Computing the Cumulative Distribution Function and Quantiles of the limit of the Two-sided Kolmogorov-Smirnov Statistic

Paul van Mulbregt

pvanmulbregt@alum.mit.edu

March 2, 2018

Abstract

The cumulative distribution and quantile functions for the two-sided one sample Kolmogorov-Smirnov probability distributions are used for goodness-of-fit testing. The CDF is notoriously difficult to explicitly describe and to compute, and for large sample size use of the limiting distribution is an attractive alternative, with its lower computational requirements. No closed form solution for the computation of the quantiles is known. Computing the quantile function by a numeric root-finder for any specific probability may require multiple evaluations of both the CDF and its derivative. Approximations to both the CDF and its derivative can be used to reduce the computational demands. We show that the approximations in use inside the open source SciPy python software result in increased computation, not just reduced accuracy, and cause convergence failures in the root-finding. Then we provide alternate algorithms which restore accuracy and efficiency across the whole domain.

Keywords: Two-sided Kolmogorov-Smirnov, probability, computation, approximations

1 Introduction

The Kolmogorov-Smirnov statistics D_n, D_n^+, D_n^- are statistics that can be used as a measure of the goodness-of-fit between a sample of size n and a target probability distribution. Computation of the exact probability distribution for these statistics is a little complicated, but Kolmogorov and Smirnov showed that they had a certain limiting behaviour as $n \to \infty$.

To be used as part of a statistical test, either the value of the Survival Function (SF) (or its complement the CDF) needs to computable for a given value of D_n/D_n^+, or values need to be known corresponding to the desired critical probabilities (e.g. $p = 0.1, 0.01, \ldots$). The quantile functions associated with these distributions can be used to generate a table of critical values, but they can also be used to generate random

1

variates for the distribution, and have also found applications to rescaling of the axes in some types of graphing applications.

For the one-sided D_n^+, a formula is known which can be used to compute the SF. But a simple formula for the two-sided D_n is unknown. The quantile functions do not have a closed-form solution, hence need to be calculated either by interpolating some known values, or by a numerical root-finding approach. It is here especially that approximations may be used to reduce the computational requirements.

The Python package SciPy (v0.19.1) [1] provides the `scipy.stats.kstwobign` class for the distribution of the two-sided $\lim_{n\to\infty} \sqrt{n}D_n$, which in turn make calls to the "C" library `scipy.special`, to calculate both the SF and its inverse, the ISF.

An analysis of the approximations used in the algorithms determined that these approximations were only valid in a subset of the domain, resulting in loss of accuracy and/or increase in computation. Several causes of root-finding failure are identified. Computation which takes time proportional to $\frac{1}{x}$ is exposed.

We then provide alternative algorithms which have lower relative error as well as lower (and bounded) computation. For the quantile functions, the number of Newton-Raphson iterations is reduced by a factor of 6, with all convergence failures removed. The number of terms needed to evaluate the CDF/SF for $\lim_{n\to\infty} \sqrt{n}D_n$ is also reduced by a factor of 6, and the relative error of the results improved, often by orders of magnitude.

This paper is organized as follows. Sect. 2 provides a quick review of Kolmogorov-Statistics with special emphasis on the formulae needed for computation. Sect. 3 provides an analysis of the formulae for computing the CDF/SF of the two-sided $\lim_{n\to\infty} \sqrt{n}D_n$. Sect. 4 then analyses the SciPy implementation, and provides an alternate recipe for computing the CDF/SF. This is followed by an analysis Sect. 5 and recipe Sect. 6 for the quantile function. Sect. 7 provides numeric results showing the change in performance resulting from use of these algorithms, along with interpretation of results.

The formulae for computing the SF/CDF/PDF have been available for quite some time. The novelty in this work is the analysis of the SciPy implementation and the details of the recipes, especially for the quantile functions. Explicit formulae for brackets containing the root are provided which enable root-finding algorithms to proceed with many fewer iterations.

The "C" code computing the CDF & SF for this distribution was written quite some time ago, when computers had considerably slower clock speeds and sample sizes were considerably smaller than they are today. To a user of the software, the answers may have seemed plausible for most real-world inputs.

2 Review of Kolmogorov-Smirnov Statistics

In 1933 Kolmogorov [2, 3] introduced the the empirical cumulative distribution function (ECDF) for a (real-valued) sample $\{Y_1, Y_2, \ldots, Y_n\}$, each Y_i having the same continuous distribution function $F(Y)$. He then enquired how close this ECDF would

be to $F(Y)$. Formally he defined

$$F_n(y) = \frac{1}{n} \# \{i : Y_i <= y\} \tag{1}$$

$$D_n = \sup_y |F_n(y) - F(y)| \tag{2}$$

After wondering whether $\mathsf{P}\{D_n \leq \epsilon\}$ tends to 1 as $n \to \infty$ for all ϵ, he then answered affirmatively with the asymptotic result [2, 3]

$$\lim_{n \to \infty} \mathsf{P}\{D_n \leq x\, n^{-1/2}\} = L(x) = 1 - 2\sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2 x^2} \tag{3}$$

Kolmogorov's proof used methods of classical physics. Feller [4, 5] provided a more accessible proof in English.

Smirnov [6, 7] instead used the one-sided values $D_n^+ = \sup_y (F_n(y) - F(y))$ and $D_n^- = \sup_y (F(y) - F_n(y))$ and showed that they also had a limiting form

$$\lim_{n \to \infty} \mathsf{P}\{D_n^+ \leq x\, n^{-1/2}\} = \lim_{n \to \infty} \mathsf{P}\{D_n^- \leq x\, n^{-1/2}\} = 1 - e^{-2x^2} \tag{4}$$

Magg & Dicaire[8] gave a tightened asymptotic. For a fixed x

$$\mathsf{P}\{D_n^+ \leq x\} \underset{n \to \infty}{\asymp} 1 - \exp\left(\frac{-(6nx+1)^2}{18n}\right) \tag{5}$$

Fig. 1 illustrates the ECDF, and the construction of D_n, D_n^+ and D_n^- for one sample. The distributions of D_n^+ and D_n^- are the same. The distributions of D_n and D_n^+ are related, as $D_n = \max D_n^-, D_n^+$, and hence $\mathsf{P}\{D_n \geq x\} = 2 * \mathsf{P}\{D_n^+ \geq x\}$ for all $x \geq 0.5$.
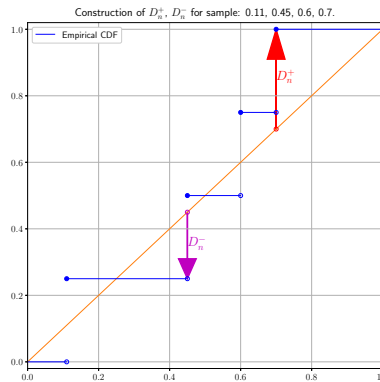

Figure 1: Construction of Kolmogorov-Smirnov statistics for $n = 4$.

