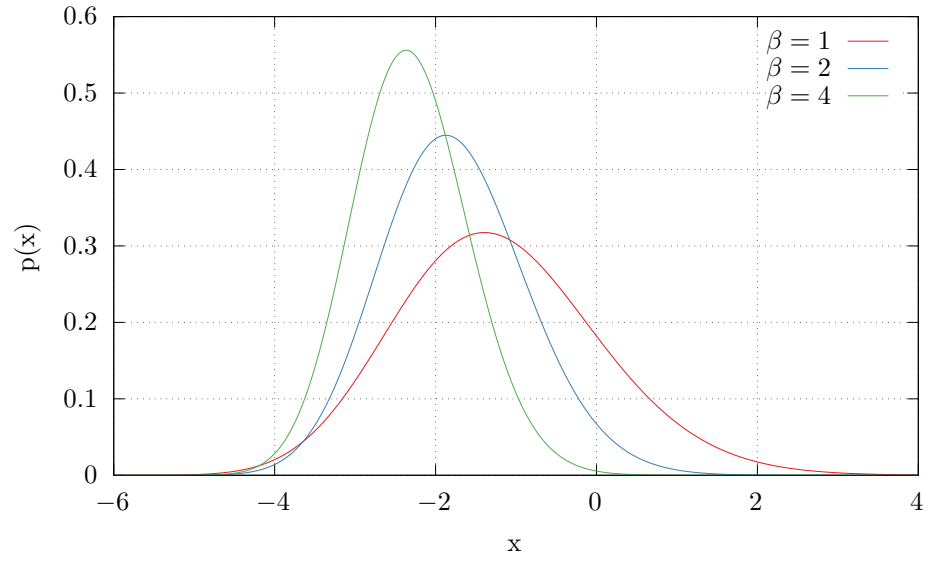

Figure 1: Various Tracy-Widom distributions for $\beta = 1, 2, 4$. The different values of β depend on which division algebra is used for the Gaussian ensemble (real/complex/quaternion). This work considers only the TW_1 distribution defined for real numbers, shown in red as the shallowest peak compared with the complex and quaternion versions.

Compared with a normal distribution, it is also difficult to distinguish between the two visually. Figure 2 shows a comparison between a Tracy-Widom and a normal distribution:

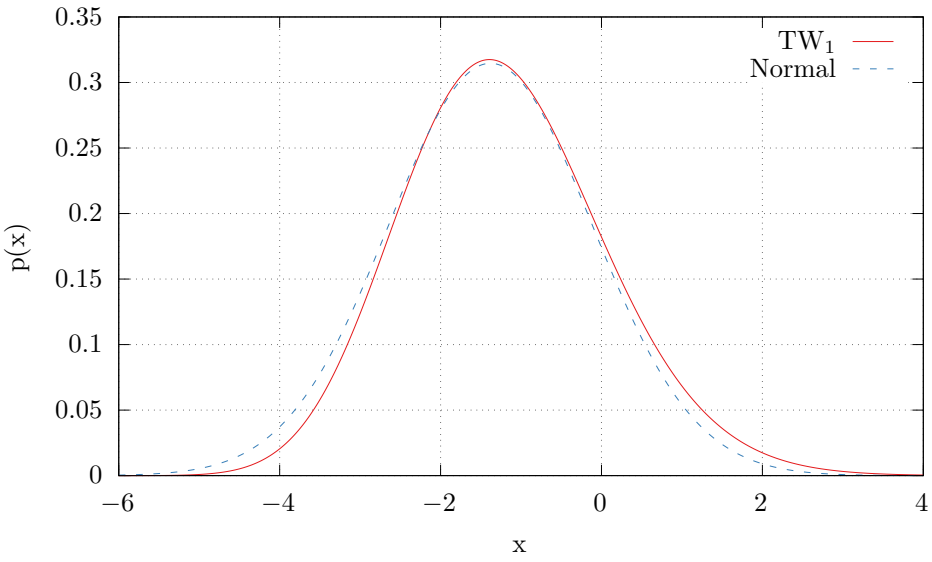

Figure 2: The TW_1 distribution compared with a normal distribution of arbitrary mean and standard deviation. Note that while the peaks are centered, the Tracy-Widom has a 'fatter' tail on the $+x$ side and a slimmer one on the $-x$ side. Both distributions are quite similar visually, so this asymmetry will play an important role in quantifying their differences.

Note that the Tracy-Widoms plotted above have no closed-form expression and must be approximated, typically by using a scaled Gamma function [3], or more cleverly by downloading some else's finished ap-

proximation off of the MATLAB file exchange [4]. Because the distributions are for visual purposes only, a smart approach to validating the generation of a Tracy-Widom should include some type of quantifiable, statistical assurance not based on the simple value of the distribution itself.