For the purpose of showing that the ECDF approaches $F(Y)$, these limit formulae are sufficient. Later authors turned this around and used the D_n statistic as a measure of "goodness-of-fit" between the sample and $F(Y)$, for *any* distribution function $F(Y)$. It

is clear that a large value for any of D_n, D_n^+ and D_n^- may be indicative of a mismatch. But a too-small value may also be cause for concern, as the fit may "too good". In order to use D_n for this goodness-of-fit purpose, knowledge of the distribution of D_n is needed.

Determining the exact distribution of the two-sided D_n is non-trivial. Birnbaum [9] showed how to use Kolmogorov's recursion formulas to generate exact expressions for $\mathsf{P}(D_n \leq x)$, $n \leq 6$. Durbin [10] provided a recursive formula to compute $\mathsf{P}(D_n \leq x)$ (implemented by Marsaglia, Tsang and Wang [11], made more efficient by Carvalho [12]), which involved calculating a particular entry in a potentially large matrix raised to a high power. Pomeranz [13] provided another formulation which involved calculating a specific entry in a large-dimensional matrix. Drew Glen & Leemis [14] generated the collection of polynomial splines for $n <= 30$. Brown and Harvey [15, 16, 17] implemented several algorithms in both rational arithmetic and arbitrary precision arithmetic. Simard and L'Ecuyer [18] analyzed all the known algorithms for numerical stability and sped.

For the one-sided statistics the situation is much cleaner. An exact formula was discovered early-on [6, 19, 20]

$$\mathsf{P}(D_n^+ \leq x) = 1 - S_n(x) \tag{6}$$

where

$$S_n(x) = x \sum_{j=0}^{\lfloor n(1-x) \rfloor} \binom{n}{j} \left(x + \frac{j}{n}\right)^{j-1} \left(1 - x - \frac{j}{n}\right)^{n-j} \tag{7}$$

$$\tag{8}$$

$S_n(x)$ is a sum of relatively simple n-th degree polynomials, forming a spline with knots at $0, \frac{1}{n}, \frac{2}{n}, \ldots, 1$. This has made computations involving $S_n(x)$ a somewhat easier task, though the simplicity can be a little misleading [21].

3   Computation of the Survival Function

The scipy.special subpackage of the Python SciPy package provides two functions for computations of the limiting $\sqrt{n}D_n$ distribution. `kolmogorov(n, x)` computes the Survival Function $S_n(x)$ for D_n^+ and `kolmogi(n, p)` computes the Inverse Survival Function. The source code for the computations is written in "C", and are performed using the IEEE 754 64 bit double type (53 bits in the significand, and 11 bits in the exponent.)

The Survival Function is implemented directly as

$$K(x) = 1 - L(x) = \lim_{n \to \infty} \mathsf{P}\{\sqrt{n}D_n \geq x\} = 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2 x^2} \tag{9}$$

3.1   Evaluating near x=0

On the face of it, this appears to be a great series to be summing. It is an alternating series. The exponent involves $-k^2$. And indeed for large x, the series converges quite

quickly, needing only a few terms. But for small x, the situation is different. For $x = 0.01$ the first few terms in the series are: $0.999800 - 0.999200 + 0.998201 - 0.996805 + 0.995012 - 0.992825 + 0.990247 - 0.987281 + 0.983930 - \ldots$. After summing enough terms, the result should be 0.5 (actually $0.5 - 3.2 \times 10^{-5356}$.) Instead the sum is computed as $0.50000000000000044 (= 0.5 + 2^{-51})$ hence the calculated value of $K(x)$ is $1.0 + 2^{-50} > 1$. This particular computed SF value has a *very* small relative error, perhaps surprisingly small given that it has added up 400 terms and the condition number of the sum is 124, However the value is outside the range of K, and using it to compute the CDF has a rather large relative error.

Writing $q = e^{-2x^2}$, the sum is

$$K(x) = 2\sum_{k=1}^{\infty}(-1)^{k-1}q^{k^2} = q - q^4 + q^9 - q^{16}\ldots \tag{10}$$

The terms are certainly all smaller than 1, decreasing in size, with $\log(k\text{th term}) = k^2 \log(1\text{st term})$. However if the 1st term is too close to 1, then the terms decrease in magnitude *very* slowly: $q^{k^2} < \epsilon \iff k > \frac{\sqrt{\log\frac{1}{\epsilon}}}{\sqrt{2}x}$. For $\epsilon = 10^{-16}$ say, a reasonable number for 64-bit floating point operations, $k > \frac{4}{x}$ to get to terms that are that small. In general, the number of terms needed is inversely proportional to x. For $x = 0.01$, that requires summing over 400 mixed-sign terms, which provides many operations for rounding errors to accumulate, in addition to the loss of accuracy due to subtractive cancellation. It is clear that a lot of precision is required to calculate an accurate value of $L(0.01)$.

3.2 Combining adjacent terms

One approach is to pair up the terms as per Monahan [22]. This leads to

$$\sum_{j=1}^{\infty}q^{(2j-1)^2}(1 - q^{4j-1}) = q(1 - q^3) + q^9(1 - q^7) + \ldots \tag{11}$$

The example summation above would then become: $0.000600 + 0.001397 + 0.002187 + 0.002966 + 0.003732 + 0.004480 + \ldots$ The terms are all positive, which provides some added stability, but still involves many terms. This paired formulation actually requires more terms than the original Eq. (10) in order for the terms to decrease enough in size, but it has addressed the subtractive cancellation and now only suffers a precision issue.

Taking the combining a a step further, one can rewrite the sum as an infinite Horner method:

$$K(x) = 2q(1 - q^3(1 - q^5(1 - q^7(1 - \ldots)))) \tag{12}$$

Dropping the terms $q^{(2j+1)}(1 - \ldots)$ and beyond has an error less than $q^{(j+1)^2}$. There is no difference in the number of terms required, but the powers of q involved are much lower, and truncating the computation provides an obviously non-negative answer.

3.3 Alternate formulation via functional equation of Theta functions

The Kolmogorov probability can be expressed as a (Jacobian) theta function [4]:

$$\mathsf{P}\{\sqrt{n}D_n \le x\} = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2} = \theta\left(z = \frac{1}{2}; \tau = \frac{2ix^2}{\pi}\right) \quad (13)$$

where

$$\theta(z; \tau) = \sum_{k\in\mathbb{Z}} e^{\pi i k^2 \tau + 2\pi i k z} \quad \text{for } \tau \in \mathbb{H}, \text{ the complex upper plane} \quad (14)$$

This theta function has a remarkable functional equation ([23, 24]), a simple form of which is:

$$\theta\left(\frac{z}{\tau}; \frac{-1}{\tau}\right) = e^{-\pi i/4}\sqrt{\tau}e^{\frac{\pi i z^2}{\tau}}\theta(z; \tau) \quad (15)$$

After substituting values for z, τ and some simplification we arrive at

$$\mathsf{P}\{\sqrt{n}D_n \le x\} = L(x) = \frac{\sqrt{2\pi}}{x}\sum_{k=1}^{\infty} e^{\frac{-(2k-1)^2\pi^2}{8x^2}} \quad (16)$$

$$= \frac{\sqrt{2\pi}}{x}\sum_{\substack{n\in\mathbb{Z}^+ \\ n \text{ odd}}} t^{n^2} \quad \text{where } t = e^{-\pi^2/8x^2} \quad (17)$$

This new formulation also contains a sum of some powers: $L(x) = t + t^9 + t^{25} + t^{49} + \ldots$. The difference in outcome is that for small x, the t in this summation is *much* smaller than the q in Eq. (10), so these powers of t become negligible after very few terms. For $x = 0.01$ the first few terms are: $1.278\mathrm{E}{-5358} + 9.105\mathrm{E}{-48222} + \ldots$ I.e. a single term is sufficient. In general,

$$t^{n^2} < \epsilon \iff n > \sqrt{\frac{-8\log\epsilon}{\pi^2}}\, x \quad (18)$$

For $\epsilon = 10^{-16}$ say, $n > 6x$ will ensure small terms. In fact, a single term will suffice for many x! Because all these terms are positive, the sum is positive, and lies in the interval $[0, 1]$. Combining terms in an infinite Horner method leads to an effective computation formulation

$$L(x) = \frac{\sqrt{2\pi}}{x}t(1 + t^8(1 + t^{16}(1 + t^{24}(1 + \ldots)))) \quad (19)$$

Fig. 2 shows a plot of $K(x)$ and the two approximations arising from taking just the first term in the two series Eq. (9) and Eq. (16). The approximations have different regions of quality, which fortunately overlap.
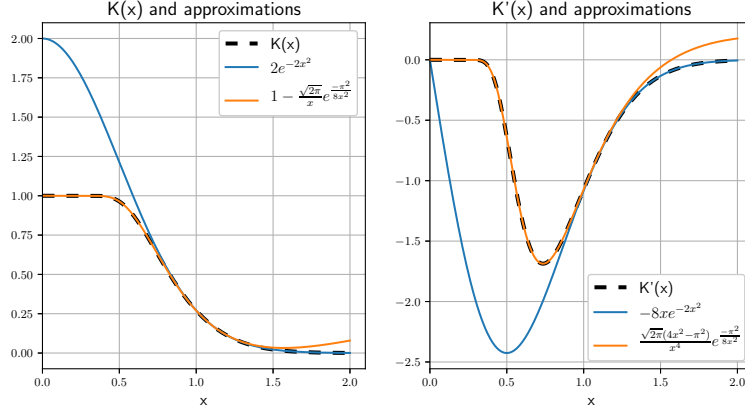
Figure 2: $K(x)$ and two approximations; its derivative $K'(x)$ and two approximations.

3.4 Evaluation of the PDF

The PDF can be evaluated by differentiating either Eq. (9) or Eq. (16)

$$PDF(x) = \frac{d \lim_{n\to\infty} \mathsf{P}\{\sqrt{n}D_n \geq x\}}{dx} \tag{20}$$

$$= -K'(x) = L'(x) \tag{21}$$

$$-K'(x) = 8x \sum_{k\in\mathbb{Z}^+} (-1)^{k-1} k^2 e^{-2k^2x^2} \tag{22}$$

$$= 8x(q - 4q^4 + 9q^9 - 16q^{16}\dots) \tag{23}$$

$$= 8xq(1 - q^3(4 - q^4(9 - q^7(16 - \dots)))) \tag{24}$$

$$L'(x) = \frac{\sqrt{2\pi}}{4x^4} \sum_{\substack{n\in\mathbb{Z}^+ \\ n \text{ odd}}} (-4x^2 + \pi^2 n^2) e^{\frac{-\pi^2 n^2}{8x^2}} \tag{25}$$

$$= \frac{-L(x)}{x} + \frac{\sqrt{2}\pi^{5/2}}{4x^4}(t + 9t^9 + 25t^{25} + 49t^{49} + \dots) \tag{26}$$

$$\tag{27}$$

$$= \frac{-L(x)}{x} + \frac{\sqrt{2}\pi^{5/2}}{4x^4}t(1 + t^8(9 + t^{16}(25 + t^{24}(49 + \dots)))) \tag{28}$$

Neither sum (Eq. (24), Eq. (27)) has a high condition number when restricted to its appropriate regions. The cost of evaluating $K'(x)$ is a little more the same cost of evaluating $K(x)$, and the cost of evaluating $L'(x)$ is a little more than the cost of evaluating $L(x)$. Both sums are clearly positive.

7

3.5 The SciPy implementation

The SciPy implementation sums Eq. (10), until $|\frac{q^{n^2}}{\text{partial sum}}| < 10^{-16}$, with no recombination or use of alternate formulations. For x large enough, say $x > 0.75$ this works perfectly well.

- As $x \downarrow 0$, the computation suffers from more and more accuracy loss due to subtractive cancellation. For some values of x, the returned value is greater than 1.0.

- $K(x)$ is *a priori* a monotonically decreasing function of x, but the calculated values are far from monotonic. As $x \downarrow 0$, the number of terms kept in the summation changes frequently (as it is inversely proportional to x.) If the number of terms changes from even to odd, $K(x)$ jumps up. Similarly if the number of terms becomes even, $K(x)$ jumps down. Every time there is a switch, the monotonicity is lost (and also continuity!), so the results oscillate. Root-finders appreciate the monotonicity as it makes that task easier. In this case a lack of it means that there may not be a solution to $K(x) - p = 0$ inside an interval even though the value of the function at the endpoints have opposite signed values. Another use of the ISF is to generate random variables from the distribution, given a random value in $[0, 1]$. For this it is is desirable to have the ISF be monotonic. If the SF isn't monotonic then it's likely the ISF isn't monotonic either.

- SciPy doesn't provide a separate function to compute the PDF. Instead it numerically differentiates the CDF $1 - K(x)$. This involves multiple evaluations of $K(x)$, and often returns negative values for $x \in [0, 0.2]$.

4 Algorithm for computing Kolmogorov SF, CDF and PDF

Here we propose an algorithm to compute the CDF/SF of the limit of the Kolmogorov two-sided statistic within a specified tolerance ϵ, which addresses the issues discovered. It is an extension of Monahan's SF algorithm [22], to also cover the PDF and an arbitrary tolerance.