2.3 Statistical Differentiation

There are various ways to go about discerning a Tracy-Widom distribution from a normal one. Perhaps the most elegant is to exploit some basic differences in properties between the two. As mentioned earlier, the Tracy-Widom has a fatter side and a slimmer side, which makes it asymmetric. This is not the case with the normal distribution which is identical on either side of the mean value μ. It turns out that the symmetry of a distribution is readily quantifiable using existing statistical tools, and as such that will be the backbone of the statistical analysis. First consider the *expected value* of a probability density function (PDF) $p(x)$, defined as [5]:

$$E[x] = \int_s xp(x)\,dx \approx \left(\sum_i x_i\,p(i)\right)\Delta x \tag{1}$$

where the approximation applies for a discrete PDF with equi-distance bins along x. The expected value has *moments* that give quantifiable insights into the behaviour and shape of a PDF. The general equation for the nth central moment of $E(x)$ is given as:

$$E\left[(x-a_n)^n\right] \tag{2}$$

where a_n is some constant. The first moment (taken as a so-called *raw moment* with $a = 0$) simply reduces to 1. The second moment is the variance of the function and is taken with $a = \mu$:

$$\sigma^2 = E\left[(x - E[x])^2\right] = E\left[(x-\mu)^2\right] \tag{3}$$

The next two moments are of particular value for this project as they quantify the symmetry of the distribution. The third moment is the *skewness* of the distribution, which deviates from the central moments above and is normalized by the standard deviation σ:

$$\text{Skewness} = E\left[\frac{(x-\mu)^3}{\sigma}\right] \tag{4}$$

Lastly, the 4th moment is the kurtosis of the distribution. Kurtosis comes from the Greek word for *bulging*, and as a parameter it quantifies the 'sharpness' of the PDF. A larger value of kurtosis implies that the distribution has fatter tails [5]. Note that kurtosis is always a positive value and is equal to 3 for a standard normal distribution. Therefore, the *excess* kurtosis is considered by subtracting 3. The relation is given as:

$$\text{Excess Kurtosis} = E\left[\frac{(x-\mu)^4}{\sigma}\right] - 3 \tag{5}$$

These tests, while perhaps simple, will do a fine job in the strict context of differentiating a Tracy-Widom distribution from a normal distribution. In the case of the normal distribution, the skewness and excess kurtosis are both zero. For comparison, the approximate Tracy-Widom has a skewness of ≈ 0.2935 and an excess kurtosis of ≈ 0.1292.This was calculated using an Octave script[1] that sampled 50,000 bins in the range $[-9, 9]$ for both distributions. Performing calculations with (1,3,4,5) reconstructed the mean and standard deviation for the normal distribution sampled from its the closed-form equations down to an error of $\sim 10^{-5}$.

3 Numerical Methods

The natural extension of defining the prerequisite statistical tests to confirm a Tracy-Widom is to actually sample a Tracy-Widom. The issue with this, however, is that it can be quite computationally demanding. The Tracy-Widom is quite similar to the normal except for near the very tails, meaning that most samples

[1] Available online as part of the Git repository for the project.

(occurring near the peaks) will show agreement between the two. Additionally, it is theorized that the dimension of the matrix N has to be on the order of thousands to get a very Tracy-Widom distribution. The combination of these two points motivates finding a convenient way to calculate the maximum eigenvalue of a large matrix repeated such that thousands of calculations can be performed in a meaningfully short amount of time.

3.1 The Power Method

One relatively cheap way to find the maximum eigenvalue is through the so-called power method. The power method is an iterative approach to finding the maximum eigenvalue of a matrix that exploits the spectral gap ($\lambda_1 - \lambda_2$). Consider an initial guess in the form of a vector of length N given as $\mathbf{x_0}$. Then, perform the multiplication:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k \tag{6}$$

After multiplying, then normalize the vector such that $|\mathbf{x}_{k+1}|$ is unity:

$$\mathbf{x}_{k+1} = \frac{\mathbf{x}_{k+1}}{|\mathbf{x}_{k+1}|} \tag{7}$$

Or, more compactly written as one step:

$$\mathbf{x}_{k+1} = \frac{\mathbf{A}\mathbf{x}_k}{|\mathbf{A}\mathbf{x}_k|} \tag{8}$$

Repeating this will converge to the dominant eigenvalue λ_1 so long as $\mathbf{x}_0$ is not orthogonal to the dominant eigenvector [6]. The convergence of the method is $\mathcal{O}\left(|\lambda_1/\lambda_2|^k\right)$ and is largely dependent on the spectral gap. The main advantage of this method is that computationally it consists of only a matrix multiplication ($\mathcal{O}(n^2)$) and normalization ($\mathcal{O}(n)$) which is rather inexpensive compared with other methods per iteration [7]. Worth noting, however, is that in cases where $\mathbf{A}$ has a small spectral gap it is not guaranteed to converge in an acceptable number of iterations. This poses an issue as it would bias the calculations for reconstructing the Tracy-Widom by not including random ensembles specifically with lower spectral gaps. To ratify this, more expensive methods exist which guarantee a solution that are used as a fallback.