Algorithm 1 (kolmogorov). *Compute the Kolmogorov SF, CDF and PDF for real x.*

```
function [SF, CDF, PDF] = kolmogorov(x:real, ϵ:real)
```

Step 1 If $x \leq 0.82$, set

$$t \leftarrow \exp\left(\frac{-\pi^2}{8x^2}\right)$$

$$U \leftarrow \exp\left(\frac{-\pi^2}{x^2}\right)$$

$$R \leftarrow \left\lfloor \sqrt{-2\log(\epsilon)} * \frac{x}{\pi} + 1 \right\rfloor$$

$$S_R, D_R \leftarrow 1, (2R+1)^2$$

Then loop over $r \leftarrow R, R-1, \ldots, 1$:

$$S_{r-1} \leftarrow 1 + U^r * S_r$$

$$D_{r-1} \leftarrow (2r-1)^2 + U^r * D_r$$

Finally set

$$\texttt{CDF} \leftarrow \sqrt{2\pi} * t * S_0/x$$

$$\texttt{SF} \leftarrow 1 - \texttt{CDF}$$

$$\texttt{PDF} \leftarrow \sqrt{2\pi} * t * \left(\pi^2 D_0/(4x^2) - S_0\right)/x^2$$

Step 2 If $x > 0.82$, set

$$q \leftarrow \exp\left(-2x^2\right)$$

$$R \leftarrow \left\lfloor \frac{\sqrt{-2\log(\epsilon)}}{3x} \right\rfloor$$

$$S_R, D_R \leftarrow 1, (R+1)^2$$

Then loop over $r \leftarrow R, R-1, \ldots, 1$:

$$S_{r-1} \leftarrow 1 - q^{2r+1} * S_r$$

$$D_{r-1} \leftarrow r^2 - q^{2r+1} * D_r$$

Finally set

$$\texttt{SF} \leftarrow 2 * q * S_0$$

$$\texttt{CDF} \leftarrow 1 - \texttt{SF}$$

$$\texttt{PDF} \leftarrow 8 * q * x * D_0$$

Step 3 Clip $\texttt{SF}$, $\texttt{CDF}$ to lie in the interval $[0,1]$ and $\texttt{PDF}$ to lie in $[0, \infty)$. Return $[\texttt{SF},$ $\texttt{CDF}, \texttt{PDF}]$.

4.1 Remarks

Specifying the API to return both the CDF and SF probabilities enables the returned values to retain as much accuracy as had been computed. It also enabled computing the CDF probabilities more accurately for values of x close to 0.

- If $x < 0.82$, Step 1 computes $p_{CDF} = L(x)$ using the series obtained from the functional equation for theta functions. If $x >= 0.82$, Step 2 computes $p_{SF} = K(x)$. This algorithm ensures that the values returned are between 0 and 1, with few items summed. The cutoff of 0.82 is approximately the median of the distribution, so that the computation used will compute the smaller of $\{p_{SF}, p_{CDF}\}$, hence not incur loss if the complement needs to be returned. For the smallest number of terms kept in the summation, the cutoff should be slightly higher, around $1.10 - 1.15$. For the lowest error, the cutoff could be different for the CDF and the SF.

- Calculation of the number of iterations required for the loop can be avoided by always looping the maximum number of times, which is 2 for Step 1 and 4 for Step 2. The loops can then be unrolled and the powers replaced with explicit squarings or cubings.

5   Evaluation of the Inverse Survival Function

Given a survivor probability p_{SF}, it is often desirable to know the x that corresponds to it. There is no nice formula to invert $K(x)$, to go from a probability p_{SF} back to a (scaled) difference x.

One way to numerically approximate the root of $K(x) = p_{SF}$ is to first solve for q:

$$\frac{p_{SF}}{2} = p = q - q^4 + q^9 - q^{16} \dots \tag{29}$$

and then set

$$x = \sqrt{\frac{-\log(q)}{2}} \tag{30}$$

(or use this x as the starting point for solving the original $K(x) = p_{SF}$).

Instead of addressing the full infinite sum, one can further approximate by truncating after n terms. Solve

$$p = f_n(q) = \sum_{k=1}^{n}(-1)^{k-1}q^{k^2} = q - q^4 + \dots + (-1)^{n-1}q^{n^2} \tag{31}$$

Fig. 3 shows the graphs of the first few of these truncations. All the curves start at the lower left $(q, p) = (0, 0)$, and head towards a height of $p = 0.5$. The f_n for even n top out a little below $p = 0.5$, and then turn down to $(q, p) = (1, 0)$. The f_n for odd n try to hug $p = 0.5$ as long as they can, and then shoot up to $(q, p) = (1, 1)$. Only the first truncation, f_1, deviates from the plot of $K(x(q))$ far from $p = 0.5$.

When solving first for q then setting x, a relative change of δ in q results in a relative change in x of $\frac{q\frac{dx}{dq}}{x}\delta = \frac{1}{\log q}\delta = \frac{-1}{4x^2}\delta$. This can be used to guide the required tolerance when solving for q, or provide an estimate of the error in $x = x(q)$ and hence how much work needs to be done to polish it up.
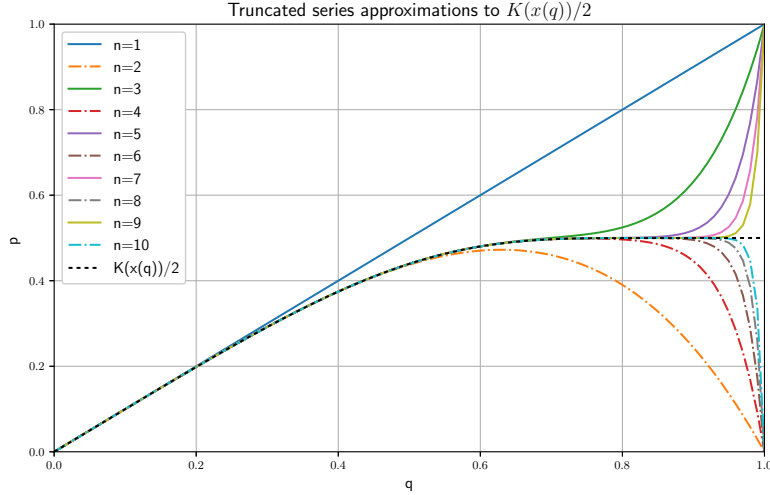
10

Figure 3: Truncations to $K(x(q))/2$ after n terms

5.1 Use of root-solvers on f_n

Inverting f_1 as a function is clearly trivial, but already inverting $f_2(q) = q - q^4$ requires some thought. The maximum height of $f_2(q)$ is $3 \cdot 4^{-4/3} \approx 0.4724$ occurring at $q = 4^{-1/3}$, so not all the values of interest of $p \in [0, \frac{1}{2}]$ have a corresponding q.

One can use any polynomial root-finder to solve for $f_n(q) = p$. The non-zero roots of $f_n(z) = 0$ lie close to the unit circle in the complex plane, and for small p the roots of $f_n(z) = p$ are just small perturbations of these. One or two steps of Laguerre iteration, or two or three steps of Newton-Raphson, provide a really good estimate q_n solving $f_n(q_n) = p$. This in turn approximates the root of the original problem with small relative error (less than 1%), at least until p gets up to about 0.45. For these higher values of p, n needs to be larger and larger in order to generate a good estimate for the original problem.

5.2 Analytical inversion

Another approach is to treat the polynomial $f_n(z)$ as a function to $\mathbb{C}$ from the unit disk $D(0,1)$ inside $\mathbb{C}$. The function $f_n(z)$ is analytic, has non-zero derivative at $z = 0$, and $f_n(z) = 0$ hence there is a disk $D(0,r)$ and an analytic function $g(w)$, such that $f(g(w)) = w$ for all $w \in D(0,r)$ (and $g(f(z)) = z$).

One way to find such a disk and g is to treat the polynomial $f_n(q)$ as a formal power series, formally revert it and then analyze its region of convergence. For $f_2(q) = q - q^4$, the formal power series reversion is

$$q = \sum_{k=0}^{\infty} \binom{4k}{k} \frac{p^{3k+1}}{3k+1} = p + p^4 + 4p^7 + 22p^{10} + 140p^{13} + 969p^{16} + \dots \quad (32)$$

This series converges for $|p| < 3 \cdot 4^{-4/3} \approx 0.4724$. [If $p = 0$, there are 4 solutions to $p = q - q^4 = q(1 - q^3)$, namely $q \in \{0, 1, \omega, \omega^2\}$, where $\omega = e^{-2\pi i/3}$ is a primitive cube root of 1. For small p, the series Eq. (32) returns the root closest to 0. As p increases, this root increases towards 1, and the root which starts at 1 decreases towards 0. For $p = 3 \cdot 4^{-4/3}$, these two roots meet at $q = 4^{-1/3}$, and the power series reversion no longer converges. This also happens to be where the derivative $f_1'(q) = 1 - 4q^3 = 0$. For $p > 3 * 4^{-4/3}$ all roots are complex.]

n	$f_n^{-1}(p)$
1	p
2	$p + p^4 + 4p^7 + 22p^{10} + 140p^{13} + 969p^{16} + \dots$
3	$p + p^4 + 4p^7 - p^9 + 22p^{10} - 13p^{12} + 140p^{13} - 136p^{15} + 969p^{16} \dots$
4	$p + p^4 + 4p^7 - p^9 + 22p^{10} - 13p^{12} + 140p^{13} - 136p^{15} + 970p^{16} \dots$

Table 1: Formal power series inversion/reversion f_n for $n = 1, 2, 3, 4$.

Table 5.2 lists the first few terms for each of the first few f_n, with changing coefficients highlighted. The terms of f_n^{-1} agree with those of f_{n-1}^{-1} up to p^{n^2-1}. Inverting the full q-series

$$p = q - q^4 + q^9 - q^{16} + q^{25} - q^{36} + q^{49} - \dots \tag{33}$$

leads to

$$
\begin{aligned}
q = {} & p + p^4 + 4p^7 - p^9 + 22p^{10} - 13p^{12} + 140p^{13} - 136p^{15} + 970p^{16} \\
& + 9p^{17} - 1330p^{18} + 7104p^{19} + 231p^{20} - 12650p^{21} + 54096p^{22} \\
& + 3900p^{23} - 118780p^{24} + 423890p^{25} + 54810p^{26} - 1108380p^{27} + \dots
\end{aligned}
\tag{34}
$$

The coefficients are all integers but they are not bounded, so the convergence properties of this expression are non-trivial to analyze. The radius of convergence r is no greater than 0.5, hence the $n-$th coefficient must be about as big as $const \times 2^n$ infinitely often.

While there is no universal formula to invert $2p = K(x)$, truncating this series Eq. (34) and using for $p = \frac{p_{SF}}{2} \ll 0.5$ is reasonable.

5.3 Bracketing the root

Regardless of whether a good estimate of the root is available, an interval bracketing the root is useful as a guide/constraint on any numerical root-finders. Starting from

$$p_{SF} = 2q(1 - q^3 + q^8 + \dots) \tag{35}$$

rearranging as

$$2q = \frac{p_{SF}}{1 - q^3 + q^8 + \dots} \tag{36}$$

and truncating the sum in the denominator, the following chain of inequalities emerge

$$\frac{p_{SF}}{1} \leq \frac{p_{SF}}{1 - q^3 + q^8} \leq \dots \leq \quad 2q \quad \leq \dots \leq \frac{p_{SF}}{1 - q^3 + q^8 - q^{15}} \leq \frac{p_{SF}}{1 - q^3} \tag{37}$$

If X_a is any fixed positive real number and $x >= X_a$, then $q = e^{-2x^2} < e^{-2X_a^2}$, and Eq. (37) provides bounds on $2q$ and hence on x. Taking X_a to be median of the distribution, $X_a = K^{-1}(0.5) \approx 0.82757$, with corresponding $q_a \approx 0.254 \approx e^{-1.37}$

$$\sqrt{-0.5 \log\left(\frac{p_{SF}}{2}\right)} \geq x \geq \sqrt{-0.5 \log\left(\frac{p_{SF}}{2(1 - e^{-4})}\right)} \qquad \text{for } p_{SF} \leq 0.5 \quad (38)$$

The upper bound is independent of any choice of X_a, but becomes less useful as $p_{SF} \to 1$.

5.4 Inverting for survivor probabilities close to 1

For p_{SF} close to 1, the approach taken in Eq. (34) is not practical, but solving Eq. (16) is.

$$p_{CDF} = L(x) = \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} t^{(2k-1)^2} \qquad (39)$$

Before applying Newton-Raphson to this we note that

$$L'(x) = \frac{\sqrt{2\pi}}{4x^4} \sum_{\substack{n \in \mathbb{Z}^+ \\ n \text{ odd}}} (-4x^2 + \pi^2 n^2) t^{n^2} \qquad (40)$$

$$L''(x) = \frac{\sqrt{2\pi}}{16x^7} \sum_{\substack{n \in \mathbb{Z}^+ \\ n \text{ odd}}} \left(32x^4 - 20\pi^2 n^2 x^2 + \pi^4 n^4\right) t^{n^2} \qquad (41)$$

As $x \to 0$, $\frac{L''(x)}{2L'(x)} \sim \frac{\pi^2}{8x^3}$, which is unbounded. Since the errors of each N-R step approximately follow $e_{n+1} \sim \frac{L''(x)}{2L'(x)} e_n^2$, the initial error must already be very small in order to achieve rapid convergence. Unfortunately there is no obvious good initial estimate for x or t. Rewrite Eq. (39) as

$$t = \frac{x p_{CDF}}{\sqrt{2\pi}} \left(1 + t^8 + t^{24} + t^{48} + \ldots\right)^{-1} \qquad (42)$$