3.2 Schur Decomposition

One method that guarantees a solution $\forall\mathbf{A}$ is Schur Decomposition. Named after the late Russian mathematician Issai Schur (1875-1941), Schur Decomposition involves decomposing the matrix $\mathbf{A}$ into a combination of some unitary matrix $\mathbf{U}$ and upper-triangular matrix $\mathbf{T}$ [7]:

$$\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^{\mathrm{T}} \tag{9}$$

In this case, the diagonal entries in $\mathbf{T}$ are then the eigenvalues of the original matrix $\mathbf{A}$. The proof in Schur's theorem is non-constructive and does not provide a solution without knowing the eigenvectors of $\mathbf{A}$, but it can be still be used in combination with the so-called QR algorithm developed independently by John Francis (1934-present) and Vera Kublanovskaya (1920-2012) starting from 1959 [8]. Essentially, the QR algorithm is the computational scheme that provides the matrices $\mathbf{U}$ and $\mathbf{T}$ from the input $\mathbf{A}$. It does so with a complexity of $\mathcal{O}(n^3)$, which for larger values of N deviates significantly from the $\mathcal{O}(n^2)$ of the power method. For this reason it is used only when the power method fails.

4 Implementation and MPI Parallelization

The last remaining hurdle to computing the eigenvalues is the magnitude of memory required to store such large matrices. Because the requirements will be in the gigabytes range for most calculations, shared-memory parallelism can have issues supporting large-scale calculations needed to accurately define the Tracy-Widom. Because of this, the solution is designed around MPI parallelism with distributed and independent memory between ranks. There are fewer opportunities for race conditions and threads 'overstepping' each other in MPI, however care must be taken that the now independent workers can communicate with each other properly.

4.1 Worker-Manager Structure

The program used in this project employed a worker-manager structure to drive the eigenvalue calculations. The first worker is assigned to be a manager and handle the seeding and results collection for the other workers. Everyone else is focused solely on calculating eigenvalues and reporting their results to the manager when finished. This setup allows for the independent calculation of as many eigenvalues as MPI ranks are supported at once, which can approach the order of hundreds or even thousands if compute resources are available on a respectable cluster.

4.2 The Manager Process

The manager process is relatively straightforward and focuses more on the 'book keeping' side of the eigenvalue calculations. To demonstrate this, a pseudocode explaining the behaviour of the manager MPI rank is presented below in Algorithm 1:

Algorithm 1 MPI Manager Process

```
N,L ← Matrix size and number of data points
Open logfile and write N, L for reference
while Workers not finished do
    for i = 1 : N_workers do
        if Worker i is ready and needed then
            seed ← /dev/urandom
            MPI Broadcast: send the worker seed
        end if
    end for
    if All workers are finished then
        Break loop and finish managing
    end if
    MPI Receive: wait for a worker to finish calculating...
    Identify which worker is finished calculating and their result
    if Eigenvalue was calculated successfully then
        Tally to the number of successful eigenvalue calculations
        Write the worker number and eigenvalue to logfile
    else
        Tally to the number of failed eigenvalue calculations
        Print a warning message
    end if
    if Worker is still needed then
        Set worker flag to 'Ready'
    else
        Tell worker to conclude
    end if
end while
```

Note that the actions labelled with involving MPI communication are blocking, i.e., the worker/manager will wait to receive a broadcast before continuing at all.

4.3 The Calculation (Worker) Process

The worker process involves calculating the eigenvalues and is more complicated than the manager process. Various sub-processes are included in different algorithms, namely for building the matrix, computing the eigenvalue, and applying the power method. The overview of the whole process is outlined in Algorithm 2:

Algorithm 2 MPI Worker Process

size ← `MPI Receive`: matrix size
Initialize matrix space in memory
while Worker not Finished **do**
    seed ← `MPI Receive`: new matrix seed
    Call `open_prng_seed`(seed)
    matrix ← `build_random_matrix`(size)
    eig ← `calculate_eigenvalue`(matrix)
    `MPI Send`: eig and relevant information
end while

The algorithm for assembling the random matrix given a size is provided below in Algorithm 3:

Algorithm 3 MPI Worker: Random Matrix Building

procedure BUILD_RANDOM_MATRIX(size)
    for $i = 1 : \text{size}$ **do**
        $\mathbf{A}$(i,i) = N(0,2)
        for $j = 1 : i - 1$ **do**
            $\mathbf{A}$(i,j) = N(0,1)
            $\mathbf{A}$(j,i) = $\mathbf{A}$(i,j)
        end for
    end for
    return A
end procedure

And likewise for the eigenvalue calculation in Algorithm 4:

Algorithm 4 MPI Worker: Eigenvalue Calculation

procedure CALCULATE_EIGENVALUE(matrix)
    Initialize convergence tolerance ε and maximum iterations for power iteration
    Call `power_iteration`(matrix, shift=0)
    if power iteration converged **then**
        if $\lambda < 0$ **then**
            Call `power_iteration`(matrix, shift=λ)
        end if
    else
        call `MKL_schur_decomposition`(matrix)
        if decomposition successful **then**
            eig ← Largest eigenvalue from decomposition
            Set 'converged' flag to 'true'
        else
            Print warning message ▷ Not touching the converged flag leaves it 'false'
        end if
    end if
end procedure

So far the structure has involved a minimal amount of math. This is only because it has been hidden behind abstraction and by using the MKL library to call pre-made functions for computing eigenvalues. The power iteration method was implemented by hand and involves more math as shown in Algorithm 5:

Algorithm 5 MPI Worker: Power Iteration

procedure POWER_ITERATION(matrix, shift=0)
  Initialize a random vector $\mathbf{x}_o$ = N(0,1)
  $\mathbf{x}_0 \leftarrow \frac{\mathbf{x}_0}{|\mathbf{x}_0|}$ ▷ Normalize
  mem $\leftarrow -100$
  for $i = 1$: max_iterations **do**
    $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}_{i-1}$
    $\mathbf{y} \leftarrow \mathbf{y}$+shift$\times\mathbf{x}_{i-1}$
    eig $\leftarrow$ `dot_product`($\mathbf{x}_i$,$\mathbf{x}_{i-1}$) - shift
    error $\leftarrow \frac{|\text{mem}-\text{eig}|}{|\text{mem}|}$
    if err $< \varepsilon$ **then**
      Set 'converged' flag to 'true'
      return eig
    else
      $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{|\mathbf{x}_i|}$
      mem $\leftarrow$ eig
    end if
  end for
end procedure

These algorithms compose the entirety of the program for reconstructing the Tracy-Widom distributions discussed earlier. It is worth noting that the expected complexity is a function of the required calculation complexity n, the number of MPI ranks used m, and the sample size l. Assuming a total FLOPs count of $\sim \mathcal{O}(n^3)$ in a worst-case scenario for Schur decomposition, repeated l times, in reality each rank will only have to contribute 1-mth of the work. Note that while the QR algorithm is not done in parallel, repeating in l times is embarrassingly parallel. So the total time complexity should be roughly:

$$\sim \mathcal{O}\left(\frac{n^3\, l}{m}\right) \tag{10}$$

which will be tested in the next section.

5 Results

Before the eigenvalues can be directly compared to a Tracy-Widom they must be scaled to match the existing distribution. It is possible to perform the scaling empirically, but a closed-form solution is available that makes things easier [9]:

$$\lambda' = \left(\lambda - 2\sqrt{N}\right) \times N^{1/6} \tag{11}$$

This specific transform is derived from the Wigner semicircular distribution mentioned earlier in section 2. Although it describes the global distribution, taking the extrema yields a bound for the maximum eigenvalue.

5.1 Tracy-Widom Convergence

Following some software issues related to the OneAPI suite, the author was unable to produce enough samples in time to complete this report. However, a moment is appropriate to thank all of the teammates for being kind enough to share their data for analysis in this report. The following numbers are from Joe's runs but the analysis was done individually. Samples here are presented with respect to their matrix dimension N, number of samples L, skewness S and excess kurtosis K. Taking, for example, different sample sizes for $N = 1750, 500$ yields a half-dozen examples of a clear trend emerging where the histograms tend towards a normal-*ish* shape, shown in Figure 3:

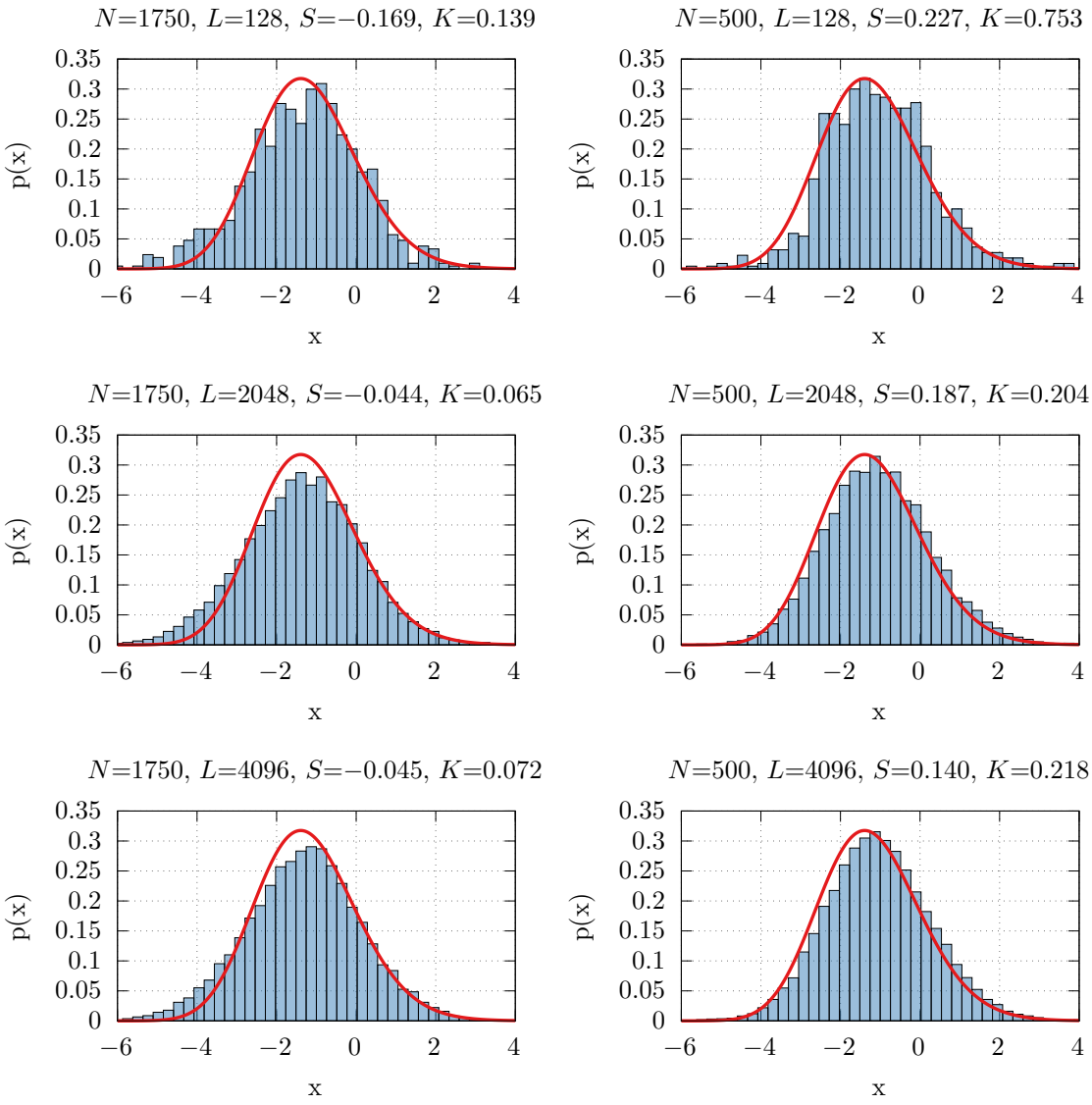

Figure 3: Progression of the Tracy-Widom convergence as a function of matrix size (N) and sample size (L). Also included are the skewness (S) and excess kurtosis (K) for each sample. An increase in both sample size and matrix size yields a distribution comparable with but not necessarily closer to the Tracy-Widom reference. Asymmetry is noted especially for the cases with $N = 1750$, but with an incorrect sign that implies a fatter tail in the $-x$ direction.

Despite the visuals, however, the metrics tell a bleaker story. Recall that the Tracy-Widom is authentically constructed with a fatter tail on the right and thinner on the left, which seems to be the opposite of what is presented above. This is also reflected in the skewness assuming a negative value for the curves associated with $N = 1750$, implying that the distribution is more favourable to the left of the mean value. As the sample size increases, the shape of the distribution becomes more refined and tends to that of the expected profile, which seems intuitive. It is possible to consider the maximum number of samples available for each matrix size N to confirm whether or not this can yield better results, shown in Figure 4:

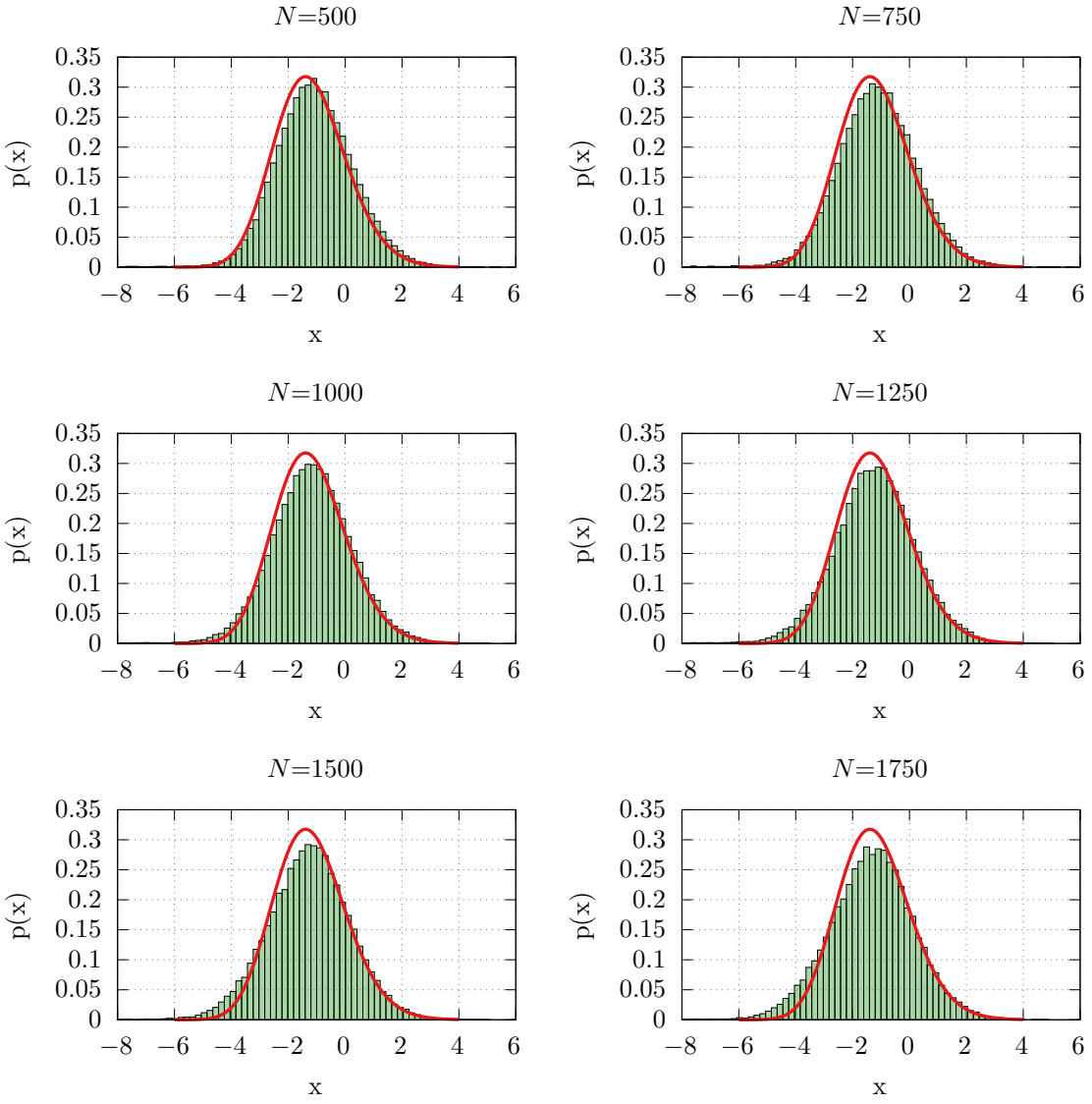

Figure 4: Reconstructed distributions from the maximum number of samples L_{max} for each size N. Table 1 provides more information on the behaviour of skewness and excess kurtosis for the high-sample runs. Qualitatively, under filling seems prevalent at larger N and a strong $-x$ tail is visible.

The parameters calculated for the distributions are presented in Table 1 for convenience:

Table 1: Parameters of the Larger-L Distributions for Various N

N	L	Skewness	Excess Kurtosis
500	50,104	-8.993×10^{-3}	8.588×10^{-1}
750	55,397	-5.745×10^{-2}	7.666×10^{-1}
1,000	55,536	-9.220×10^{-2}	6.924×10^{-1}
1,250	55,726	-1.185×10^{-1}	6.616×10^{-1}
1,500	55,948	-1.188×10^{-1}	4.567×10^{-1}
1,750	56,097	-1.471×10^{-1}	4.227×10^{-1}

Despite the significant increase in sample size ($\sim 10\times$ at a minimum), the skewness and excess kurtosis do not converge to their expected values. Rather, the skewness diverges from the expected value of 0.2935, and the excess kurtosis has a minimum error of at least 200% relative to the expected 0.1292 described earlier. These trends are shown below in Figure 5:

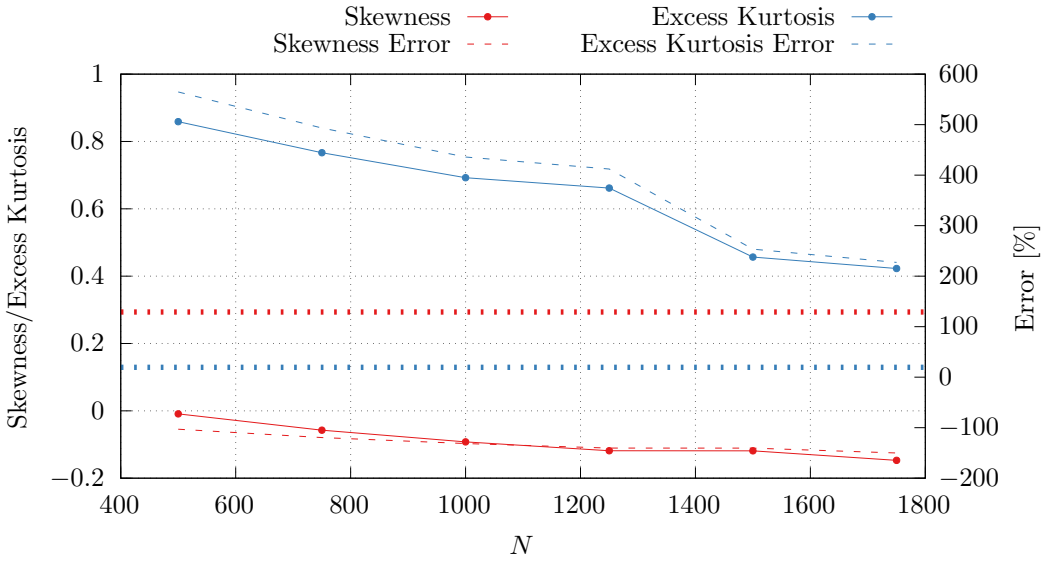

Figure 5: Behaviour of the two parameters skewness S and excess kurtosis K as function of matrix size N for a maximal number of samples L. Observed responses are shown in solid lines, with the expected values being horizontal dotted lines and the percent-error represented with dashed lines. Neither of the trends observed in the skewness or expected kurtosis are consistent with theory, and their behaviours cannot confirm the existence of a Tracy-Widom distribution from the calculations.

The tendency of the skewness to diverge from zero as the matrix size increases does point to a non-Gaussian distribution being observed, but it cannot be concluded that it is a Tracy-Widom.

5.2 Wall Times

Lastly, the wall time of running the simulations was investigated. These results were more aligned with the expected values of complexity. Two different perspectives are considered: the variation of sample size with a constant matrix size, and the vice-versa case. The results for the constant sample sizes is discussed first as it should correspond to the complexity mentioned in Section 4. Figure 6 shows a complexity tending to $\mathcal{O}(n^3)$ (with slight variations over the matrix size domain) for all sample sizes:

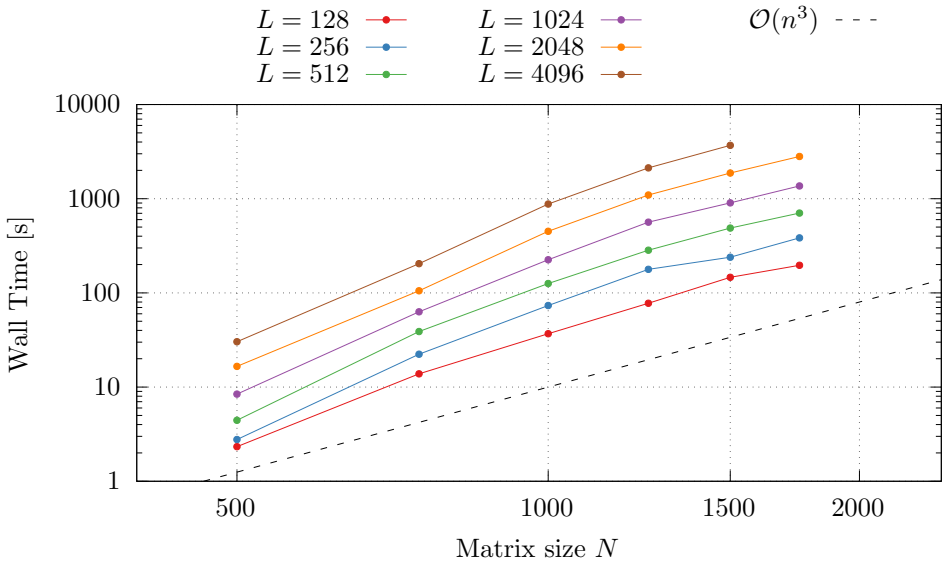

Figure 6: Wall times for different sample sizes L as a function of the matrix size N. All runs were conducted using 128 MPI ranks for this figure. Wall times show a roughly $\mathcal{O}(n^3)$ scaling consistent between all sample sizes tested, although there is a magnitude difference between the fastest ($L = 128$) and slowest ($L = 4096$) runs.