Both t and $L(x)$ are monotonically increasing functions of x. If X_a, X_b are any fixed positive real numbers and $X_a \leq x \leq X_b (\iff L(X_a) \leq p_{CDF} \leq L(X_b))$, then

$$\frac{X_a p_{CDF}}{\sqrt{2\pi}} \frac{1}{1 + t_b^8 + t_b^{24} \ldots} \leq t \leq \frac{X_b p_{CDF}}{\sqrt{2\pi}} \frac{1}{1 + t_a^8 + t_a^{24} \ldots} \qquad (43)$$

In particular, taking $X_b = 1 (p_b \approx 0.73, t_b \approx 0.29)$, $X_a = 0.0406 (t_a \approx 1 \times 10^{-325})$

$$\left. \begin{array}{c} \dfrac{0.04 * p_{CDF}}{\sqrt{2\pi}} \leq t \leq \dfrac{p_{CDF}}{\sqrt{2\pi}} \\[2em] \implies \dfrac{\pi}{\sqrt{-8 \log\left(\frac{0.04 * p_{CDF}}{\sqrt{2\pi}}\right)}} \leq x \leq \dfrac{\pi}{\sqrt{-8 \log\left(\frac{p_{CDF}}{\sqrt{2\pi}}\right)}} \end{array} \right\} \qquad \text{for } p_{CDF} \leq 0.73$$

$$(44)$$

($X_a = 0.040596694\ldots$ was used as $L(X_a) = 2^{-1073}$, so $L(x)$ is too small to be represented in 64 bits for any $x < X_a$.) The upper bound is reasonable, but the lower bound is not so useful.

To find a better lower bound, drop all but the first summation term of $L(x)$ and solve

$$p_{CDF} = \frac{\sqrt{2\pi}}{x}t = \frac{\sqrt{2\pi}}{x}e^{\frac{-\pi^2}{8x^2}} \tag{45}$$

which is equivalent to finding the fixed point x_p of

$$g_p(x) \triangleq \frac{\pi}{\sqrt{-8\log\left(\frac{p_{CDF}*x}{\sqrt{2\pi}}\right)}} \tag{46}$$

The function is contractive around the fixed-point x_p and the derivative there satisfies

$$g_p'(x_p) = \frac{4x_p^2}{\pi^2} \ll 1 \tag{47}$$

Starting with any value of x and iterating with g_p generates a monotonic sequence of values $\{x, g_p(x), g_p(g_p(x)), \ldots\}$ converging to x_p. If p_{CDF} is so small that only the first term in the summation contributes to the answer in machine precision, then x_p is also the solution to $L(x) = p$. [That occurs whenever $t^8 < \epsilon \iff x < 0.523 \iff p_{CDF} < 0.0529$ for the 64-bit floats.] Starting with any upper bound for $L^{-1}(p_{CDF})$ (E.g. $x = 1$), all the iterates will be upper bounds not only for x_p but also $L^{-1}(p)$. [This is the same as the upper bound of Eq. (44).] Starting with any of $x \in \{p_{CDF}, \sqrt{p_{CDF}}, 0.04\}$, the first few iterates will still be smaller than $L^{-1}(p_{CDF})$, so can be used for bracketing purposes.

Given the bracket, the bracket midpoint can be used as the starting point for methods such as N-R. Though there may be some simple approximations for certain sub-intervals which lead to rapid convergence. One example is:

$$t \approx 0.23530414p^2 + 0.2136641p - 0.00076411 \qquad \text{for } 0.1 \le x \le 0.5 \tag{48}$$

5.5   The SciPy implementation

The function kolmogi(p_{SF}), in SciPy 's `special` sub-package returns an estimate of $K^{-1}(p_{SF})$. It uses the Newton-Raphson method (without bracketing), approximates the derivative using just the first term of Eq. (9) ($K'(x) \approx -8xe^{-2x^2}$), and halts whenever the relative change in the estimate is small enough ($|(x_{n+1} - x_n)/x_{n+1}| < 1 \times 10^{-10}$), or the number of iterations exceeds 500. The starting point for the N-R iterations is generated by using f_1^{-1} to generate q_0, then $x_0 = \sqrt{\frac{-\log(q_0)}{2}}$.

- This works well whenever p_{SF} is small. In that situation, the first term in the summation dominates, so that $K(x) \approx 2q$. Even though x_0 is greater than the desired root, the overshoot is small enough that the iterate stays in domain.

- For p_{SF} close to 1, the initial estimate x_0 is no longer close to the root, but this in itself is not a problem that the N-R couldn't overcome. However the estimate used in place of $K'(x)$ is quite different from the actual value of $K'(x)$, as shown in Fig. 2. In particular, the true derivative is close to 0, while the approximation to the derivative may be orders of magnitude larger. The result is that the adjustment made in each N-R step is *much* too small. This affects the convergence: not just the rate of convergence when the algorithm does converge to the actual root, but it gives the appearance of convergence even when still far from the root. In particular, as $p_{SF} \rightarrow 1$, kolmogi(p_{SF}) wouldn't return any number lower than 0.32, even though 0.18 should be achievable with 64 bits.

- The slow rate of convergence also quadratically affected the amount of computation needed. For p_{SF} close to 1, kolmogi(p_{SF}) required many iterations of N-R, each of which made a call to kolmogorov(x), and that in turn used many terms in its summation (as the number of terms $\propto \frac{1}{x}$). The net effect was that a single call to kolmogi(p_{SF}) could generate 5000 calls to `exp`.

6 Algorithm for Computing Kolmogorov Quantile

Next we propose modifications to the existing algorithm which will find x such that kolmogorov(x) = p within the specified tolerances.

Algorithm 2 (kolmogi). *Compute the Kolmogorov* ISF/PPF *for real p_{SF}, p_{CDF}.*

```
function [X] = kolmogi(pSF:real, pCDF:real)
```

Step 1 Immediately handle $p_{CDF} = 0$, $p_{SF} = 0$ as special cases, returning $X \leftarrow 0$ or ∞ respectively.

Step 2 Set
$$f(x) = \begin{cases} \texttt{kolmogorov(x).SF} - p_{SF} & \text{if } p_{SF} <= 0.5 \\ p_{CDF} - \texttt{kolmogorov(x).CDF} & \text{otherwise} \end{cases}$$

Step 3 If $p_{SF} \leq 0.5$, set

$$P \leftarrow \frac{p_{SF}}{2} \tag{49}$$

$$[Q_A, Q_B] \leftarrow \left[P * \frac{1}{1.0 - e^{-4}}, P \right] \tag{50}$$

$$Q_0 \leftarrow P + P^4 + 4P^7 - P^9 + 22P^{10} - 13P^{12} + 140P^{13} \tag{51}$$

$$[A, B] \leftarrow \left[\sqrt{\frac{-\log(Q_A)}{2}}, \sqrt{\frac{-\log(Q_B)}{2}} \right] \tag{52}$$

$$X_0 \leftarrow \sqrt{\frac{-\log(Q_0)}{2}} \tag{53}$$

Skip to Step 5.

Step 4 If $p_{SF} > 0.5$, set

$$[A_0,\ B_0] \leftarrow [\max(\sqrt{p_{CDF}}, 0.04),\ 1] \tag{54}$$

$$[A,\ B] \leftarrow [g_p(g_p(A_0)),\ g_p(g_p(B_0))] \tag{55}$$

$$X_0 \leftarrow \begin{cases} 0.2353p^2 + 0.2136p - 0.000764 & \text{if } p_{CDF} >= 0.1 \\ \frac{A+B}{2} & \text{otherwise} \end{cases} \tag{56}$$

Proceed to Step 5.

Step 5 Perform iterations of bracketed N-R with function f, starting point X_0 and bracketing interval $[A, B]$, using the actual derivative of $f(x)$, until the desired tolerance is achieved, or the maximum number of iterations is exceeded. Set $X \leftarrow$ the final N-R iterate. Return X.

6.1 Remarks

SciPy does contain multiple root-finders but we avoid using them here. The code for `kolmogorov` is written in C as part of the `cephes` library in the `scipy.special` subpackage. In order to enable an implementation of this K-S algorithm to remain contained within this subpackage, we only use a bracketing Newton-Raphson root-finding algorithm, as this can be easily implemented.

Changing the API (2) enables the code to use which ever probability allows the greatest precision, which will usually be the smaller of the two. It also enables computing x more accurately for values of p_{SF} close to 1.

- If p_{SF} and p_{CDF} are both non-zero, the root-finding will use a bracketed Newton-Raphson algorithm.

- In Step 2, both expressions for $f(x)$ would compute the same answer if using infinite precision — any difference between them should be approximately the order of the machine epsilon.

- For small p_{SF} (which corresponds to $x \gtrsim 0.82$) Step 3 determines a good bracket and a good initial estimate X_0. The brackets come from Eq. (37), the initial estimate from Eq. (34).

- In practice, it's been found that Eq. (50) is a little tight for some machine architectures/library implementations when dealing with very small p_{SF}, and Q_B should be a little smaller, say $Q_B = \frac{p_{SF}}{2} * (1 - 256 * \epsilon)$, where $\epsilon \approx 2^{-52}$ is the "machine epsilon". Similarly Q_A should be a little larger.

- For large p_{SF} (which corresponds to $x \lesssim 0.82$) Step 4 first determines a good estimate and bracket for q, and then x. The brackets come from Eq. (44), the initial estimate from Eq. (48).

- The N-R iterations require implementing a `kolmogorovp` function to calculate the derivative, which can be done with the obvious minor modifications to `kolmogorov`. f is C^∞ so use of an order 1 method is justifiable.

The main substance to the algorithm is the determination of a suitable bracket and initial starting point. The choice of root-finder to complete the task is less important.

7 Results

7.1 Kolmogorov

The methods compared are the SciPy v0.19 ("Baseline") implementation, and an implementation of Algorithm 2, using $x = 0, (0.001)1.7$.

Table 2 shows some statistics for the number of summation terms used in the computation of the CDF/SF, (Formula 10 or Formula 19) (The mean and std deviations are calculated over the values of x which do not exceed 500 terms. The Failure column is the percentage of values of x that exceeded 500 terms. The Tolerance column lists the percentage of values returned whose relative error exceeded 10^{-9}.)

	mean	std	max	Failure	Tolerance
Baseline	12.4	31	481	0.4%	0.3%
Algorithm 2	2.2	0.9	4	0.0%	0.0%

Table 2: Kolmogorov SF: Summation Terms, Failure and Disagreement Rates

Algorithm 2 typically needs to evaluate just over 2 terms when computing the CDF or SF probabilities within a tolerance of 2.2×10^{-16}. (The maximum relative error in the computed value is actually higher than this, due to errors in $\log/\exp$, and roundoff in the multiplication/summation of the various terms.) This compares to an average of about 12 in the Baseline, which also uses a much higher tolerance of 10^{-10}. The maximum number of iterations is also much reduced to about 4, with no failures to converge observed. Most of the change in the number of iterations is due to using Formula 19.

7.2 Inverting Kolmogorov

The methods compared are the SciPy ("Baseline") implementation, and an implementation of Algorithm 2, using $p = 0, (0.001)1.0$.

Table 3 shows statistics for the number of N-R iterations used in the computations of the ISF/PPF. (The mean and std deviations are calculated over the values of p which do not exceed 500 iterations. The failure is the percentage of values of p that exceed 500 iterations. The tolerance column lists the percentage of values returned whose relative error exceeded 10^{-9}.)

Typically 2-3 iterations are required for convergence within a tolerance of $2^{-52} \approx 2.2 \times 10^{-16}$ using Algorithm 2, compared with 15 iterations (and a much higher tolerance of 10^{-10}) for the Baseline. The maximum is much reduced, with no failures to converge observed. (The maximum relative error in the computed value is potentially higher than 2^{-52}, due to errors in $\log/\exp$, errors in computing $\text{kolmogorov}(x)$ and roundoff in the multiplication/summation of the various terms.)

17

	mean	std	max	fail	tol
Baseline	15.5	27.3	379	0.2%	0.6%
Algorithm 2	2.5	0.9	4	0.0%	0.0%

Table 3: Kolmogorov ISF: Iteration Counts, Failure and Disagreement Rates

As noted in Sect. 5.5, the Baseline computation of $\mathrm{kolmogi}(p)$ increased quadratically as p approached 1, as the number of iterations was essentially unbounded, and each iteration did an increasing amount of work.

- The starting point is close enough to the root that drastic overshoot below $x = 0$ does not occur. And it if did, the use of the bracket would prevent it.

- Using the exact derivative extends the range of $\mathrm{kolmogi}(p_{SF})$ returnable values from $x \in [0.32, \infty)$ to $x \in [0.18, \infty)$ with 64 bit doubles.

- Extending $\mathrm{kolmogi}(p_{SF})$ to $\mathrm{kolmogi}(p_{SF}, p_{CDF})$ allowed small p_{CDF} values to be passed in exactly, rather than as $p_{SF} = 1 - p_{CDF}$, and that extended the range of $\mathrm{kolmogi}(p_{SF}, p_{CDF})$ to $x \in [0.04, \infty)$ for 64 bit doubles.

- If p_{SF} is very close to 0, the number of N-R iterations required for solving $K(x) = p_{SF}$ is actually 0, as the initial estimate $\sqrt{(-0.5 \log(0.5 p_{SF}))}$ is *a priori* within tolerance. For p_{CDF} close to 0 ($p_{CDF} = \{2^{-n} : n = 52, \ldots 1022\}$), typically 2 iterations were required.

- If the probabilities at smaller values of x are needed, it becomes necessary to work with higher precision floats or work with log probabilities, which extend the domain beyond $x \in [0.04, \infty)$. Since $g_p(x)$ only uses the log of p_{CDF} and converges very quickly for very small p_{CDF}, iterating Eq. (46) is an efficient approach.