The vice-versa case shows a different story, with a complexity closer to $\mathcal{O}(n)$. This makes intuitive sense, as MPI ranks can calculate different matrices of a common sample in parallel, explaining the rather weak coupling. Compare this to increasing the matrix size itself, which is solved in serial by one rank at time, making the scaling appear much more sensitive to changes in the independent variable. The trend is visualized below in Figure 7:

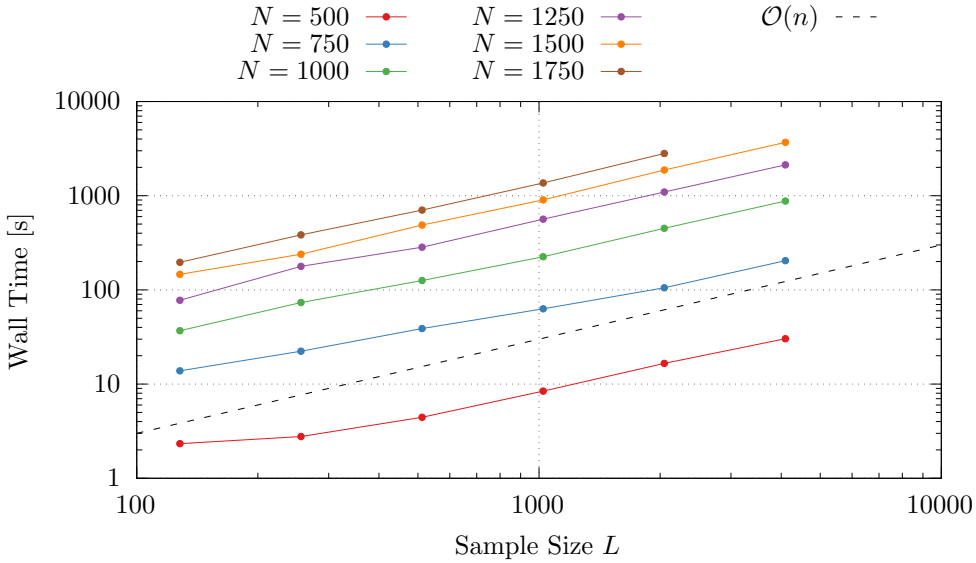

Figure 7: Wall times for different matrix sizes N as a function of the number of sampled eigenvalues L. All runs were conducted using 128 MPI ranks for this figure. Wall times show a roughly $\mathcal{O}(n)$ scaling consistent between all matrix sizes tested, although there was a ~ 2 magnitude difference between the fastest ($N = 1$) and slowest ($N = 1750$) runs. Small deviations exist for low sample sizes.

6 Conclusions

This work set out to reconstruct a Tracy-Widom distribution via the direct sampling of calculated eigenvalues of Gaussian orthogonal ensembles. A code leveraging MPI for distributed-memory parallelism and the MKL BLAS library for high-performance numerical methods was used to sample various numbers of ensembles, sizes of matrices, and various other factors to generate data and draw conclusions. While the worker-manager structure worked well for the batch computing of eigenvalues, the statistical results proved inconclusive. It can be said with confidence that the resulting distribution was non-Gaussian but it was not necessarily a Tracy-Widom distribution. This is because of the statistical testing, consisting of the skewness and excess kurtosis, were not in support of a Tracy-Widom distribution. The discrepancies showed at least one trend divergent to the expected value and therefore it is unlikely that further sampling of eigenvalues would mitigate this issue. Rather, it is likely that there is an inconsistency related to transforming the eigenvalues to fit the standard Tracy-Widom. The wall time scaling, however, was consistent with expectations and was invested with respect to both matrix and ensemble size.

References

[1] G. W. Anderson, A. Guionnet, and O. Zeitouni, *An Introduction to Random Matrices*. Cambridge University Press, February 2009. Chapters 1 and 2.

[2] E. P. Wigner, "Characteristic Vectors of Bordered Matrices With Infinite Dimensions," *The Annals of Mathematics*, vol. 62, pp. 548–564, November 1955. Available online.

[3] M. Chiani, "Distribution of the Largest Eigenvalue for Real Wishart and Gaussian Random Matrices and a Simple Approximation for the Tracy-Widom Distribution," *Journal of Multivariate Analysis*, vol. 129, pp. 69–81, April 2014. Available online.

[4] Marco, "Approximation for the Tracy-Widom Laws." Published to the MATLAB Central File Exchange, 2013. Retrieved April 25th, 2026. Available online.

[5] K. Siegrist, *Probability Theory Probability, Mathematical Statistics, and Stochastic Processes (2021 Edition)*. Open Education Resource LibreTexts, 2021. Chapter 4. Available online.

[6] K. A. Novak, *Numerical Methods for Scientific Computing*. Sharon, Vermont: Equal Share Press, March 2022. Chapter 4. Available online.

[7] D. S. Watkins, *Fundamenals of Matrix Computations (2nd Edition)*. New York: John Wiley and Sons, January 2002. Chapter 5.

[8] G. Golub and F. Uhlig, "The QR Algorithm: 50 Years Later its Genesis by John Francis and Vera Kublanovskaya and Subsequent Developments," *IMA Journal of Numerical Analysis*, vol. 29, pp. 467–485, June 2009. Available online.

[9] W. Konig, "Orthogonal Polynomial Ensembles in Probability Theory," *Probability Surveys*, vol. 2, pp. 385–447, December 2005. Available online.