- Other root-finding algorithms could be used as $K(x)$ is well-behaved, when computed as above. Given a tight starting bracket, Brent's method averaged 5.4 iterations, Sidi's method [25] (with $k = 2$) and False Position with Illinois both averaged about 3.5 iterations. An argument could be made that N-R requires two function evaluations per iteration, for $K(x)$ and $K'(x)$, so that just counting iterations is underestimating the N-R work. The additional work in simultaneously calculating the PDF with the CDF/SF is small, so comparing the number of iterations between methods is reasonable.

8 Summary

In some parts of its domain, the CDF/SF is a sum of many relevant terms. Using an alternate formula, based on Jacobi theta functions, reduces the number of relevant terms to no more than 4.

Using Newton-Raphson successfully to compute the ISF requires a good initial estimate, and a good approximation to the derivative. The contributions of the higher order terms in the Taylor series expansions for the quantiles can make the root-finding a little problematic. We showed how to generate a narrow interval enclosing the root, a good starting value for the iterations, and a way to calculate the derivative with little additional work, so that many fewer N-R iterations are required and the computed values have smaller errors.

References

[1] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–.

[2] A. N. Kolmogoroff. "Sulla Determinazione Empirica di Una Legge di Distribuzione". In: *G. Ist. Ital. Attuari.* 4 (1933), pp. 83–91.

[3] A. Kolmogoroff. "Confidence Limits for an Unknown Distribution Function". In: *Ann. Math. Statist.* 12.4 (Dec. 1941), pp. 461–463. DOI: `10.1214/aoms/1177731684`.

[4] W. Feller. "On the Kolmogorov-Smirnov Limit Theorems for Empirical Distributions". In: *Ann. Math. Statist.* 19.2 (June 1948), pp. 177–189. DOI: `10.1214/aoms/1177730243`.

[5] W. Feller. "Errata: "On the Kolmogorov-Smirnov Limit Theorems for Empirical Distributions"". In: *Ann. Math. Statist.* 21.2 (June 1950), pp. 301–302. DOI: `10.1214/aoms/1177729850`.

[6] N.V. Smirnov. "Näherungsgesetze der Verteilung von Zufallsveränderlichen von empirischen Daten." In: *Usp. Mat. Nauk* 10 (1944), pp. 179–206.

[7] N. Smirnov. "Table for Estimating the Goodness of Fit of Empirical Distributions". In: *Ann. Math. Statist.* 19.2 (June 1948), pp. 279–281. DOI: `10.1214/aoms/1177730256`.

[8] Urs R. Maag and Ghislaine Dicaire. "On Kolmogorov-Smirnov Type One-Sample Statistics". In: *Biometrika* 58.3 (1971), pp. 653–656. DOI: `10.2307/2334402`.

[9] Z. W. Birnbaum. "Numerical Tabulation of the Distribution of Kolmogorov's Statistic for Finite Sample Size". In: *Journal of the American Statistical Association* 47.259 (1952), pp. 425–441. DOI: `10.2307/2281313`.

[10] J. Durbin. "The Probability that the Sample Distribution Function Lies Between Two Parallel Straight Lines". In: *Ann. Math. Statist.* 39.2 (Apr. 1968), pp. 398–411. DOI: `10.1214/aoms/1177698404`.

[11] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. "Evaluating Kolmogorov's Distribution". In: *Journal of Statistical Software, Articles* 8.18 (2003), pp. 1–4. DOI: `10.18637/jss.v008.i18`.

[12] Luis Carvalho. "An Improved Evaluation of Kolmogorov's Distribution". In: *Journal of Statistical Software, Code Snippets* 65.3 (2015), pp. 1–8. DOI: `10.18637/jss.v065.c03`.

[13] John Pomeranz. "Algorithm 487: Exact Cumulative Distribution of the Kolmorgorov-Smirnov Statistic for Small Samples". In: *Commun. ACM* 17.12 (Dec. 1974), pp. 703–704. DOI: `10.1145/361604.361628`.

[14] John H. Drew, Andrew G. Glen, and Lawrence M. Leemis. "Computing the Cumulative Distribution Function of the Kolmogorov-Smirnov Statistic". In: *Comput. Stat. Data Anal.* 34.1 (Aug. 2000), pp. 1–15. DOI: `10.1016/S0167-9473(99)00069-9`.

[15] J. Brown and Milton Harvey. "Rational Arithmetic Mathematica Functions to Evaluate the One-sided One-sample K-S Cumulative Sample Distribution". In: *Journal of Statistical Software, Articles* 19.6 (2007), pp. 1–32. DOI: `10.18637/jss.v019.i06`.

[16] J. Brown and Milton Harvey. "Rational Arithmetic Mathematica Functions to Evaluate the Two-Sided One Sample K-S Cumulative Sampling Distribution". In: *Journal of Statistical Software, Articles* 26.2 (2008), pp. 1–40. DOI: `10.18637/jss.v026.i02`.

[17] J. Brown and Milton Harvey. "Arbitrary Precision Mathematica Functions to Evaluate the One-Sided One Sample K-S Cumulative Sampling Distribution". In: *Journal of Statistical Software, Articles* 26.3 (2008), pp. 1–55. DOI: `10.18637/jss.v026.i03`.

[18] Richard Simard and Pierre L'Ecuyer. "Computing the Two-Sided Kolmogorov-Smirnov Distribution". In: *Journal of Statistical Software, Articles* 39.11 (2011), pp. 1–18. DOI: `10.18637/jss.v039.i11`.

[19] Z. W. Birnbaum and Fred H. Tingey. "One-Sided Confidence Contours for Probability Distribution Functions". In: *Ann. Math. Statist.* 22.4 (Dec. 1951), pp. 592–596. DOI: `10.1214/aoms/1177729550`.

[20] Leslie H. Miller. "Table of Percentage Points of Kolmogorov Statistics". In: *Journal of the American Statistical Association* 51.273 (1956), pp. 111–121. DOI: `10.1080/01621459.1956.10501314`.

[21] Paul van Mulbregt. "Computing the Cumulative Distribution Function and Quantiles of the One-sided Kolmogorov-Smirnov Statistic". In: (Feb. 2018). arXiv: `1802.06966`.

[22] John H. Monahan. *Evaluating the Smirnov distribution function*. Tech. rep. North Carolina State University. Dept. of Statistics, 1989.

[23] E.T. Whittaker and G.N. Watson. *A Course of Modern Analysis*. 4th ed. Cambridge University Press, 1927.

[24] D. Mumford et al. *Tata Lectures on Theta I*. Modern Birkhäuser Classics. Birkhäuser Boston, 2006.

[25] Avram Sidi. "Generalization of the secant method for nonlinear equations." In: *Applied Mathematics E-Notes [electronic only]* 8 (2008), pp. 115–123.